Chapter - 4

LDPC and SHA Based

Iris Recognition for Image Authentication

4.1 Introduction

In recent years, the use of iris for human identification has significantly grown due to the outstanding advantages over the traditional authentication methods based on Personal Identification Numbers (PINs) or passwords. In fact, since iris is intrinsically and uniquely associated with an individual, they cannot be forgotten, easily stolen or reproduced. However, the use of iris may also have some drawbacks related to possible security breaches. Since iris characteristics are limited and immutable, the system security is irreparably compromised when an attacker has access to the database where they are stored. To overcome this problem, the secure template storage techniques [Vetr09], [Sant10] were introduced in the iris recognition systems. In these systems, irreversible cryptographic transformations, such as hash functions, are used to produce secure templates before storing them. Unfortunately, slight differences in the acquired iris data, due to acquisition noise, result in a large difference in the cryptographic function’s output. In these conditions, even comparisons between templates acquired from the same user will fail. To overcome the problem raised by the acquisition noise, Error Correction Codes (ECCs) concepts can be used. Since the application of the ECCs has a great influence on the FRR and FAR values of the system, the choice of the code must be done carefully. In this Chapter, the ECCs properties which influence the performance of the system are analysed. To illustrate how these properties influence the performance of the system, Low Density Parity Check (LDPC) codes and Reed Solomon (RS) codes are used, which are two of the most commonly used ECCs in the iris systems with
secure template storage. LDPC codes are used with a hash function to provide secure iris template storage [Vetr09]. To enhance the security of these kinds of systems, a universal mask which selects only the 5,142 most reliable bit positions of the 9,600 bits in the iris templates was introduced [Sant10]. Sutcu et al. [Sutc08] and Nagar et al. [Naga10] developed secure biometric systems based on LDPC codes and fingerprints. Kanade et al. [Kana08] concatenated Hadamard and RS codes for iris template secure storage on smart cards. The thesis uses LDPC code for correcting the errors in the iris templates.

In the field of pattern recognition, Daugman [Daug94] proposed an algorithm for iris recognition. Subsequently many researchers used that algorithm as a benchmark. This Chapter uses IRS designed in Chapter 2, which presents a novel approach on iris recognition. CASIA-IrisV3 [CASI06] iris database is used for conducting experimental tests.

Recent idea of using message digest algorithm to make cancellable biometrics enable the tendency to use SHA. The SHA is a series of cryptographic hash functions published by the National Institute of Standards and Technology (NIST). The NIST published SHA as Federal Information Processing Standard Publication (FIPS PUB) 180-2 [Nati02] consisting of four algorithms, namely SHA-160, SHA-256, SHA-384 and SHA-512. For transforming the error corrected iris code into cancellable iris code, SHA-512 is used. In the proposed research work, SHA-512 hash is employed for authentication due to its security and uniqueness.

It is observed from the literature that the digital watermarking is the well-known approach for image authentication. In most conventional authentication techniques based on watermarking, the original image is distorted permanently due to the authentication itself. Typically, this distortion cannot be removed completely due to quantization, bit-replacement, or truncation at the grayscale values 0 and 255. Although the distortion is often quite small, these distortions are not allowed in some sensitive applications, such as medical or legal
imagery or images with a high strategic importance in certain military applications. Thus, it is
desired to get back the changes introduced by authentication. Data embedding techniques
satisfying this requirement are referred to as *reversible* (also known as *lossless*) image
authentication techniques. To achieve the reversibility, invertible integer-to-integer wavelet
transforms [Daub98] are used. Tian [Tian03] embeds the data using the difference expansion
technique, which is one of the best reversible data hiding methods. Yang *et al.* proposed a
reversible watermarking scheme based on integer Discrete Cosine Transform (DCT)
[Yang04]. Further, Xuan *et al.* [Xuan05] used the Least Significant Bit-plane (LSB) method
to embed the data into the high frequency of (Cohen-Daubechies-Fauraue) CDF (2, 2) integer
wavelet transform coefficients whose magnitudes are smaller than a certain predefined
threshold. As this method is one of the best reversible data hiding methods, the thesis utilizes
the lossless data hiding method based on IWT and threshold embedding technique for digital
image authentication reasons.

Embedding the iris in the form of cancellable biometrics into an image increases the
secrecy, and the use of reversible watermarking does not degrade the quality of original
image i.e., with naked eye no one can find the differences between original image and
watermarked-image, since data are embedded in the LSB of Integer Wavelet Transform
(IWT) coefficients. This Chapter proposes a new way to authenticate an image using LDPC
and SHA based iris recognition method with reversible watermarking scheme, which is based
on IWT and threshold embedding technique. The rest of the Chapter is organized as follows:
The LDPC coding scheme is presented in Section 4.2. Section 4.3 exhibits the lossless data
hiding using IWT and threshold embedding technique. The idea of enrolment process of the
proposed system is presented in Section 4.4. The verification process (authentication) of the
proposed system is discussed in Section 4.5. Some experimental results and performance
analysis are given in Section 4.6. The conclusion is drawn in Section 4.7.
4.2 LDPC coding scheme

The LDPC codes [Gall63] were first developed for a doctoral dissertation in 1963 by R. G. Gallager. Gallager’s work was largely ignored for approximately 30 years until connections were drawn between the iterative methods used for decoding both LDPC codes and turbo codes. The study of LDPC codes was resurrected in the mid 1990’s with the work of Mackay [Mack99] and Luby et al. [Luby01], who noticed the advantages of linear block codes which possess sparse (low-density) parity-check matrices. Today, LDPC codes are becoming paramount in digital decoding and reliable digital communication over a noisy channel. Applications of LDPC codes range from cellular telephones to worldwide computer communication.

4.2.1 Basic Construction

A Gallager LDPC code has a very sparse random parity-check matrix. The parity-check matrix $H$ can be constructed as follows: A transmitted block length $N$ and a source block length $K$ are selected. Define $M = N - K$ to be the number of parity checks. Select a column weight $t$, which will initially be an integer greater than or equal to 3. Create a rectangular $M \times N$ matrix ($M$ rows and $N$ columns) $H$ at random with exactly weight $t$ per column and any weight per row as uniform as possible. If $N/M$ is chosen to be an appropriate ratio of integers then the number of 1s per row can be constrained to be exactly $tN/M$; in this case, the resulting code is called as a regular Gallager LDPC code because the bipartite graph defined by the parity-check matrix is regular. Then, the Gaussian elimination and reordering of columns are performed to derive an equivalent parity-check matrix in systematic form $[P^T|I_M]$. There is a possibility that the rows of the original matrix are not independent (though for odd $t$, this has small probability); in this case, $H$ is a parity-check matrix for a code with the same $N$ and with smaller $M$, that is, a code with greater rate.
Redefining the $H$ to be the original matrix with its columns reordered as in the Gaussian elimination. The matrix $H = [C_1 C_2]$ is composed of two very sparse matrices $C_1$ and $C_2$ as follows:

The matrix $C_1$ is a rectangular $M \times K$ matrix that is very sparse.

The matrix $C_2$ is a square $M \times M$ matrix that is very sparse and invertible. The inverse $C_2^{-1}$ of this matrix in modulo 2 arithmetic has been computed during the Gaussian elimination which produces the matrix $P = C_2^{-1} C_1$, where $P$ is the modulo 2 product of the two matrices $C_2^{-1}$ and $C_1$.

4.2.2 LDPC Encoding

Encoding is performed by putting $H$ in the form $[P^T I]$ via Gauss-Jordan elimination method, from which the generator matrix can be put in the systematic form $G = [I P]$. The problem of encoding via $G$ is that the sub-matrix $P$ is generally not sparse.

4.2.3 LDPC Decoding

Suppose a codeword $c$ is transmitted by a $(n, k)$ binary LDPC code $C$. It is assumed that the vector $Y$ is received at the receiver end. In the syndrome $HY^T$, each received bit $Y_i$ affects at most $c$ components of that syndrome because the $i^{th}$ bit is in $c$ parity checks. If among all the bits involved in these $c$ parity checks, call it as $S$, only the $i^{th}$ is error then these components, $c$ of $HY^T$ will be equal to 1. This indicates that the parity check equations are not satisfied and the $i^{th}$ bit is to be flipped. If there is more than one error among $S$, then several of the $c$ components of $HY^T$ will be equal to 1. This indicates that multiple bits must be flipped. The Gallager hard decision LDPC decoding algorithm is adopted in the thesis as follows:
Step 1: Compute $HY^T$ and determine the number of unsatisfied parity checks. That is, the parity checks is equal to 1 in $HY^T$.

Step 2: For each of the $n$ bits, compute the number of unsatisfied parity checks involving that bit.

Step 3: Change the bits of $Y$ that are involved in the largest number of unsatisfied parity checks; reset the resulting vector to $Y$ again.

Step 4: Iteratively repeat step 1, 2 and 3 until either $HY^T = 0$, in which case the received vector is decoded as the latest $Y$, or until a certain number of iterations is reached or in which case the received vector is not decoded.

4.3 The lossless data hiding using integer wavelet transform and threshold embedding technique

It is decided to use the CDF (2, 2) integer wavelet transform, which is used in JPEG2000 for image lossless compression, to obtain the wavelet coefficients. Because of frequency mask, the data embedded into in the first level high frequency sub-bands will have less visible artifact to human eyes. The enrolment process of the proposed authentication system embeds parity checks, parity matrix and SHA-512 hash of iris code (generated in the phase quantization process of iris recognition system, which is discussed in Chapter 2) into the first level high frequency sub-bands of images using threshold embedding technique without any loss.

To embed parity checks, parity matrix and SHA-512 hash of iris code into a high frequency coefficient $x$, the absolute value of the coefficient is compared with predefined $T$. If $|x| < T$, the coefficient value is doubled and then to-be-embedded bit is added. The resultant coefficient is denoted by $x'$. Otherwise, if $x \geq T$, the coefficient is added by $T$; if $x \leq -T$, the coefficient is subtracted by $(T \cdot 1)$, and no bit is embedded into this coefficient. These rules are summarized in equation (3.5).
Histogram modification (discussed in Chapter 3) is performed prior to the embedding to ensure no overflow or underflow takes place. The bookkeeping data of histogram modification, parity checks, parity matrix and SHA-512 hash string of iris code are embedded into the high frequency IWT coefficients. The watermarked-image carrying hidden parity checks, parity matrix and SHA-512 hash is obtained after inverse IWT.

In the authentication process of the proposed system, simply by reversing the embedding process the original image, parity checks, parity matrix and SHA-512 hash are extracted back from watermarked-image. Then this extracted parity checks, parity matrix and SHA-512 hash are used to authenticate the original image by matching with the new hash generated from the live person. At the outset, IWT is applied on watermarked-image to find eligible sub-bands then bookkeeping data of histogram modification, parity checks, parity matrix and SHA-512 hash are extracted from these sub-bands. For a coefficient, if it is less than \( 2T \) and larger than \(-2T + 1\), the LSB of this coefficient is the bit embedded into this coefficient. Otherwise, jump to the next coefficient since the current coefficient has no hidden bit in it. Concretely, each high frequency coefficient is restored to its original value by applying the equation (3.6). After the extraction of code, inverse IWT is applied with untouched sub-band and processed sub-bands with parity checks, parity matrix and SHA-512 hash. Finally, original image is recovered by making inverse histogram modification.

4.4 Enrolment process of the proposed system

The IRS designed in Chapter 2 is used for generating \( n \) number of iris codes from \( n \) number of eye samples collected from same person on different time intervals. From the \( n \) number of iris code, a unique iris code \( x \) is constructed by using majority voting scheme. The LDPC encoding scheme operates on \( x \) and produces codewords, also called as Error Corrected Iris Code (ECIC). These ECIC consist of iris code \( x \) and parity checks \( p \). Simultaneously, SHA-512 produces hash \( h \) from code \( x \) Finally, parity checks \( p \) of ECIC,
parity matrix $H$ and hash $h$ from SHA-512 make code $s$, which is embedded into a digital image using integer wavelet transform and threshold embedding technique. The entire enrolment process is depicted in Figure 4.1.

**Figure 4.1** Block diagram of enrolment process of the proposed system

The novelty of IRS includes improving the speed and accuracy of the iris segmentation process, extracting the iris image in a way to reduce the recognition error, producing a feature vector with discriminating texture features and a proper dimensionality in a path to improve the recognition accuracy and computational efficiency. The Canny edge detection and circular Hough transforms are used for the segmentation process. The segmented iris is normalized using Daugman’s rubber sheet model in the ranges $[-32^\circ, 32^\circ]$ and $[148^\circ, 212^\circ]$. The phase data from 1D Log-Gabor filter is extracted and encoded efficiently to produce a proper feature vector using phase quantization method. The results of
process of segmentation, normalization and phase quantization for a sample eye image from CASIA-IrisV3-Interval database is given in Figure 4.2. Once the iris region is successfully segmented from an eye image, the next stage is to transform the iris region to a fixed dimension to allow comparisons. The dimensional inconsistencies between eye images are mainly due to the stretching of the iris caused by pupil dilation from varying levels of illumination. Other sources of inconsistency include, varying imaging distance, rotation of the camera, head tilt, and rotation of the eye within the eye socket.

![Eye images with various steps of processing](image)

(a) Original eye image from CASIA-IrisV3-interval database  
(b) After applying Canny edge detector  
(c) After applying Hough transform  
(d) Isolated iris region

(e) Normalized iris for $[0^\circ, 360^\circ]$  
(f) Normalized iris in the ranges $[-32^\circ, 32^\circ]$ and $[148^\circ, 212^\circ]$

-3.14 -i0.23 -1.44 -i2.82 1.75 -i2.69 3.23 +i0.17 1.44 +i2.92 -1.84 +i2.64 .... 00 00 10 11 11 01......
-3.68 -i0.27 -1.67 -i3.31 2.06 -i3.14 3.76 +i0.21 1.68 +i3.42 -2.15 +i3.10..... 00 00 10 11 11 01......
-3.33 -i0.23 -1.58 -i3.00 1.86 -i2.92 3.48 +i0.17 1.55 +i3.15 -1.98 +i2.82..... 00 00 10 11 11 01......
-3.32 -i0.18 -1.61 -i2.97 1.82 -i2.92 3.46 +i0.14 1.57 +i3.11 -1.93 +i2.82..... 00 00 10 11 11 01......
-3.28 -i0.09 -1.66 -i2.89 1.73 -i2.92 3.14 +i0.07 1.61 +i3.02 -1.82 +i2.82..... 00 00 10 11 11 01......
-3.37 -i0.32 -1.53 -i3.07 1.95 -i2.92 3.53 +i0.25 1.51 +i3.25 -2.09 +i2.82..... 00 00 10 11 11 01......

(g) After convolution with 1D Log-Gabor wavelet filter using FFT and inverse FFT  
(h) Iris template

**Figure 4.2** Segmentation, normalization and phase quantization processes of iris recognition system

Construction of unique iris code $x$ from $n$ number of iris code is done in a simple method called *majority voting*. Finding the majority in each bit formulates the unique iris code. Fabrication of such unique iris code $x$ from three sample iris codes is explained in
Figure 4.3. From the unique code $x$, ECIC is formed by LDPC and hash version $h$ is transformed by SHA-512. Rest of this Section illustrates LDPC encoding and then deliberates the SHA-512.

![Three sample iris codes](image1)

**Figure 4.3 Construction of unique iris code**

In general, LDPC codes are defined by a sparse parity-check matrix. This sparse matrix is often randomly generated, subject to the sparsity constraints. Figure 4.4 is a graph fragment of an example of LDPC code using Forney's factor graph notation. In this graph, $n$ variable nodes in the top of the graph are connected to $(n-k)$ constraint nodes in the bottom of the graph. This is a popular way of graphical representation of an $(n, k)$ LDPC code. The bits of a valid message, when placed on top of the graph, satisfy the graphical constraints. Specifically, all lines connecting to a variable node (box with an ‘$=$’ sign) have the same value, and all values connecting to a factor node (box with a ‘$+$’ sign) must sum, modulo two to zero (in other words, they must sum to an even number). After construction of unique iris code $x$, each column in the iris code $x$ is considered as message in LDPC encoding scheme, and $x$ is encoded to make ECIC with the help of generator matrix $G$.

![Graph fragment of an example LDPC encoding](image2)

**Figure 4.4 Graph fragment of an example LDPC encoding**
After forming ECIC by multiplying all columns with \( G \), the parity checks \( p \) from each codeword of ECIC is segregated. The following example illustrates the method of LDPC encoding. Ignoring any lines going out of the picture, there are 8 possible 6-bit strings corresponding to valid codewords (i.e., 000000, 011001, 110010, 101011, 111100, 100101, 001110, 010111). This LDPC code fragment represents a 3-bit message encoded as six bits. Redundancy is used here, to increase the chance of recovering from channel errors. This is a \((6, 3)\) linear code, with \( n = 6 \) and \( k = 3 \). By ignoring lines going out of the picture, the parity-check matrix representing this graph fragment is given in the following equation:

\[
H = \begin{pmatrix}
1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 1 & 0
\end{pmatrix}
\]  

(4.1)

In this matrix, each row represents one of the three parity-check constraints, while each column represents one of the six bits in the received codeword. In this example, the eight codewords are obtained by putting the parity-check matrix \( H \) into this form \([-P^T \mid I_{n-k}\] through basic row operations as shown in the following equation:

\[
H = \begin{pmatrix}
1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 1 & 0
\end{pmatrix} \sim \ \begin{pmatrix}
1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 1
\end{pmatrix} \sim \ \begin{pmatrix}
1 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 1
\end{pmatrix}
\]  

(4.2)

From this, the generator matrix \( G \) is obtained as \([I_k \mid P]\). The obtained generator matrix from equation (4.2) is shown in equation (4.3). It is to be noted that in the special case of this being a binary code \( P = -P \).

\[
G = \begin{pmatrix}
1 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 & 0
\end{pmatrix}
\]  

(4.3)
Finally, by multiplying all eight possible 3-bit strings by $G$, all eight valid codewords are obtained. For example, equation (4.4) shows the codeword obtained for the bit-string '101'.

\[
(1 \ 0 \ 1) \cdot \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} = (1 \ 0 \ 1 \ 0 \ 1 \ 1) \quad (4.4)
\]

Simultaneously, the unique iris code $x$ is transformed to hash $h$, also called as cancellable iris code, by SHA-512 message digest algorithm. The SHA-512 found in FIPS PUB 180-2 documentation is adopted for this system. Algorithm for SHA-512 message digest is as follows:

**Step 1: Setting the length of one word and the number of output words**

Define the length of one word as $n=64$

Define the number of output words $m=8$

**Step 2: Padding the message M**

The message, $M$, is padded before hash computation begins. The purpose of this padding is to ensure that the padded message is a multiple of 1,024 bits. Suppose the length of the message $M$ is $l$ bits. Append the bit $j$ to the end of the message, followed by $k$ zero bits, where $k$ is the smallest non-negative solution to the equation $l + 1 + k \equiv 896 \mod 1024$. Then append the 128-bit block that is equal to the number $l$ expressed using a binary representation. The length of the padded message is now being a multiple of 1,024 bits.

**Step 3: Parsing the padded message into message blocks**

SHA-512 parses the padded message into $N$ 1,024 bits blocks denoted by $M^{(1)}, M^{(2)}, \ldots, M^{(N)}$. For each 1,024-bit $M^{(i)}$, the $M$ is divided into sixteen 64-bit sub-blocks denoted by $M^{(i)}_0, M^{(i)}_1, \ldots, M^{(i)}_{15}$. 
Step 4: Setting the initial hash values

The initial hash value, \( H^{(0)} \), consist of the following eight 64-bit words, in

\[
H_{0}^{(0)} = 6a09e667f3bcc908 \\
H_{1}^{(0)} = bb67ae8584caa73b \\
H_{2}^{(0)} = 3c6ef372fe94f82b \\
H_{3}^{(0)} = a54ff53a5f1d36f1 \\
H_{4}^{(0)} = 510e527fade682d1 \\
H_{5}^{(0)} = 9b05688c2b3e6c1f \\
H_{6}^{(0)} = 1f83d9abfb41bd6b \\
H_{7}^{(0)} = 5be0cd19137e2179
\]

Step 5: Defining the logical functions

SHA-512 uses six logical functions, where each function operates on 64-bit words, which are represented as \( x, y \), and \( z \) The result of each function is a new 64-bit word.

\[
Ch(x, y, z) = (x \land y) \oplus (x \land z) \tag{4.5}
\]

\[
Maj(x, y, z) = (x \land y) \oplus (x \land z) \oplus (y \land z) \tag{4.6}
\]

\[
\Sigma_{0}^{(512)}(x) = ROTR^{28}(x) \oplus ROTR^{34}(x) \oplus ROTR^{39}(x) \tag{4.7}
\]

\[
\Sigma_{1}^{(512)}(x) = ROTR^{14}(x) \oplus ROTR^{18}(x) \oplus ROTR^{41}(x) \tag{4.8}
\]

\[
\sigma_{0}^{(512)} = ROTR^{1}(x) \oplus ROTR^{8}(x) \oplus SHR^{7}(x) \tag{4.9}
\]

\[
\sigma_{1}^{(512)} = ROTR^{19}(x) \oplus ROTR^{61}(x) \oplus SHR^{6}(x) \tag{4.10}
\]

where \( \land \) is bitwise AND operation, \( \oplus \) is bitwise XOR (“exclusive-OR”) operation, \( ROTR \) is rotate right (circular right shift) operation, and \( SHR \) is right shift operation.
Step 6: Defining the constants

SHA-512 uses the same sequence of eighty constant 64-bit words, $K_0^{(512)}, K_1^{(512)}, ..., K_{29}^{(512)}$. These words represent the first sixty-four bits of the fractional parts of the cube roots of the first eighty prime numbers. In hex, these constant words are shown from left to right.

428a2f98d728ae22 7137449123ef65cd b5c0fbcf4ec4d2b7f e9b5db9819dbbc
3956c25bf348b538 50911f16b56d019 923f82a4af194f9b ab1c5ed5a6d8818
d807aa98a3303242 12835b0145706fbee 243185be4ee4b28c 550c7dc3d5f8b4e2
72be5d74f27b896f8 80deb1fe3b169619 9bdc06a725c71235 c19bf174cfc69269
e49b69c19ef14ad2 efbe4786384f25e3 0fc19dc68b8cd5b5 240ca1cc77ac9c65
2de92c6f592b0275 4a748aa6ea6e483 5cb09dcd41f9b47 76f988da81153b5
983e5152ee66d027f a831c662db43210 b00327c898fb213f bf597fc7beef0ee4
c6e00b33da88fc2 d5a79147930aa725 06ca6351e003826f 14292670a0e6e70
27b70a854d22ffc 2e1b21385c26c926 4d26dcf5ac42aed 53380d139d95b3df
650a73548baf63de 766a0abb3c77b2a8 81c2c92e7edaee6 92722c85148235b3
a2bfe8a4cf10364 a81664b0432001 c248b50d0f89791 c76c51a30654be30
d19e819d6ef5218 d69062455565a910 f40e3585571202a 106aa07032bb1b8
19a4c11682d2d0c8 1e376c085141ab53 2748774cdf8eeb99 34b0bcb5e1948a8
391c0cb3c5c9a5c6 4ed8a4a4e3418acb 5b9cca4f7763e373 682e6ff3db8a3
748f82ee5defb2fc 78a5636f431726f6 84c8791a1f0ab72 80c072081a6433ec
90befffa23631e8a 4506cebe82bde9 bef9a3f7b2c67915 c67178f2e372532b
c4f74c16b8953a9 bda981c647c154b8 171f08e589b67024 b41c8f2f87d7e178
06f067aa72176fba 0a637dc5a2c898a6 113f9804bef90daa 1b710b3513c471b
28db7f523047d84 32caab7b40c72493 3c9ebe0a15c9bebc 431d67c49c10d4c
4cc54d4beeb3a42b6 597f299cfc65f7e2a 5fcb6fab3ad6faec 644198c4a75817

Step 7: SHA-512 hash computation

Addition (+) is performed in modulo $2^{64}$. Each message block, $M^{(1)}, M^{(2)}, ..., M^{(N)}$, is processed in order, using the following steps:
For $i = 1$ to $N$:

\[
\{ \\
\}
\]

1. Prepare the message schedule, $\{W_t\}$:

\[
W_t = \begin{cases} 
M_t^{(i)} & 0 \leq t \leq 15 \\
\sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16} & 16 \leq t \leq 79 
\end{cases}
\]

(4.11)

2. Initialize the eight working variables, $a, b, c, d, e, f, g$ and $h$, with the $(i-1)$th hash value:

\[
a = H_0^{(i-1)} \\
b = H_1^{(i-1)} \\
c = H_2^{(i-1)} \\
d = H_3^{(i-1)} \\
e = H_4^{(i-1)} \\
f = H_5^{(i-1)} \\
g = H_6^{(i-1)} \\
h = H_7^{(i-1)}
\]

3. For $t=0$ to $79$:

\[
\{ \\
T_1 = h + \sum_1^{512}(e) + Ch(e, f, g) + K_t^{512} + W_t \\
T_2 = \sum_0^{512}(a) + Maj(a, b, c) \\
h = g \\
g = f
\]
\[ f = e \]
\[ e = d + T_1 \]
\[ d = c \]
\[ c = b \]
\[ b = a \]
\[ a = T_1 + T_2 \]

4. Compute the \( i^{th} \) intermediate hash value \( H^{(i)} \):

\[
\begin{align*}
H^{(i)}_0 &= a + H^{(i-1)}_0 \\
H^{(i)}_1 &= b + H^{(i-1)}_1 \\
H^{(i)}_2 &= c + H^{(i-1)}_2 \\
H^{(i)}_3 &= d + H^{(i-1)}_3 \\
H^{(i)}_4 &= e + H^{(i-1)}_4 \\
H^{(i)}_5 &= f + H^{(i-1)}_5 \\
H^{(i)}_6 &= g + H^{(i-1)}_6 \\
H^{(i)}_7 &= h + H^{(i-1)}_7
\end{align*}
\]

After repeating steps 1 through 4 for \( N \) times (i.e., after processing \( M^{(N)} \)), the resulting 512-bit message digest of the message, \( M \), is

\[
H^{(N)}_0 \| H^{(N)}_1 \| H^{(N)}_2 \| H^{(N)}_3 \| H^{(N)}_4 \| H^{(N)}_5 \| H^{(N)}_6 \| H^{(N)}_7
\]

(4.12)

By applying the above algorithm hash \( h \) is found for any unique iris code. A sample hash \( h \) from SHA-512 for the unique iris code \( \chi \) is given in Figure 4.5.
SHA-512 hash length: 64


SHA-512 hash string length: 128

SHA-512 hash string:
dc5c228f6938c6ffe0c056be1357abbd241d3b6ca5ea6652356774ffe9829d098f219da9371aab5726ee4476dd8f546f34da2dd75aa1d3eb089dc25660f4a60

Figure 4.5 Sample hash $h$ of SHA-512

4.5 Verification process of the proposed system

Iris code $\hat{x}$ is generated by IRS of eye sample collected from live person. The LDPC decoding scheme operates on $\hat{x}$ and produces $\hat{x}$ with the help of parity matrix $H$ and parity checks $p$, which are extracted from watermarked-image using IWT and threshold embedding technique. Like in enrolment process, SHA-512 produces hash $\bar{h}$ from code $\hat{x}$. Finally, hash $\bar{h}$ from SHA-512 and hash $h$ extracted from watermarked-image is compared for authentication. This verification process is illustrated in Figure 4.6.

To illustrate LDPC decoding process, it is assumed that the first three bits from live person and the next three bits are appended from parity checks. The valid codeword 101011 is considered from the example discussed in Section 4.4. If the first bit of iris code from live person is changed then codeword becomes 001011. Since the iris code must have satisfied the code constraints, the iris code is represented by writing them on the top of the factor graph. The result is validated by multiplying the corrected codeword $r$ (transpose of codeword from live person’s iris) with the parity-check matrix $H$, which is given in equation (4.1).
**Figure 4.6** Block diagram of verification process of the proposed system

\[ z = Hr = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \quad (4.13) \]

As the outcome \( z \) (the syndrome) of this operation is the \( 3 \times 1 \) non-zero vector in equation (4.13), look at column 1 of \( H \) which is the only equivalent to the outcome \( z \). So the first bit is flipped as 1 and the validation is continued.

\[ z = Hr = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad (4.14) \]

Now in the equation (4.14), outcome \( z \) (the syndrome) of this operation is the \( 3 \times 1 \) zero vector, the resulting codeword \( r \) is successfully validated. Thus, the iris code is decoded iteratively.
4.6 Experimental results

From the public database CASIA-IrisV3 [CASIO6], 200 classes (eyes) and 1,500 images in the subset labeled as CASIA-IrisV3-Interval are chosen for the experiment. For each iris class, four samples are chosen for enrolment process. In verification process, rest of the iris image in the database is compared with the other entire iris. The total number of comparisons is \((1500 \times 1499)/2 = 1,124,250\), where the total number of intra-class comparisons is 7,648 and that of inter-class comparisons is 1,116,602. Figure 4.7 shows distributions of intra-class (solid line) and inter-class (dashed line) matching distances for CASIA-IrisV3-interval data sets. From Figure 4.7, it can be found that the distance between the intra-class and the inter-class distribution is large, and the portion that overlaps between the intra-class and the inter-class is very small. So that almost, 100% correct recognition rates are obtained on CASIA-IrisV3-interval data sets.

![Figure 4.7](image)

**Figure 4.7** Distributions of intra-class and inter-class distances for CASIA-IrisV3-Interval database
To show the error correction capability of LDPC, Reed Solomon code is considered from the family of ECCs. Figure 4.8 shows selected curves of the RS and LDPC codes are overlaid on top of the genuine and impostor normalized HD distributions. As can be easily observed, the RS correction curves are significantly less steep than the LDPC curves. Moreover, the RS code is also less granular than the LDPC. This leads to performance degradation, with False Rejection Rate (FRR) and False Acceptance Rate (FAR) values varying from 0.08% to 21.293% and from 0.014% to 57.36%, respectively. The corresponding Equal Error Rate (EER) value is 2.44%. But for LDPC, the resulting FRR and FAR values range from 0.754% to 1.87% and from 0.036% to 0.365%, respectively. For this situation, the estimated EER is 0.41%.

Figure 4.8 RS and LDPC codes overlaid on top of the genuine and impostor normalized HD distributions
The proposed scheme is experimented with Lena image, Boat, Baboon, Pepper, House, Nithusha (new image) images etc., and various eye images in the CASIA-IrisV3 database for authentication purpose. The quality of watermarking is generally measured with PSNR. If it is more than 30 dB, it is good but cannot be identified by naked eye. For most of the experiments using IWT and threshold embedding technique, it is more than 45 dB. The sample results for the experiment conducted on Lena and Nithusha are shown in Figure 4.9. The PSNR value for the Lena image and its watermarked-image is 52.42 dB for $T = 6$, which is 2 dB more than difference expansion method [Tian03] and 4 dB more than DCT method [Yang04]. Similarly, The PSNR value of the Nithusha image and its watermarked-image is 54.62 dB for $T = 6$, which is also approximately 2 dB more than difference expansion method and 4 dB more than DCT method.

![Image](image1.png)

**Figure 4.9** Results of lossless data hiding and extraction
4.7 Conclusion

Compared with Daugman’s method, significant decrements of the error rates are observed. Using majority voting scheme and LDPC, the fuzziness i.e., the variability and noise in the type 2 regions (reflections of external light sources) in the iris code is resolved. The LDPC codes have shown to lead to better recognition performance results than RS codes, due to the better steepness and granularity properties. Low FRR and FAR is achieved by using LDPC codes in this system. The MD5 algorithm could produce identical hashes for two different messages if the initialization vector could be chosen, so MD5 cannot be adopted for authentication. The security complexity of SHA-512 is $2^{256}$ under Birthday attack. As SHA-512 is used, security of this scheme is very high. For authenticating the image, cancellable iris code is embedded in the form of SHA-512 hash using IWT and threshold embedding technique, which gives the better performance than difference expansion method and DCT method. Compared with the method discussed in Chapter 3, this scheme is more accurate in terms of recognition rate and more secure due to the iris code embedded in the form of cancellable biometrics. It can be concluded that, the proposed system has superior performance in terms of security, accuracy and consistency compared with other existing systems.