CHAPTER 4

SYSTEM DEVELOPMENT AND IMPLEMENTATIONS

This chapter explores platform, software specifications, hardware specifications and online databases used for testing of proposed grey scale, color and medical image watermarking system. The investigation of proposed system includes various image watermarking techniques. The watermark embedding, watermark extraction algorithm along with relevant necessary block diagrams and other implementation details are presented under individual watermarking technique.

4.1 Development Environment and Databases

The proposed system is implemented using Java Net beans Integrated Development Environment (IDE) version 7.3[77][78] and Matlab[34] with MEO(version 8.0.0.7837, R2012b). The experimentation is carried out on personal computer with intel(R), core(TM) i3-2310M, processors rated at 2.10 GHz, main memory of 4 GB and 32 bit Microsoft windows 7 operating system. The cover grey scale, color and medical images of size 51x512 are used for testing. For each type sample 500 images are selected from
online databases including CVG-UGR image database [79], OsiriX database [80] and medical image samples [81]. Other 350 color images are also picked from existing open image databases available on internet and own database.

The grey scale watermark images of size 64x64, 128x128 and 256x256 and 512x512 are used as watermark. The implementation details of all proposed watermarking techniques are explored under Tech_Num_1 to Tech_Num_8.

4.2 Proposed Image Watermarking System

4.2.1 Implementation of Tech_Num_1: Modified LSB Insertion Based Technique.

FIGURE 4.1 shows Java pixel representation. Each 32 bit pixel consists of alpha, red(r), green(g) and blue(b) components.

<table>
<thead>
<tr>
<th>alpha value</th>
<th>red plane</th>
<th>green plane</th>
<th>blue plane</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>11111111</td>
<td>00000000</td>
<td>00000000</td>
</tr>
<tr>
<td></td>
<td>(a)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000000</td>
<td>red plane</td>
<td>00000000</td>
<td>00000000</td>
</tr>
<tr>
<td></td>
<td>(b)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>red plane</td>
</tr>
<tr>
<td></td>
<td>(c)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(d)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FIGURE 4.1: Details of 32 bits Java pixel representation a) 8 bits allocation to each of alpha value, red, green and blue planes b) The 0xff0000 required for separation of red, green and blue planes c) separation of red plane after AND(&) operation of (a) and (b) values d) result after performing right shift(>>) operation
Algorithm 4.1: Modified LSB watermark embedding

**Input:** Cover Image, Binary Input String

**Output:** Watermarked Image

**Step 1:** Read Cover Image = c:girl.jpg.
**Step 2:** Grab pixels of Cover Image into grabber.
**Step 3:** Formulate Binary Input String using Step-4 and Step-5.
**Step 4:** StringBuilder binary = new StringBuilder();
**Step 5:** StringBuilder b = binary.append(Integer.toBinaryString(x));
**Step 6:** Display Binary Input String which is to be embedded.
**Step 7:** int pixels[] = new int[width*height]
   where w and h are width and height of Cover Image
**Step 8:** Cover Pixel = (int[]) grabber.getPixels();
**Step 9:** for i=0 to Cover Pixel.length
   **Step 10:** int c = Cover Pixel[i];
   **Step 11:** int r = (c&0xff0000) >> 16;
   **Step 12:** int g = (c&0x00ff00) >> 8;
   **Step 13:** int b = (c&0x0000ff);
   **Step 14:** if (i < Watermark String Length)
      **Step 15:** if (Binary Input String .charAt(i) == '0')
         **Step 16:** b = b & 254;
      **Step 17:** else
         **Step 18:** b = b | 1;
      **Step 19:** pixels[i] = ((255 << 24) | (r & 0xff) << 16) | (g & 0xff) << 8) | (b & 0xff));
   **Step 20:** end if
   **Step 21:** else
      **Step 22:** pixels[i] = ((255 << 24) | (r & 0xff) << 16) | (g & 0xff) << 8) | (b & 0xff));
   **Step 23:** end if
**Step 24:** end for
**Step 25:** Create Watermarked Image using pixels.
Chapter 4. System Development and Implementations

The modified LSB watermark embedding is shown in **ALGORITHM 4.1**.

Algorithm embeds input binary string into color cover image of wxh size and gives watermarked image as output. The color components r, g and b are separated by performing right shift operations.

### 4.2.1.2 Watermark Extraction for Modified LSB Based Technique

**Algorithm 4.2 : Modified LSB watermark extraction**

<table>
<thead>
<tr>
<th>Input:</th>
<th>Watermarked Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>Output_Extracted_String</td>
</tr>
<tr>
<td>Step-1:</td>
<td>Read Watermarked Image formed in embedding</td>
</tr>
<tr>
<td>Step-2:</td>
<td>Grab pixels of Watermarked Image in grabber1</td>
</tr>
<tr>
<td>Step-3:</td>
<td>int pixels1[] = new int[width*height];</td>
</tr>
<tr>
<td></td>
<td>where, w is width of Watermarked Image</td>
</tr>
<tr>
<td></td>
<td>h is height of Watermarked Image</td>
</tr>
<tr>
<td>Step-4:</td>
<td>for i=0 to Watermark_String_Length-1</td>
</tr>
<tr>
<td>Step-5:</td>
<td>int c = watermarkedArray[i];</td>
</tr>
<tr>
<td>Step-6:</td>
<td>int r = (c&amp;0xff0000)&gt;&gt;16;</td>
</tr>
<tr>
<td>Step-7:</td>
<td>int g = (c&amp;0x00ff00)&gt;&gt;8;</td>
</tr>
<tr>
<td>Step-8:</td>
<td>int b = (c&amp;0x0000ff);</td>
</tr>
<tr>
<td>Step-9:</td>
<td>String binString = Integer.toBinaryString(b);</td>
</tr>
<tr>
<td>Step-10:</td>
<td>Char s = binString.charAt ((binString.length()-1));</td>
</tr>
<tr>
<td>Step-11:</td>
<td>Output_Extracted_String .append(s);</td>
</tr>
<tr>
<td>Step-12:</td>
<td>end for</td>
</tr>
<tr>
<td>Step-13:</td>
<td>Display Output_Extracted_String</td>
</tr>
</tbody>
</table>

The modified LSB watermark extraction is shown in **ALGORITHM 4.2**.

The algorithm accepts watermarked image as input and displays extracted string as output.
4.2.2 Implementation of Tech_Num_2: DWT Based Image Watermarking

4.2.2.1 Watermark Embedding for DWT Based Image Watermarking

Algorithm 4.3: DWT based watermark embedding

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>Cover Image, Watermark</td>
</tr>
<tr>
<td>Output</td>
<td>Watermarked Image.</td>
</tr>
<tr>
<td>Step-1</td>
<td>Decompose the Cover Image using simple Haar Wavelet.</td>
</tr>
<tr>
<td></td>
<td>[LL1,HL1,LH1,HH1] = dwt2(Cover Image,’Haar’).</td>
</tr>
<tr>
<td>Step-2</td>
<td>Perform second level DWT on LL1 to give LL2, HL2, LH2 , HH2.</td>
</tr>
<tr>
<td></td>
<td>[LL2,HL2,LH2,HH2] = dwt2(LL1,’Haar’).</td>
</tr>
<tr>
<td>Step-3</td>
<td>Repeat decomposition for LL2 to give next level: LL3, HL3, LH3, HH3.</td>
</tr>
<tr>
<td>Step-4</td>
<td>Find Arnold periodicity P of watermark using equation 2.16.</td>
</tr>
<tr>
<td>Step-5</td>
<td>Generate Pn_Sequence depending on KEY and find SUM.</td>
</tr>
<tr>
<td></td>
<td>Where 0 (\leq) KEY (\leq) P. SUM is summation of Pn_Sequence.</td>
</tr>
<tr>
<td>Step-6</td>
<td>If SUM &gt; predefined Threshold T,</td>
</tr>
<tr>
<td></td>
<td>then find Scrambled_W1, Scrambled_W2 by applying Arnold Transform</td>
</tr>
<tr>
<td></td>
<td>to Watermark with KEY1 and KEY2 respectively</td>
</tr>
<tr>
<td></td>
<td>Where KEY1=KEY+Count,KEY2=KEY-Count ,KEY+Count (\leq) P.</td>
</tr>
<tr>
<td>Step-7</td>
<td>Watermark1 = Absolute difference(Scrambled_W1, Scrambled_W2)</td>
</tr>
<tr>
<td>Step-8</td>
<td>If SUM (\leq) T, then</td>
</tr>
<tr>
<td></td>
<td>Apply Arnold Transform to Watermark to get Watermark1.</td>
</tr>
<tr>
<td></td>
<td>endif</td>
</tr>
<tr>
<td>Step-9</td>
<td>Add Watermark1 to HL3 coefficients of Cover Image,</td>
</tr>
<tr>
<td></td>
<td>New_HL3(i,j) = HL3(i,j)+ K1*Watermark1(i,j)</td>
</tr>
<tr>
<td></td>
<td>Where, K1 is scale factor,</td>
</tr>
<tr>
<td></td>
<td>New_HL3 (i, j) is newly calculated coefficients of level3,</td>
</tr>
<tr>
<td></td>
<td>Watermark1 (i, j) is Final_Scrambled Image.</td>
</tr>
<tr>
<td>Step-10</td>
<td>Take IDWT at Level3, Level2 and Level1 to get Watermarked Image</td>
</tr>
</tbody>
</table>
Chapter 4. System Development and Implementations

**Algorithm 4.3** shows DWT based image watermark embedding. The cover image decomposition is done at three levels using Haar wavelet to obtain HL3 component. Algorithm is non-blind and takes cover image, watermark as input and generates watermarked image as output. PN sequence, thresholding and scrambling are used for security.

### 4.2.2.2 Watermark Extraction for DWT Based Image Watermarking

**Algorithm 4.4** shows DWT based image watermark extraction process.

**Algorithm 4.4 : DWT based watermark extraction**

**Input:** Watermarked Image, Cover Image  
**Output:** Extracted Watermark.

**Step-1:** Decompose the Cover Image using simple Haar Wavelet.  
\[ [LL1,HL1,LH1,HH1] = dwt2(Cover Image,'Haar'). \]

**Step-2:** Perform second level DWT on LL1 to give LL2, HL2, LH2 and HH2.

**Step-3:** Decompose HL2 sub-band using Haar wavelet to get HL3.

**Step-4:** Decompose Watermarked Image, using Haar wavelet to get $HL3'$. 

**Step-5:** Apply Extraction formula as follows,

\[
Value(i,j) = \frac{abs(HL3(i,j), HL3'(i,j))}{K1}
\]

**Step-6:** If $Value(i,j) < Threshold\_Value$  
then Scrambled\_Watermark(i,j) = 0  
else  
Scrambled\_Watermark(i,j) = 1  
endif

**Step-7:** Apply Arnold Transform to Scrambled\_Watermark with same KEY used in embedding process to get Extracted\_Watermark.

**Step-8:** Display Extracted\_Watermark.

**Algorithm 4.3** and **Algorithm 4.4** are further extended for RGB color image watermarking by embedding watermark into blue component of RGB image.

DYPIET, Pune 82
4.2.3 Implementation of Tech_Num_3: Combine DWT-DCT Based Watermarking Technique

4.2.3.1 Watermark embedding in combine DWT-DCT Domain

The DWT-DCT based watermark embedding with YIQ color space is shown in **Figure 4.2**. It accepts color cover image and grey scale watermark as input and generates watermarked image as output. The RGB color space is converted into YIQ color space and Q component is taken into DWT-DCT domain. Detail watermark embedding process in DWT-DCT domain is given in **Algorithm 4.5**.
Algorithm 4.5: Watermark embedding in combine DWT-DCT domain

**Input:** Color Cover Image, Watermark.

**Output:** Watermarked Image.

**Step-1:** Read Color Cover Image of 512x512 sizes.

Convert into YIQ color space using equations 2.45.

**Step-2:** Apply one level DWT to Q component. Consider LH1 sub-band.

**Step-3:** Read grey scale Watermark of 64x64 size.

**Step-4:** Depending upon Key K1,

- Generate Pn.Sequence for given Watermark,
- Compute SUM, summation of all elements in Pn.Sequence.

**Step-5:** Determine Arnold Periodicity P for watermark using equation 2.16.

**Step-6:** If SUM > predefined threshold T

- then perform watermark scrambling by Key K2=P+Count,
- else perform watermark scrambling by,

Where, Count is programmer defined counter.

endif

**Step-7:** Generate two Pn.Sequence.0 and Pn.Sequence.1, depending upon sum of all elements of mid band used for 4x4 DCT transformation.

**Step-8:** Perform watermark embedding,

If Watermark bit is 0, then

\[ D' = D + K \times Pn\text{-}Sequence\_0 \]

else

\[ D' = D + K \times Pn\text{-}Sequence\_1 \]

endif

Here, D is matrix of mid band coefficients of DCT transformed block,

**Step-9:** Apply Inverse DCT to get New_LH1 component.

**Step-10:** Apply inverse DWT with LL1, HL1, New_LH1, HH1 to get New_Q.

**Step-11:** Convert YIQ to RGB using Y, I and New_Q using equations 2.46.
4.2.3.2 Watermark Extraction in Combine DWT-DCT domain

Algorithm 4.6: Watermark extraction in DWT-DCT domain

**Input:** Watermarked Image

**Output:** Extracted Watermark

**Step-1:** Read Color Watermarked Image.
Separate R, G and B components.
Convert to YIQ color space using equation 2.45.

**Step-2:** Now select Q component
Apply one level DWT to retrieve LH1 sub-band.

**Step-3:** Use 4x4 size for DCT blocks.
Generate Pn.Sequence_0 and Pn.Sequence_1, depending upon SUM of all elements of mid band used for 4x4 DCT transformation.
Use same seed that was used in watermark embedding process.

**Step-4:** if rand (state, 15) is used in embedding process,
then same process is to repeated here.
endif

**Step-5:** Extract mid band elements from DCT block,
Find correlation(extracted mid band coefficients, Pn.Sequence_0) and correlation(extracted mid band coefficients, Pn.Sequence_1)

**Step-6:** Determine watermark bits as follows,
If correlation between extracted mid band coefficients and Pn.Sequence_0 is greater than extracted mid band coefficients and Pn.Sequence_1,
then record Scrambled Watermark bit as 0
else
record Scrambled Watermark bit as 1.
endif

**Step-7:** Apply Arnold transform to Scrambled watermark to get Extracted Watermark.

The detail DWT-DCT based watermark extraction is shown in **ALGORITHM 4.6**.
Here, watermark extraction process is exactly reverse as that of watermark embedding process. The DWT-DCT based watermark extraction with YIQ color space is shown in FIGURE 4.3.

![Diagram of DWT-DCT based watermark extraction in YIQ color space](image)

FIGURE 4.3: DWT-DCT based watermark extraction in YIQ color space

The ALGORITHM 4.5 and ALGORITHM 4.6 were further modified for single DCT domain instead of DWT-DCT domains for comparative performance.

### 4.2.4 Implementation of Tech_Num_4: ROI Based Watermarking Technique for Medical Images

The proposed technology is especially applicable for health care applications. The application requires transmission of image data between remote hospital and physician at speciality hospital.

FIGURE 4.4 shows information flow at transmitter end while FIGURE 4.5 shows information flow at receiving end.
Chapter 4. System Development and Implementations

In emergency cases, the patient required to be treated at remote locations by consulting physicians at speciality hospitals.

![Diagram of medical image transmission](image1)

**Figure 4.4:** Generalized block diagram for medical image at transmitter

Transmission of patients reports in the form of image data demands high security.

![Diagram of medical image decompression](image2)

**Figure 4.5:** Generalized block diagram for medical image at receiver.

Hence, strong authentication and verification techniques must be applied.
4.2.4.1 Watermark Embedding in ROI Based Technique

Algorithm 4.7: ROI based watermark embedding

**Input:** Cover_Medical_Imag, Watermark  
**Output:** Compressed_Watermarked_Imag  

**Step-1:** Read Cover_Medical_Imag of size 512x512.  
Separate ROI and Select RONI part of medical Cover_Imag.  
Now, apply one level DWT. Consider LH1 sub-band.  

**Step-2:** Read grey scale hospital logo of size 64x64 as Watermark.  

**Step-3:** Depending upon Key K, generate Pn_Sequence for given watermark and Calculate summation say SUM, which is summation of all elements in generated Pn_Sequence.  

**Step-4:** Determine Arnold Periodicity P for given watermark  

**Step-5:** If SUM > T ,  
then Scrambled_Watermark with Key K2= P+Count,  
else  
Find Scrambled_Watermark with Key K3 = P+ Count,  
endif  

Here, T: Predefined threshold value,Count: Predefined counter.  

**Step-6:** Generate Pn_Sequence_0 and Pn_sequence_1, depending upon SUM, summation of all elements of mid band used for DCT transformation.  

**Step-7:** Perform watermark embedding using following equations,  
If Watermark bit is 0,  
then $D' = D + K \times Pn\_Sequence\_0$  
else  
then $D' = D + K \times Pn\_Sequence\_1$  
endif  

Where, $D'$ is Watermarked DCT block.  
D is matrix of mid band coefficients of DCT transformed block ,  

**Step-8:** Apply IDWT and then apply IDCT to get Watermarked_Imag.  

**Step-9:** Apply Wavelet Compression to get Compressed_Watermarked_Imag’.
In medical image watermarking the ROI and RONI of medical image are separated. **Algorithm 4.7** shows details of embedding process. The resultant watermarked image is compressed before transmission. The overall ROI based embedding process is shown in **Figure 4.6**.

**Figure 4.6:** ROI based watermark embedding process in medical image.

As ROI contains important information required for diagnosis of patient disease and refereed by physician, only RONI is used for watermark embedding.

### 4.2.4.2 ROI Based Watermark Extraction

**Figure 4.7:** ROI based watermark extraction process in medical image.

Here, the combine DWT-DCT domain is used along with watermark scrambling. The decompression is applied to compressed watermarked image at receiver.

DYPIET, Pune 89
This ROI based watermark extraction process in medical images is depicted in FIGURE 4.7. ALGORITHM 4.8 shows step by step implementation of watermark extraction from medical image.

Algorithm 4.8 : ROI based watermark extraction

<table>
<thead>
<tr>
<th>Input:</th>
<th>Compressed_Watermarked_Image.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>Extracted_Watermark.</td>
</tr>
<tr>
<td>Step-1:</td>
<td>Read Compressed_Watermarked_Image.</td>
</tr>
<tr>
<td></td>
<td>Decompress it to recover Watermarked_Medical_Image.</td>
</tr>
<tr>
<td>Step-2:</td>
<td>Separate RONI.</td>
</tr>
<tr>
<td></td>
<td>Apply one level DWT to retrieve LH1 sub-band.</td>
</tr>
<tr>
<td>Step-3:</td>
<td>Use 4x4 sizes for DCT blocks.</td>
</tr>
<tr>
<td>Step-4:</td>
<td>Generate Pn_Sequence_0 and Pn_Sequence_1, depending upon sum of all elements of mid band used for DCT transformation.</td>
</tr>
<tr>
<td></td>
<td>Use same seed that was used in watermark embedding.</td>
</tr>
<tr>
<td>Step-5:</td>
<td>If rand(state, 15) is used in embedding then,</td>
</tr>
<tr>
<td></td>
<td>Repeat same step here.</td>
</tr>
<tr>
<td></td>
<td>endif</td>
</tr>
<tr>
<td>Step-6:</td>
<td>Extract mid band elements from DCT block.</td>
</tr>
<tr>
<td></td>
<td>Find correlation(extracted mid band coefficients,Pn_Sequence_0)</td>
</tr>
<tr>
<td></td>
<td>Find correlation(extracted mid band coefficients,Pn_Sequence_1).</td>
</tr>
<tr>
<td>Step-7:</td>
<td>If correlation(extracted mid band coefficients, Pn_Sequence_0) $\geq$ correlation(extracted mid band coefficients, Pn_Sequence_1) then</td>
</tr>
<tr>
<td></td>
<td>Scrambled_Watermark bit as 0</td>
</tr>
<tr>
<td></td>
<td>else</td>
</tr>
<tr>
<td></td>
<td>Scrambled_Watermark bit as 1.</td>
</tr>
<tr>
<td></td>
<td>endif</td>
</tr>
<tr>
<td>Step-8:</td>
<td>Apply Arnold Transform to Scrambled_Watermark to get Extracted_Watermark.</td>
</tr>
</tbody>
</table>

FIGURE 4.7. ALGORITHM 4.8 were further extended for telemedicine applications.
4.2.5 Implementation of Tech_Num_5: Image Watermarking Based on DWT-DFT-SVD Domain

4.2.5.1 Watermark Embedding in DWT-DFT-SVD Domain

The proposed watermark embedding process embeds grey scale watermark of size 256x256 into color cover image of size 512x512. The process is formulated by implementing 8-stages security with thresholding and randomization (8-SSTR) as given in TABLE 4.1.

<table>
<thead>
<tr>
<th>Phases</th>
<th>level</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cover Image</td>
<td>L1</td>
<td>Color space conversion</td>
</tr>
<tr>
<td>Processing</td>
<td>L2</td>
<td>DWT sub-band selection with randomization</td>
</tr>
<tr>
<td></td>
<td>L3</td>
<td>Applying 2D DFT</td>
</tr>
<tr>
<td></td>
<td>L4</td>
<td>Singular values decomposition</td>
</tr>
<tr>
<td></td>
<td>L5</td>
<td>Harris corner detection</td>
</tr>
<tr>
<td></td>
<td>L6</td>
<td>PN_Sequence generation</td>
</tr>
<tr>
<td></td>
<td>L7</td>
<td>Thresholding</td>
</tr>
<tr>
<td></td>
<td>L8</td>
<td>Scrambling watermark</td>
</tr>
</tbody>
</table>

Here, watermark embedding process is implemented through 8 levels (L1-L8) providing strong security. Levels L1 to L4 are used for cover image processing whereas levels L5 to L8 are used for watermark processing.

Harris corner detection method is used to determine corner peaks K. This K is used to generate Pn_Sequence from watermark and SUM of Pn_Sequence is computed. Now, depending on SUM, predefined threshold (T) and Arnold periodicity (P), Key for scrambling the watermark is determined.

The detail watermark embedding process for Q component in YIQ color space is given in ALGORITHM 4.9.
Algorithm 4.9 : DWT-DFT-SVD based embedding in YIQ color space

**Input:** Cover Image, Watermark.

**Output:** Watermarked Image

**Step-1:** Read Cover Image and Watermark

**Step-2:** Separate R, G, B components.

**Step-3:** Convert into YIQ using equation 2.45.

**Step-4:** Apply one level DWT to Q to get Q LL1, Q HL1, Q LH1, Q HH1 bands.

**Step-5:** Apply 2D DFT to Q HL1 to get DFT Q HL1

**Step-6:** $[U, S, V] = \text{SVD}(\text{DFT}_Q \text{HL1})$

**Step-7:** Using Harris Corner Detection find K depending on corner peaks.

**Step-8:** Generate Pn Sequence of Watermark using K.

Compute \( \text{SUM} = \text{Summation of Pn Sequence} \).

**Step-9:** T=Threshold, P= Arnold Periodicity, Key+Count P, If SUM > T then

Key=P+Count

else

Key=P-Count

endif

**Step-10:** Using Key scramble Watermark to give Watermark1.

**Step-11:** Find S1 with W1 and flexing factor K1

\( S1 = S + K1 \times \text{Watermark1} \)

**Step-12:** Apply SVD for above S1 component

\([U1, SS, V1] = \text{SVD}(S1)\)

**Step-13:** Find New Q HL1 using SS in step 12.

\( \text{New}_Q\text{HL1} = U \times SS \times V' \)

**Step-14:** Apply inverse DFT to get New DFT Q HL1

**Step-15:** Apply one level DWT to Q LL1, New DFT Q HL1, Q LH1, Q HH1 to get New Q component.

**Step-16:** Combine Y, I and ‘New Q’ to generate New YIQ image.

**Step-17:** Convert New YIQ to RGB using to get Watermarked Image
4.2.5.2 Watermark Extraction in DWT-DFT-SVD Domain

**Algorithm 4.10** shows detail watermark extraction process that extracts grey scale watermark of size 256x256 from color watermarked image of size 512x512.

**Algorithm 4.10 : DWT-DFT-SVD based extraction in YIQ color space**

**Input:** Watermarked Image, Cover Image,

**Output:** Extracted Watermark

**Step-1:** Read Watermarked Image

**Step-2:** Separate R, G, B components.

**Step-3:** Convert RGB into YIQ using equation 2.45.

**Step-4:** Apply one level DWT to Q to get New Q HL1.

**Step-5:** Apply 2D DFT to New Q HL1 to get New DFT Q HL1

**Step-6:** Apply SVD to New DFT Q HL1 as,

\[
[U,U,S,V,V] = \text{SVD}(\text{New DFT Q HL1})
\]

**Step-7:** \( S_{\text{New}} = U^1* S^2 * V^1' \)

**Step-8:** Read Cover Image.

**Step-9:** Separate R, G, B components using equations 2.42, 2.43 and 2.44.

**Step-10:** Convert into YIQ using equation 2.45.

**Step-11:** Apply one level DWT to Q as shown to get Q HL1.

**Step-12:** Apply 2D DFT to Q HL1 to get DFT Q HL1.

**Step-13:** Apply SVD to DFT Q HL1 as,

\[
[U,S,V] = \text{SVD}( \text{DFT Q HL1})
\]

**Step-14:** Using \( S_{\text{New}} \) obtained in step 7 and \( S \) obtained step 13 as,

Scrambled Watermark = \((S_{\text{New}}-S)/K_1\);

Where \( K_1 \) is scale factor.

**Step-15:** Apply Arnold transform to Scrambled Watermark with Key to get Extracted Watermark.

**Algorithm 4.9** and **Algorithm 4.10** were further extended for YUV, YCgCb and LUV color spaces. Finally the combine technique was developed for comparative performance YUV, YIQ, YCgCb and LUV color spaces.
4.2.6 Implementation of Tech_Num_6: DWT-FWHT-SVD Based Image Watermarking Robust to Noise Addition and Filtering Attacks

4.2.6.1 Watermark Embedding in DWT-FWHT-SVD Domain

Algorithm 4.11: DWT-FWHT-SVD based watermark embedding

Input: Cover Image, Watermark.
Output: Watermarked Image.

Step-1: Read grey scale Cover Image of size 512x512.
\([M, N] = \text{size}(\text{Cover Image})\)

Step-2: Apply one level DWT to Cover Image using Haar wavelet to give four non overlapping sub-bands,
\([L_{L1}, H_{L1}, L_{H1}, H_{H1}] = \text{dwt2}(\text{Cover Image}, 'Haar')\);

Step-3: Apply Fast Walse-Hadamard Transform to HL1 sub-band.
\(\text{temp} = \text{fwht}(\text{HL1})\)

Step-4: Apply Singular Value Decomposition (SVD) to temp.
\([U, S, V] = \text{SVD}(\text{temp})\)

Step-5: Read grey scale Watermark of size 256x256.

Step-6: Apply Fibonacci-Lucas rotation affine transform (FLRAT) with angle theta clockwise to give scrambled watermark Watermark1.

Step-7: Perform embedding of Watermark1 with scale factor K1.
\(S_1 = S + K_1 \times \text{Watermark1}\)

Step-8: \([U_1, S_1, V_1] = \text{SVD}(S_1)\)
\(CW_1 = U \times S_1 \times V'\)

Step-9: Apply inverse SVD to get New HL1 component as,
\(\text{New HL1} = \text{ifwht}(\text{CW})\)

Step-10: Perform one level IDWT with New HL1 as,
\(\text{Watermarked Image} = \text{idwt2}(L_{L1}, \text{New HL1}, L_{H1}, H_{H1}, 'haar', [M, N])\);

Step-11: Display Watermarked Image.
**Algorithm 4.11** shows detail embedding processing in DWT-FWHT-SVD domain. The FLRAT is used effectively for implementation of security.

### 4.2.6.2 Watermark Extraction in DWT-FWHT-SVD Domain

**Algorithm 4.12: DWT-FWHT-SVD based watermark extraction**

**Input:** Watermarked Image, Cover Image

**Output:** Extracted Watermark

**Step-1:** Read Watermarked Image of size 512x512.

**Step-2:** Apply One level DWT to Watermarked Image as,

\[
[LL1,HL1,LH1,HH1] = \text{dwt2}(\text{Watermarked Image},'\text{Haar}');
\]

**Step-3:** Apply Fast-Walse-Hadamard Transform,

\[
\text{NCWI} = \text{fwht}(\text{Recovered HL1})
\]

**Step-4:** Apply Singular Value Decomposition

\[
[U,U,S,2,VV] = \text{SVD}(\text{NCWI})
\]

**Step-5:** Read grey scale Cover Image of size 512x512.

**Step-6:** Apply one level DWT to Cover Image using Haar wavelet.

\[
[LL1,HL1,LH1,HH1] = \text{dwt2}(\text{Cover Image},'\text{Haar}');
\]

**Step-7:** Apply Fast Walse-Hadamard Transform to HL1 sub-band.

\[
\text{temp} = \text{fwht} (\text{HL1})
\]

**Step-8:** Apply Singular Value Decomposition (SVD) to temp.

\[
[U,U,S,V] = \text{SVD} (\text{temp})
\]

**Step-9:** Using S2 from step 4 and U1,V1 from step 8 of embedding process.

\[
SN = U1*S2*V1
\]

**Step-10:** Scrambled Watermark can be found as follows,

\[
\text{Scrambled Watermark} = (SN-S)/K1;
\]

Where, K1 is Scale Factor.

**Step-11:** Apply Fibonacci-Lucas rottion affine transform(FLRAT) with angle theta clockwise to get Extracted Watermark.

**Step-12:** Display Extracted Watermark.
ALGORITHM 4.12 shows detail watermark extraction process in DWT-FWHT-SVD domain. Further this technique was extended for comparative performance in DWT-FWHT-SVD and DWT domains keeping all parameters same as given in ALGORITHM 4.11 and ALGORITHM 4.12.

4.2.7 Implementation of Tech_Num_7: Image Watermarking Using MEO

This technique is implemented in DWT-SVD domain. Initially, it was implemented without optimization and further extended with MEO.

4.2.7.1 Watermark Embedding of MEO Based Technique

Initially, as per state with Key1, Pn sequence of watermark is generated. Further average (AVG) of Pn sequence is calculated. The Key K is computed based on threshold (T). This K is used for scrambling the watermark with Fibonacci-Lucas transform.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Detail description of given stage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step-1:</td>
<td>Take cover image into wavelet domain(DWT)</td>
</tr>
<tr>
<td>Step-2:</td>
<td>Pn.Sequence of Watermark at given state with Key1 is generated. AVG=average of Pn.Sequence</td>
</tr>
<tr>
<td>Step-3:</td>
<td>Apply Thresholding with Key1 to generate Key K required for scrambling.</td>
</tr>
<tr>
<td>Step-5:</td>
<td>Apply Singular Value Decomposition.</td>
</tr>
<tr>
<td>Step-6:</td>
<td>Using actual Embedding Formula with given scale factor K1. This K1 is as per optimization in stage 7.</td>
</tr>
<tr>
<td>Step-7:</td>
<td>Optimization of PSNR and NC using MEO.</td>
</tr>
</tbody>
</table>

TABLE 4.2: Applying multistage security provision and ALGORITHM 4.13 gives details steps of watermark embedding process. The sample periodicity of Fibonacci-Lucas transform for M x N images with M=N is found as M.
Chapter 4. System Development and Implementations

Algorithm 4.13: MEO based watermark embedding

**Input:** Cover Image, Watermark.

**Output:** Watermarked Image.

**Step-1:** Read grey scale Cover Image, size MxN.

**Step-2** Apply one level DWT to Cover Image using Haar wavelet.

\[ [LL1,HL1,LH1,HH1] = \text{dwt2}(\text{Cover Image}, 'Haar') \]

**Step-3:** Apply SVD to HL1 sub-band of cover image found in step 2,

\[ [U,S,V] = \text{SVD}(HL1) \]

**Step-4:** Read grey scale watermark W of size 256x256.

**Step-5:** Generate Pn Sequence with key1.

Calculate AVG = average of Pn Sequence.

Here, key1 is as per state watermark.

**Step-6:** Use predefined Threshold T, Predefined counter count , Fibonacci-Lucas Periodicity P and key1 found in step 5 to compute key K.

**Step-7:** Key1+Count \( \leq P \),

If AVG > T then

\[ K = P + \text{Count} \]

else

\[ K = P - \text{Count} \]
endif

**Step-8:** To scramble watermark apply Fibonacci Lucas Transform given in equation 2.25 to get Watermark1.

**Step-9:** Perform Embedding using S from step 3 and scale factor K1 ,

\[ S1 = S + K1 \times \text{Watermark1} \]

\[ [U1,SS,V1] = \text{SVD}(S1) \]

**Step-10:** CWI = \( U^*SS*V' \)

New HL1 = CWI

**Step-11:** Apply one level IDWT with New HL1,

Watermarked Image = idwt2(LL1, New HL1, LH1, HH1, 'Haar',[M,N]).

**Step-12:** Display Watermarked Image and K1.
Chapter 4. System Development and Implementations

Here, we used scrambling key 100 and descrambling key 156 for watermark of size 256x256. K1 is scale factor also called flexing factor that is used for actual watermark embedding.

The grey scale watermark is scrambled to generate watermark1 that is further embedded in cover image by using multiplicative rule.

4.2.7.2 MEO Based Watermark Extraction

**Algorithm 4.14 : MEO based extraction**

| Input: | Watermarked Image, Cover Image. |
| Output: | Extracted Watermark. |
| Step-1: | Read Watermarked Image |
| Step-2: | Apply One level DWT to Watermarked Image, |
|        | [LL1, Recovered HL1, LH1, HH1] = dwt2(Watermarked Image,’Haar’); |
| Step-3: | Apply SVD to Recovered HL1 as, |
|        | [UU,S2,VV]=SVD (Recovered HL1). |
| Step-4: | Read grey scale Cover Image, size MxN. |
| Step-5: | Apply one level DWT to Cover Image using Haar wavelet. |
|        | [LL1, HL1, LH1, HH1]= dwt2(Cover Image,’Haar’); |
| Step-6: | Apply SVD to HL1 sub-band of cover image found in step 2, |
|        | [U,S,V]=SVD(HL1) |
| Step-7: | Use S2 from step 3, U1 and V1 from step 9 of embedding process. |
|        | SN=U1*S2*V1 ' |
| Step-8: | Find Scrambled Watermark using SN,S and scale factor K1 |
|        | Scrambled Watermark=(SN-S)/K1; |
| Step-9: | Apply Fibonacci-Lucas transform with scrambling key K to Scrambled Watermark to get Extracted Watermark. |
| Step-10: | Display Extracted Watermark, NC and K1. |

**ALGORITHM 4.14** gives overall watermark extraction process that is implemented through step-1 step-10.
Initially **Algorithm 4.13** and **Algorithm 4.14** were implemented without optimization. Further they were extended with MEO. The embedding and extraction algorithms were called in function Trial (K1) to include MEO keeping scale factor (K1) flexible. The detail implementation of MEO based GA process is presented in **Algorithm 4.15**.

**Algorithm 4.15: Trial(K1) function for implementation of MEO**

**Input:** Scale Factor (K1) through GA process.

**Output:** PSNR, NC, scale factor K1.

**Step-1:** Specify range of scale factor K1.

**Step-2:** Specify GA parameters: population size, reproduction rate, crossover rate and mutation rate.

**Step-3:** Specify termination criteria by number of generations.

**Step-4:** Write Watermark Embedding Algorithm given in algorithm 4.13

**Step-5:** Apply attack on Watermarked Image.

**Step-6:** Write Watermark Extraction Algorithm given in algorithm 4.14.

**Step-7:** Display parameters in step 8 at end of function as output.

**Step-8:** Display \( y(1) = \text{PSNR}, y(2) = \text{NC} \).

### 4.2.8 Implementation of Tech_Num_8: Applying Image Watermarking to Real World Applications

Here, watermarked techniques are applied to provide security solutions to real world applications. Two sample example solutions are presented here.

#### 4.2.8.1 Implementation of Tech_Num 8.1: Biometric and Wavelet Based E-voting System

Biometric and DWT based watermarking technique in YCgCb color space is proposed as one of the real world application of digital image watermarking.
Chapter 4. System Development and Implementations

The e-voting system allows casting vote through internet without geographical restrictions.

It can be used for conducting presidential, municipal and parliamentary elections by maintaining security in election process.

![Figure 4.8: Overall flow of online voting process.](image)

The secured voting system is part and parcel of any democracy. The manual voting in practice suffers many problems in developing and developed nations.

Though traditional ballot papers and boxes of conventional voting system have been replaced by EVM, fraud voting, booth capturing, voting by poll workers for absent voters are challenging issues to the election commissions of given country.

The strongly secured and highly robust e-voting system in YCgCb color space is presented with effectiveness and accuracy. The overall process of watermark embedding is depicted in **FIGURE 4.8**.
The provision of biometric security is a special feature of the e-voting system.

The Details of online voting protocol are explained through Step-1 to Step-7 as shown in **TABLE 4.3**.

<table>
<thead>
<tr>
<th>Step-1:</th>
<th>In order to cast the vote, voter logs on to system at voting center using his unique identification number and password.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step-2:</td>
<td>After successful login, left fingerprint and right fingerprint of voter are taken on fingerprint sensor.</td>
</tr>
<tr>
<td>Step-3:</td>
<td>The watermark embedding algorithm is executed at voting center and output watermarked photo is sent from remote voting center to server side administrator.</td>
</tr>
<tr>
<td>Step-4:</td>
<td>The watermark extraction algorithm is executed at server side administrator. The cover photo and extracted watermark are available at server end.</td>
</tr>
<tr>
<td>Step-5:</td>
<td>Server administrator compares original fingerprint stored in voter database with extracted fingerprint.</td>
</tr>
<tr>
<td>Step-6:</td>
<td>It authentication is successful; voter is allowed to cast his vote. Server side administrator forwards this message to client voting center.</td>
</tr>
<tr>
<td>Step-7:</td>
<td>After successful voting at voting center, account of voter will be closed. The voted bit will be set to 1 in database indicating that he has no chance to vote again.</td>
</tr>
<tr>
<td>Step-8:</td>
<td>The total vote counter for corresponding election candidate will incremented by 1.</td>
</tr>
</tbody>
</table>

Thumb impressions including left and right fingerprints make the system more reliable and highly secured.
Chapter 4. System Development and Implementations

A] Watermark Embedding In Biometric and Wavelet Based E-voting System

The Algorithm 4.16 is embedding algorithm, used for online voting system.

Algorithm 4.16: Watermark embedding process for e-voting system

<table>
<thead>
<tr>
<th>Input:</th>
<th>Voter_Photo, Left_Fingerprint, Right_Fingerprint.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>Watermarked_Image.</td>
</tr>
<tr>
<td>Step-1:</td>
<td>Read Voter_Photo.</td>
</tr>
<tr>
<td>Step-2:</td>
<td>Separate R, G, B components of Voter_Photo using equations</td>
</tr>
<tr>
<td></td>
<td>Convert RGB to YCgCb color space using equation 2.51.</td>
</tr>
<tr>
<td>Step-3:</td>
<td>Select Cg component and apply 3 level DWT using Haar wavelet.</td>
</tr>
<tr>
<td></td>
<td>select HL3 component.</td>
</tr>
<tr>
<td>Step-4:</td>
<td>Read thumb impressions (Left_Fingerprint and Right_Fingerprint).</td>
</tr>
<tr>
<td></td>
<td>Resize Right_Fingerprint to 64x64 size.</td>
</tr>
<tr>
<td>Step-5:</td>
<td>Using Left_Fingerprint generate key K as shown in Sequence.</td>
</tr>
<tr>
<td></td>
<td>This K will be used as key in Arnold scrambling.</td>
</tr>
<tr>
<td>Step-6:</td>
<td>Consider Right_Fingerprint in 64x64 size as Watermark.</td>
</tr>
<tr>
<td></td>
<td>Apply Arnold transform to Watermark with K to get Watermark1.</td>
</tr>
<tr>
<td></td>
<td>Watermark1 in in scrambled form.</td>
</tr>
<tr>
<td>Step-7:</td>
<td>Perform watermark embedding,</td>
</tr>
<tr>
<td></td>
<td>New_HL3(I,J)=HL3(I,J)+K1*Watermark1(I,J)</td>
</tr>
<tr>
<td></td>
<td>Where K1 is Scale factor.</td>
</tr>
<tr>
<td>Step-8:</td>
<td>Using inverse DWT, construct YCgCb_Image with New Cg component.</td>
</tr>
<tr>
<td>Step-9:</td>
<td>Convert YCgCb to RGB color space using equation 2.52 to give</td>
</tr>
<tr>
<td></td>
<td>Watermarked_Photo.</td>
</tr>
<tr>
<td>Step-10:</td>
<td>Repeat Step-1 to Step-9 for Y and Cb components.</td>
</tr>
</tbody>
</table>

B] Watermark Extraction In Biometric and Wavelet Based E-voting System

The Algorithm 4.17 is extraction algorithm used for online voting system.
Chapter 4. System Development and Implementations

Algorithm 4.17: Watermark extraction process for e-voting system

**Input:** Watermarked_Photo, Voter_Photo

**Output:** Extracted_Right_Fingerprint

**Step-1:** Consider Watermarked_photo and convert from RGB colorspace to YCgCb using equation 2.51.

**Step-2:** Consider New_Cg component in HL3 region using DWT transform up to 3 levels to give New_HL3 component.

**Step-3:** Read Voter_Photo.

- Convert into YCgCb color space using equation 2.51.
- Decompose Cg component up to 3 levels to give HL3 component.

**Step-4:** Apply Extraction formula as follows,

\[ \text{Value}(i,j) = \frac{\text{abs}(H_{L3}(i,j), H_{L3}'(i,j))}{K1} \]

**Step-5:** If Value(i,j) < Threshold_Value

- then Scrambled_Watermark(i, j) = 0
- else

- Scrambled_Watermark(i, j) = 1

**Step-6:** Apply Arnold Scrambling to Scrambled_Watermark to give Extracted_Right_Fingerprint.

**Step-7:** Repeat Step-1 to Step-5 for Y and Cb components.

4.2.8.2 Implementation of Tech_Num 8.2: Watermarking based Criminal Authentication System

A] Watermark Embedding for Criminal Authentication System

Here, the watermark embedding is done in YCgCb space. This is non-blind technique where three levels DWT decomposition is done using Haar wavelet.

The IPMS is used to generate watermark of size 64x64 from watermark of 128x128.
Chapter 4. System Development and Implementations

Further Arnold scrambling is used for watermark scrambling.

The detail watermark embedding is given in **Algorithm 4.18**.

![Algorithm 4.18: Watermark embedding for criminal authentication system](image)

**Input:** Criminal Image of size 512x512, Watermark of size 128x128.

**Output:** Watermarked Image

**Step-1:** Read Criminal Image

**Step-2:** Separate R, G, B components.

Convert RGB to YCgCb color space using equation 2.51.

**Step-3:** Apply 3 level DWT using Haar wavelet and select HL3 region.

**Step-4:** Read Watermark of size 128x128.

**Step-5:** Apply image partition merging scheme two S1 and S2 of size 128x128 as shown in equations 2.35 and 2.36.

**Step-6:** Find merged share M1 of sizes 64x128 as shown in equation 2.37.

**Step-7:** Generate two shares S11 and S12 of 64x128 sizes as per equations 2.35, 2.36.

**Step-8:** Find M11 of size 64x64 as shown in equation 2.37.

**Step-9:** Apply Arnold scrambling to M11, where K is Key used for scrambling.

**Step-10:** Perform Watermark embedding,

New_HL3(I,J)=HL3(I,J)+K1*M11(I,J)

**Step-11:** Using inverse DWT, construct YCgCb Image with New_Cb component.

**Step-12:** Convert YCgCb to RGB color space using equation 2.52 to give Watermarked Image.

**B] Watermark Extraction for Criminal Authentication System**

As the method is non-blind, both watermarked image and criminal image are used in extraction.

The detail extraction process is given in **Algorithm 4.19**.
Algorithm 4.19: Watermark extraction for criminal authentication system

**Input:** Watermarked_photo, Cover Image.

**Output:** Final Watermark of size 128x128.

**Step-1:** Consider Watermarked_Image.
Separate R, G, B components.
Convert RGB to YCgCb color space using equation 2.51.

**Step-2:** Consider New_Cb component in HL3 region using DWT transform up to 3 levels to give New_HL3 component.

**Step-3:** Decompose Cover Image upto 3 levels to give HL3 component.

**Step-4:** Apply extraction formule as follows,
\[ \text{Value}(I,J) = \text{abs}(\text{New}_{-}\text{HL3}(I,J),\text{HL3}(I,J))/K1 \]
where K1 is scale factor. Let Threshold equals to 200;
If Value (I, J) < Threshold_Value
Extracted_Watermark (I, J) = 0
else Extracted_Watermark (I, J) = 1.

**Step-5:** Apply Arnold Scrambling to Extracted watermark to give Recovered_M11 by using Key K that is used in embedding process.

**Step-6:** Apply image partition merging scheme logic to find Recovered_S11 and Recovered_S12 from Recovered_M11.

**Step-7:** Find Recovered_M1 using Recovered_S1 and Recovered_S2.

**Step-8:** find Recovered_S1 and Recovered_S2 from Recovered_M1 by image partition merging scheme.

**Step-9:** Generate Final Watermark of 128x128 using Recovered_S1 and Recovered_S2.

**Step-10:** Repeat step-1 to step-9 for Y and Cg components.

Similar steps of embedding algorithm and extraction algorithm are implemented for LUV color space.
Chapter 4. System Development and Implementations

In summary this chapter presents platform, software, hardware specifications and online database specifications for grey scale, color and medical image. This chapter has investigated various grey scale, color and medical image watermarking techniques. The watermark embedding algorithm, watermark extraction algorithm, supporting block diagrams and necessary implementation details are presented under individual watermarking technique for the proposed system.