Chapter 3

Enhanced LSB Steganography

3.1 Introduction

LSB steganography techniques are the simplest and well known techniques. The LSB techniques are very much vulnerable to various steganalytic attacks. So LSB steganography needs some enhancement. In this Chapter we have discussed two techniques enhanced over LSB substitution. In both of these techniques we use cryptographic algorithms to encrypt the secret message and then embed it in the image. Thus these techniques provide two levels of security. The first technique is discussed in section 3.2 and the second technique is discussed in section 3.3. In the first technique a cryptographic algorithm called a new block cipher is used for encryption/decryption and a message bit dependent randomized LSB substitution technique is used as steganography. The second technique is indicator and user key based steganography for RGB images wherein RSA algorithm is used for encryption/decryption. Out of three channels in a pixel of RGB image one channel is used as indicator and the other two channels are used as data channels. Here also LSB substitution is used in connection with some conditions. A comparison study between these two techniques is presented in section 3.4, and the Chapter is concluded in section 3.5.
3.2 A Technique for Secret Communication Using a New Block Cipher with Dynamic Steganography

This technique uses both cryptography and steganography. The cryptographic algorithm is a block cipher with a block length of 128 bits and key length of 256 bits. The secret message is encrypted by this block cipher. Two cipher text bits are to be embedded in each pixel of the image, wherein each pixel comprises of 8 bits. The embedding locations in a pixel are 6th and 7th bit locations or 7th and 6th bit locations or 7th and 8th bit locations or 8th and 7th bit locations depending upon the cipher text bits. The 8th bit means the least significant bit (LSB). As the embedding locations are decided at the run time of the algorithm, therefore it is called as dynamic steganography.

The new block cipher is illustrated in section 3.2.1 and the embedding/extraction algorithms are discussed in section 3.2.2, followed by the algorithms in section 3.2.3 and the results in section 3.2.4.

3.2.1 The New Block Cipher with 256-bit Key

This proposed block cipher has a block length of 128 bits (16 characters) and key length of 256 bits (32 characters). The subkey generation, encryption and decryption processes of this block cipher are as discussed in the following sub-sections.

3.2.1.1 Subkey Generation

The key which is 32 characters (256 bits) is represented as a matrix with 4 rows and 8 columns, say K. The eight subkeys K1, K2, K3, K4, K5, K6, K7 and K8 are generated from it. The first subkey, K1 is a 4x4 matrix obtained by taking the first 4 columns of K and fifth subkey, K5 is a 4x4 matrix obtained by taking the remaining 4 columns of K. The second subkey, K2 is obtained by interchanging the first and second column of K1. The third subkey, K3 is obtained by interchanging the first and third column of K1. The fourth subkey, K4 is obtained by interchanging the first and fourth column of K1. The sixth subkey, K6 is obtained by interchanging the first and second column of K5. The seventh subkey, K7 is obtained by interchanging the first and third column of K5. The eighth subkey, K8 is obtained by
interchanging the first and fourth column of $K_5$. In this way 8 subkeys namely $K_1$, $K_2$, $K_3$, $K_4$, $K_5$, $K_6$, $K_7$ and $K_8$ are generated.

3.2.1.2 Encryption

The message is divided into different blocks. Each block comprising of 16 characters. If the last block is less than 16 characters, then blank spaces can be appended at the end of it to make the length 16 characters. Each block is represented as a 4x4 matrix, say $P$. The encryption of a message block is done by the following procedure.

Encryption procedure:

For $i = 1$ to $8$
{ 
\[ P_1 = (K_i \times P) \mod 127 \] 
$P = P_1$
}

For $i = 1$ to $8$
{ 
\[ P_2 = \text{stir} \ (P) \] 
\[ P_3 = \text{XOR} \ (K_i, \ P_2) \] 
$P = P_3$
}

\[ C = P \]

Where $K \times P$ is the matrix multiplication of the two 4x4 matrices $K$ and $P$. Stir and XOR operations are as discussed in following sub-sections. $P_1$, $P_2$ and $P_3$ are the intermediate results.

3.2.1.3 Decryption

The cipher text block $C$ can be converted to plain text block $P$ by the following procedure.

Decryption procedure:

For $i = 8$ to $1$
{ 
\[ C_1 = \text{XOR} \ (K_i, \ C) \] 
\[ C_2 = \text{stir} \ (C_1) \] 
\[ C = C_2 \]
}

For $i = 8$ to $1$
{ 
\[ C_3 = (K^{-1}_i \times C) \mod 127 \] 
\[ C = C_3 \]
}

$P = C$
Where \( P \) is the required plain text and \( C_1, C_2, C_3 \) are intermediate results. \( K^{-1}_{mi} \) is the modular arithmetic inverse (calculated by using the extended Euclid algorithm) of the matrix \( K_i \).

### 3.2.1.4 Stir operation

If Matrix \( A = 
\begin{bmatrix}
11001100, & 00101010, & 11100110, & 11001100 \\
11011110, & 10101010, & 00100100, & 01010111 \\
00011001, & 11101111, & 01111000, & 11000111 \\
11111100, & 11011101, & 11011100, & 10011001 \\
\end{bmatrix}
\)

Then, \( B = \text{Stir} (A) = 
\begin{bmatrix}
11001111, & 00101000, & 11100111, & 00101000 \\
11100011, & 01101001, & 11100101, & 10100011 \\
00110111, & 01101100, & 10111001, & 01110011 \\
11111110, & 11010101, & 11101110, & 00110001 \\
\end{bmatrix}
\)

And \( \text{Stir} (\text{Stir} (A)) = 
\begin{bmatrix}
11001100, & 00101010, & 11100110, & 11001100 \\
11011110, & 10101010, & 00100100, & 01010111 \\
00011001, & 11101111, & 01111000, & 11000111 \\
11111100, & 11011101, & 11011100, & 10011001 \\
\end{bmatrix}
\)

This stir operation is defined by the following steps.

1. The first and second from each byte in a row of \( A \) are combined to form the first byte of \( B \) in that row.
2. The third and fourth bits from each byte in a row of \( A \) are combined to form next byte of \( B \) in that row.
3. The fifth and sixth bits from each byte in a row of \( A \) are combined to form next byte of \( B \) in that row.
4. The seventh and eighth bits from each byte in a row of \( A \) are combined to form the last byte of \( B \) in that row.

The stir operation is reversible i.e. \( \text{Stir} (\text{Stir} (A)) = A \).
3.2.1.5 XOR operation

If A and B are two matrices of same order, then XOR(A, B) is the bit by bit exclusive-OR operation of the respective elements of the two matrices. For example if A=10101011 and B=10011001, then C= XOR (A, B) = 00110010. The XOR operation is reversible i.e. If C=XOR (A, B) then A=XOR(C, B) and B=XOR(C, A).

The subkey generation, stir operation, modular arithmetic inverse and XOR operation together make the encryption and decryption methods strong. As a result the algorithm becomes very robust.

3.2.2 Embedding and Retrieving Techniques

The embedding technique is described in sub-section 3.2.2.1 and the retrieving technique is described in sub-section 3.2.2.2.

3.2.2.1 The Embedding Technique

The cover image is converted to binary. Each pixel becomes one byte. The length of cipher text is computed, say it is L. At the beginning of the image first 3000 pixels are not disturbed as these pixels may carry the image characteristics. Then another 300 pixels are reserved for hiding L. Thus the embedding of cipher text bits starts from 3301th pixel. The embedding procedure is as discussed in the following steps.

Step 1: Embed L in the reserved pixels (starting from 3001th pixel upto 3300th pixel) using two least significant bit substitution. Embed the first two bits of cipher text at 7th and 8th bit locations of 3301th pixel. Take next two bits of cipher text. Set i= 3301. Set L= L -2.

Step 2: Do any of the four sub-steps (a) through (d)

(a) If the two bits of cipher text embedded at the i\textsuperscript{th} pixel of image are 00, then the next two bits of cipher text will be embedded in 7\textsuperscript{th} and 8\textsuperscript{th} bit locations of the (i+1)\textsuperscript{th} pixel of the image.

(b) If the two bits of cipher text embedded at the i\textsuperscript{th} pixel of image are 01, then the next two bits of cipher text will be embedded in 8\textsuperscript{th} and 7\textsuperscript{th} bit locations of the (i+1)\textsuperscript{th} pixel of the image.
(c) If the two bits of cipher text embedded at the \( i \)th pixel of image are 10, then the next two bits of cipher text will be embedded in 6\( \text{th} \) and 7\( \text{th} \) bit locations of the \((i+1)\)th pixel of the image.

(d) If the two bits of cipher text embedded at the \( i \)th pixel of image are 11, then the next two bits of cipher text will be embedded in 7\( \text{th} \) and 6\( \text{th} \) bit locations of the \((i+1)\)th pixel of the image.

Step 3: Set \( i= i+1 \) and \( L=L-2 \). If \( L \) is equal to 0, then go to step 4. Otherwise take the next two bits of cipher text and go to step 2.

Step 4: Stop.

For example let us assume that the data to be embedded is 11001001 00011010 01100110 10011111. Suppose the different pixels of image where cipher text is embedded are named as A, B, C, D etc. Then the embedding into the different locations of the different pixels are as shown in Table 3.1. As any two LSBs are changed out of three LSBs, the maximum change in color value of a pixel can be 7. In 8-bit gray scale 256 colors are defined, the change in color value of a pixel can not be noticeable by human visual system.

<table>
<thead>
<tr>
<th>Image pixel (each 1 byte)</th>
<th>Operation</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>pixel A</td>
<td>embedd 11</td>
<td>7( \text{th} ) and 8( \text{th} )</td>
</tr>
<tr>
<td>pixel B</td>
<td>embedd 00</td>
<td>7( \text{th} ) and 6( \text{th} )</td>
</tr>
<tr>
<td>pixel C</td>
<td>embedd 10</td>
<td>7( \text{th} ) and 8( \text{th} )</td>
</tr>
<tr>
<td>pixel D</td>
<td>embedd 01</td>
<td>6( \text{th} ) and 7( \text{th} )</td>
</tr>
<tr>
<td>pixel E</td>
<td>embedd 00</td>
<td>8( \text{th} ) and 7( \text{th} )</td>
</tr>
<tr>
<td>pixel F</td>
<td>embedd 01</td>
<td>7( \text{th} ) and 8( \text{th} )</td>
</tr>
<tr>
<td>pixel G</td>
<td>embedd 10</td>
<td>8( \text{th} ) and 7( \text{th} )</td>
</tr>
<tr>
<td>etc.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.2.2.2 The Retrieving Technique

Step 1: Compute the length of the embedded message \( L \) by retrieving bits from the reserved pixels. Retrieve the first two cipher text bits from the 7\( \text{th} \) and 8\( \text{th} \) bit location of the
3301th pixel. Initialize the variable CIPHER with these two bits. Set L=L-2. Initialize i=3301.

Step 2: Do any one of the four sub-steps (a) through (d)
(a) If the retrieved bits from the ith pixel are 11, then retrieve 7th and 6th bits from the (i+1)th pixel.
(b) If the retrieved bits from the ith pixel are 10, then retrieve 6th and 7th bits from the (i+1)th pixel.
(c) If the retrieved bits from the ith pixel are 01, then retrieve 8th and 7th bits from the (i+1)th pixel.
(d) If the retrieved bits from the ith pixel are 00, then retrieve 7th and 8th bits from the (i+1)th pixel.

Step 3: Append these two retrieved bits to the variable CIPHER. Set L=L-2.

Step 4: If (L > 0) go to step 2, otherwise go to step 5.

Step 5: Stop.

3.2.3 The Proposed Algorithms for Dynamic Steganography

The algorithm for embedding at the sender is stated in sub-section 3.2.3.1 and the extraction algorithm at the receiver is stated in sub-section 3.2.3.2.

3.2.3.1 The algorithm at Sender

The algorithm at sender is represented by the following steps.
1. Convert the carrier image to binary.
2. Divide the secret message into blocks such that each block comprises of 16 characters.
3. Apply encryption process to convert each plain text block into a cipher text block.
4. Keep all the cipher text blocks together to form the complete cipher text.
5. Transform these cipher text to binary.
6. Embed the cipher text into binary image as per the embedding process discussed above and then we get the stego-binary-image. Now convert this stego-binary-image to stego-image and then send to receiver.

3.2.3.2 The Algorithm at Receiver

The algorithm at Receiver is as represented by the following steps.
1. Convert the received image to binary.
2. Retrieve the embedded cipher text bits (two from each pixel) from the stego-binary-image as per the retrieving process discussed above.
3. Keep them together, and divide into blocks with 16 characters in each.
4. Apply decryption process to each cipher text block to get the plain text block.
5. Keep together all the plain text blocks, thus we get the secret message.

### 3.2.4 Results and Discussion

The technique is implemented using java programming language. Five sample experimental observations are as discussed below. Fig.3.1 (a) is the original Lena image, (b) is its bar chart (equivalent to histogram in MATLAB), (c) is the stego-Lena image with 40 kilo bytes of cipher text embedded, and (d) is the bar chart of stego-Lena image. Fig.3.2 (a) is the original Fruits image, (b) is its bar chart, (c) is the stego-Fruits image with 40 kilo bytes of cipher text embedded, and (d) is the bar chart of stego-Fruits image. Fig.3.3 (a) is the original Player image, (b) is its bar chart, (c) is the stego-Player image with 50 kilo bytes of cipher text embedded, and (d) is the bar chart of stego-Player image. Fig.3.4 (a) is the original Tree image, (b) is its bar chart, (c) is the stego-Tree image with 60 kilo bytes of cipher text embedded, and (d) is the bar chart of stego-Tree image. Fig.3.5 (a) is the original Lady-Man image, (b) is its bar chart, (c) is the stego-Lady-Man image with 50 kilo bytes of cipher text embedded, and (d) is the bar chart of stego-Lady-Man image.

The PSNR, correlation, K-L Divergence and bits per pixel (BPP) for the proposed scheme is recorded in Table 3.2. The PSNR is a measure of distortion. More is the PSNR means less is the distortion. The correlation is a measure of the similarity between the original image and the stego-image. The MATLAB built in function corr2(p, q) evaluates the correlation between the cover image, p and the stego-image, q. The maximum value of corr2(p, q) can be 1, if p and q are the same images. So if distortion is lesser, then the correlation can be higher. The readings in Table 3.3 are for 2-bit LSB substitution. After comparing these two tables it can be found that, the proposed scheme performs better as compared to 2-bit LSB substitution. K-L divergence is used to measure the difference between the histograms of original and stego-images.
Enhanced LSB Steganography

Fig. 3.1 (a) Lena image, (b) bar chart of Lena image, (c) stego-Lena image, (d) bar chart of stego-Lena image

Fig. 3.2 (a) Fruits image, (b) bar chart of Fruits image, (c) stego-Fruits image, (d) bar chart of stego-Fruits image
Fig. 3.3 (a) Player image, (b) bar chart of Player image, (c) stego-Player image, (d) bar chart of stego-Player image

Fig. 3.4 (a) Tree image, (b) bar chart of Tree image, (c) stego-Tree image, (d) bar chart of stego-Tree image
This proposed technique is secure as it uses dynamic embedding i.e. message dependent embedding. Unlike traditional LSB substitution this technique can not be attacked by simple attacks like RS steganalysis because the embedding is done in two bit locations out of 6th, 7th and 8th bit locations in a randomized manner depending upon the message bits. If we observe the bar chart in the different Figures, there is no difference between the bar charts of original images and the bar charts of stego-images. Moreover the plain text is not hidden; its cipher text is hidden, which adds another level of security. The encryption algorithm is a new block cipher which uses a 256-bit key, so a stronger algorithm. In each pixel two bits can be embedded, so the amount of message that can be embedded is almost one fourth of the image size. The capacity is doubled as compared to traditional one least significant bit substitution, and it is same as traditional 2-bit LSB substitution. By observing the stego-images from Fig.3.1-3.5, one can not find any visual artifacts, thus the technique is highly imperceptible.
Table 3.2 PSNR, Correlation and K-L Divergence values in proposed technique

<table>
<thead>
<tr>
<th>Image name</th>
<th>Image size (KB)</th>
<th>Payload (cipher) (KB)</th>
<th>BPP</th>
<th>PSNR (dB)</th>
<th>Correlation</th>
<th>K-L Divergence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>768</td>
<td>40</td>
<td>0.42</td>
<td>50.98</td>
<td>0.9998</td>
<td>666.1</td>
</tr>
<tr>
<td>Fruits</td>
<td>1186</td>
<td>40</td>
<td>0.27</td>
<td>52.88</td>
<td>0.9999</td>
<td>46.9</td>
</tr>
<tr>
<td>Player</td>
<td>1837</td>
<td>50</td>
<td>0.22</td>
<td>53.82</td>
<td>0.9999</td>
<td>582.4</td>
</tr>
<tr>
<td>Tree</td>
<td>1749</td>
<td>60</td>
<td>0.28</td>
<td>52.81</td>
<td>0.9999</td>
<td>1783.1</td>
</tr>
<tr>
<td>Lady-Man</td>
<td>1149</td>
<td>50</td>
<td>0.35</td>
<td>51.75</td>
<td>0.9999</td>
<td>686.8</td>
</tr>
</tbody>
</table>

Table 3.3 PSNR, Correlation and K-L Divergence values in 2-bit LSB substitution

<table>
<thead>
<tr>
<th>Image name</th>
<th>Image size (KB)</th>
<th>Payload (KB)</th>
<th>BPP</th>
<th>PSNR (dB)</th>
<th>Correlation</th>
<th>K-L Divergence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>768</td>
<td>40</td>
<td>0.42</td>
<td>50.72</td>
<td>0.9998</td>
<td>913.4</td>
</tr>
<tr>
<td>Fruits</td>
<td>1186</td>
<td>40</td>
<td>0.27</td>
<td>52.61</td>
<td>0.9999</td>
<td>1160.7</td>
</tr>
<tr>
<td>Player</td>
<td>1837</td>
<td>50</td>
<td>0.22</td>
<td>53.54</td>
<td>0.9999</td>
<td>1913.1</td>
</tr>
<tr>
<td>Tree</td>
<td>1749</td>
<td>60</td>
<td>0.28</td>
<td>52.57</td>
<td>0.9999</td>
<td>2491.0</td>
</tr>
<tr>
<td>Lady-Man</td>
<td>1149</td>
<td>50</td>
<td>0.35</td>
<td>51.64</td>
<td>0.9999</td>
<td>9543.1</td>
</tr>
</tbody>
</table>

Fig. 3.6 (a) and (b) represents the variation of PSNR with payload in Lena and Fruits images respectively. From the graphs we can observe that if we go on increasing the payload up to a certain maximum allowable value, the PSNR may not reach less than 40.
Fig. 3.6 Variation of PSNR with payload of (a) Lena and (b) Fruits Images
3.3 A Novel Approach to RGB Channel Based Image Steganography Technique

A novel approach to RGB channel based steganography technique is proposed in this section. The RSA algorithm is used for encryption and decryption. In an RGB image each pixel comprises of 24 bits (3 bytes). The first byte is known as R-channel, the second byte is known as G-channel and the third byte is known as B-channel. The image is divided into 8 blocks and the cipher text is divided into 8 blocks. One cipher block is allocated to be embedded in only one image block depending upon a user defined subkey. Out of the three channels in each pixel of the image one is used as the indicator channel. The indicator channel for the different blocks is not the same. The other two channels (called data channels) are used for hiding cipher text bits in 4 least significant bit (LSB) locations. In a data channel four bits of cipher text can be embedded, if after embedding the change in pixel value is less than or equal to 7. The two LSBs of indicator will represent whether the cipher text is embedded in only one data channel or in both data channels, so that retrieving can be done accordingly at the receiver.

3.3.1 The Algorithms for RGB Channel Based Steganography

The algorithms followed at sender and receiver are discussed in sub-sections 3.3.1.1 and 3.3.1.2 respectively. The steps in the algorithms are illustrated with appropriate examples too.

3.3.1.1 The Algorithm at Sender

1. Divide the image into 8 blocks after leaving some reserved bytes (say, the initial 3099 bytes) at the beginning of the image. The 8 image blocks are \(B_0, B_1, B_2, B_3, B_4, B_5, B_6, \text{ and } B_7\).

2. Encrypt the plain text by using RSA algorithm, thus the cipher text is obtained.

3. Divide the cipher text into 8 blocks in the following manner.

   The cipher text comprises of so many bytes. Each byte is 8 bits. Keep together the first bits (most significant bits) of all the bytes, it becomes message block \(M_0\). Keep together the second bits of all the bytes, it becomes message block \(M_1\). Keep together the third bits of all the bytes, it becomes message block \(M_2\). Keep together the fourth bits of all the bytes, it becomes message block \(M_3\). Keep together the fifth bits of all the bytes, it becomes message
block $M_4$. Keep together the sixth bits of all the bytes, it becomes message block $M_5$. Keep together the seventh bits of all the bytes, it becomes message block $M_6$. Keep together the eighth bits (least significant bits) of all the bytes, it becomes message block $M_7$.

4. Enter the subkey $K_1$ which is to be used for the allocation of message blocks to image blocks for possible embedding. This is a string of digits with a length of 8, formed by the digits 0 through 7, such that each digit occurs only once in the string.

   For example a subkey $K_1 = 05432167$ means message block $M_0$ is to be allocated to image block $B_0$, message block $M_5$ is to be allocated to image block $B_1$, message block $M_4$ is to be allocated to image block $B_2$, message block $M_3$ is to be allocated to image block $B_3$, message block $M_2$ is to be allocated to image block $B_4$, message block $M_1$ is to be allocated to image block $B_5$, message block $M_6$ is to be allocated to image block $B_6$, and message block $M_7$ is to be allocated to image block $B_7$.

5. The second subkey $K_2$, which tells about the indicator channels of all the 8 different image blocks, is to be calculated from the image blocks. From each block one indicator channel is calculated. One of the red, green and blue channels will be the indicator channel, whose sum over all the pixels in that block is maximum. For example suppose sum1, sum2 and sum3 are the sum of red, green and blue channels of different pixels in a block respectively. If the largest is sum2, then green channel (G) is the indicator channel and red (R) and blue (B) channels are the data channels. Suppose R is the indicator channel in image block $B_0$, B is the indicator channel in image block $B_1$, B is the indicator channel in image block $B_2$, R is the indicator channel in image block $B_3$, G is the indicator channel in image block $B_4$, B is the indicator channel in image block $B_5$, G is the indicator channel in image block $B_6$ and R is the indicator channel in image block $B_7$, then the sequence is RBBRGBGR. Moreover, R corresponds to 0, G corresponds to 1 and B corresponds to 2. Therefore, the sequence is 02201210, which is the subkey $K_2$. Thus our key $K = K_1 K_2 = 05432167 02201210$, which is to be hidden in the reserved location of the image (byte number 3001 to 3099) along with block length of image using two least significant bits in each byte.

   Suppose, I is the indicator channel and the other two channels are channel1 and channel2. If R is the I channel, then G is channel1 and B is channel2. If G is the I channel, then R is channel1 and B is channel2. If B is the I channel, then R is channel1 and G is channel2.
6. Thus the different message blocks are allocated to different image blocks and in each image block the indicator is decided. Now the embedding in a block is done as follows.

(a) Take the next pixel of the image block. Take 4 bits of the cipher text.

(b) Compare 4 LSBs of channel1 with 4 bits of cipher text. If the difference is less than or equal to 7, then embed these 4 cipher text bits at those 4 LSBs of channel1. Take next 4 bits of cipher text and go to step 6(c). Otherwise (if the difference is greater than 7) with the same 4 bits of cipher text go to step 6(c).

(c) Compare 4 least significant bits of channel2 with 4 bits of cipher text. If the difference is less than or equal to 7, then embed these 4 cipher text bits at those 4 LSBs of channel2. Otherwise do not embed in channel2, and those 4 cipher text bits are to be considered for the next pixel.

(d) Go to step 6(a).

7. For a pixel if embedding is done as per step 6(b) only i.e. data is embedded in channel1 only, then the two LSBs of its indicator will be set to 00. If for a pixel embedding is done as per step 6(c) only i.e. data is embedded in channel2 only, then the two LSBs of its indicator will be set to 01. If for a pixel embedding is done as per both the steps 6(b) and 6(c) i.e. 4 bits embedded in channel1 and 4 bits embedded in channel2, then the two LSBs of indicator will be set to 10. If in a pixel data is embedded neither in channel1 nor in channel2, then the last two LSBs of its indicator will be set to 11.

8. Stop.

As an example suppose we have the three pixels as in Table 3.4 and the green channel is the indicator channel. Suppose the cipher text stream is 0010 1010 0000 1111 0011. In pixel1 channel1 is 10010110 = 150, if the four bits 0010 are embedded it becomes 10010010 = 146; the difference between 150 and 146 is less than 7, so embedding is done. Now channel2 of pixel1 is 00011110 = 30, the next 4 cipher text bits are 1010, if these four bits are embedded it becomes 00011010 = 26, the difference is less than 7, so embedding is done. The last two bits of indicator of pixel1 are made 10.
In pixel2, channel1 is 00001111 = 15, the next 4 cipher text bits are 0000, if embedded it becomes 00000000 = 0, and the difference is more than 7, so embedding is not done. Then check the channel2 of pixel2. It is 11000100 = 196, the cipher text bits are 0000, if embedded it becomes 11000000 = 192, and the difference is less than 7, so embedding is done. Last two bits of indicator are changed to 01.

In pixel3, channel1 is 10111101 = 189, the next 4 cipher text bits are 1111, if embedded, it becomes 10111111 = 191, and the difference is less than 7, so embedding is done. The channel2 of pixel3 is 11001100 = 204, the next 4 bits of cipher text are 0011, if embedded the channel value becomes 11000011 = 195, the difference is more than 7, so embedding is not done. Thus the last two bits of indicator become 00. After embedding the pixels are also shown in Table 3.4 (lower part).

<table>
<thead>
<tr>
<th>Table 3.4 Example of embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before embedding</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Red (channel1)</td>
</tr>
<tr>
<td>Pixel1</td>
</tr>
<tr>
<td>Pixel2</td>
</tr>
<tr>
<td>Pixel3</td>
</tr>
<tr>
<td>After embedding</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Red (channel1)</td>
</tr>
<tr>
<td>Pixel1</td>
</tr>
<tr>
<td>Pixel2</td>
</tr>
<tr>
<td>Pixel3</td>
</tr>
</tbody>
</table>

3.3.1.2 The Algorithm at Receiver
1. Retrieve the subkey $K_1$, subkey $K_2$, and the length of the blocks from the reserved location in the image i.e. from byte numbers 3001 to 3099.
2. Suppose $K_1 = 05432167$ and $K_2 = 02201210$. In $K_2$, 0 means red, 1 means green and 2 means blue. Thus $K_2 = 02201210$, means red is the indicator channel in image block $B_0$, blue is the
indicator channel in image block $B_1$, blue is the indicator channel in image block $B_2$, red is the indicator channel in image block $B_3$, green is the indicator channel in image block $B_4$, blue is the indicator channel in image block $B_5$, green is the indicator channel in image block $B_6$ and red is the indicator channel in image block $B_7$.

3. For each block identify the channel1 and channel2 as the indicator channel is identified. If red is the indicator channel, then green is channel1 and blue is channel2. If green is the indicator channel, then red is channel1 and blue is channel2. If blue is the indicator channel, then red is channel1 and green is channel2.

4. Declare the variable CIPHER and the variables $CIPHER_i$, for $i=0$ to $7$.

   For $i=0$ to $7$ do the following.

   Take the image Block $B_i$. Start from the first pixel.

   (a) Read the indicator of the pixel.

   (b) If the last two bits of indicator are 00, retrieve the 4 LSBs of channel1 and append to the variable $CIPHER_i$.

   (c) If the last two bits of indicator are 01, retrieve the 4 LSBs of channel2 and append to the variable $CIPHER_i$.

   (d) If the last two bits of indicator are 10, retrieve the 4 LSBs of channel1 and channel2 and append to the variable $CIPHER_i$.

   (e) If the last two bits of indicator are 11, neither retrieve from channel1 nor retrieve from channel2 of that pixel.

   (f) Take the next pixel and go to 4(a).

5. Concatenate all the cipher blocks i.e. $CIPHER_1$ to $CIPHER_7$ to get the CIPHER.

6. Now apply RSA decryption to CIPHER to get the plain text.

7. Stop.

3.3.2 The Experimental Results and Discussion

The technique is implemented using java programming language. Four sample experimental observations are as discussed below. Fig.3.7 (a) is the original Lena image and (b) is its stego-image with 14869 bytes of cipher text embedded into it, (c) is the Garden-Home image and (d) is its stego-image with 20047 bytes of cipher text embedded in it, (e) is the Road image and (f) is its stego-image with 15039 bytes of cipher text embedded in it, (g) is the Player image and (h) is its stego-image with 30079 bytes of cipher text embedded in it.
The PSNR, correlation and K-L Divergence values in proposed technique at different payloads for different images is as given in Table 3.5, and for Gutab et al.’s [19] technique in Table 3.6. In Gutab et al.’s technique only two LSB planes are used, but in our proposed technique four LSB planes are used. Although the results are not better than Gutab et al.’s technique, but the security is improved because of the eight block concept and use of secret key to allocate a message block to an image block.

The performance of various steganographic methods can be rated by the three parameters; (i) security, (ii) capacity and (ii) imperceptibility. This proposed algorithm is secure as it uses allocation of message blocks to image blocks through a subkey and the indicator channel is different in different blocks. The pixel indicator techniques proposed by Gutab et al. [19] and Kaur et al. [24] do not split the message into blocks. In our proposed technique each character from the message is converted to eight bits and these eight bits are distributed to eight message blocks. These eight message blocks are allocated to eight image blocks through a user defined key for embedding. This thing makes the technique advantageous over the other pixel indicator techniques. Moreover we are hiding the encrypted text, not the plain text, which adds another level of security. The encryption algorithm is the RSA algorithm. From Fig.3.7 one can observe...
that there is no visual artifacts with the stego-images, they are looking exactly same as their corresponding original images.

Table 3.5 PSNR, correlation and K-L Divergence values in proposed technique

<table>
<thead>
<tr>
<th>Image name</th>
<th>Image size (KB)</th>
<th>Payload (cipher) (bytes)</th>
<th>BPP</th>
<th>PSNR (dB)</th>
<th>Correlation</th>
<th>K-L Divergence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>768</td>
<td>14869</td>
<td>0.45</td>
<td>49.67</td>
<td>0.9998</td>
<td>1235</td>
</tr>
<tr>
<td>Garden-Home</td>
<td>1214</td>
<td>20047</td>
<td>0.39</td>
<td>50.26</td>
<td>0.9998</td>
<td>2019</td>
</tr>
<tr>
<td>Road</td>
<td>1050</td>
<td>15039</td>
<td>0.34</td>
<td>50.79</td>
<td>0.9999</td>
<td>1513</td>
</tr>
<tr>
<td>Player</td>
<td>1837</td>
<td>30079</td>
<td>0.38</td>
<td>50.35</td>
<td>0.9999</td>
<td>5167</td>
</tr>
</tbody>
</table>

Table 3.6 PSNR, correlation and K-L Divergence values in Gutab et al.’s Technique

<table>
<thead>
<tr>
<th>Image name</th>
<th>Image size (KB)</th>
<th>Payload (bytes)</th>
<th>BPP</th>
<th>PSNR (dB)</th>
<th>Correlation</th>
<th>K-L Divergence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>768</td>
<td>14869</td>
<td>0.45</td>
<td>55.86</td>
<td>0.9999</td>
<td>46.36</td>
</tr>
<tr>
<td>Garden-Home</td>
<td>1214</td>
<td>20047</td>
<td>0.39</td>
<td>56.41</td>
<td>0.9999</td>
<td>4020.6</td>
</tr>
<tr>
<td>Road</td>
<td>1050</td>
<td>15039</td>
<td>0.34</td>
<td>57.24</td>
<td>0.9999</td>
<td>538.8</td>
</tr>
<tr>
<td>Player</td>
<td>1837</td>
<td>30079</td>
<td>0.38</td>
<td>56.61</td>
<td>0.9999</td>
<td>137.1</td>
</tr>
</tbody>
</table>

Fig.3.8 (a)-(b) represents the variation of PSNR with payload in Lena and Road images respectively. From the graphs we can observe that if we go on increasing the payload up to a certain maximum allowable value, the PSNR may not reach less than 40.
3.4 Comparison

The first technique discussed in this Chapter provides two levels of security; (i) at cryptography level and (ii) at steganography level. The steganography used is a message dependent and randomized embedding. Compared to the existing LSB substitution techniques ours is a stronger one because of message bit dependent randomized LSB substitution.

The second technique provides three levels of security; (i) at cryptography level, (ii) at steganography level, and (iii) at user defined key level. The indicator channel is not the same for
all the eight blocks. Compared to existing pixel indicator techniques for RGB images ours is a stronger one because of the use of user defined key for allocation of message blocks to different image blocks.

The first technique is developed for gray images, whereas the second technique is developed for RGB images. Both the techniques are stronger and unique in their own style.

3.5 Conclusion

The first technique in this Chapter is a new way of embedding called dynamic or message dependent embedding. The randomized distribution of message bits over the three LSB planes makes the technique a stronger one.

The second technique is known as indicator and user key based steganography for RGB images. It provides three levels of security. The embedding capacity is good. The indicator channel is computed freshely for each image block. The embedding into channel1 and/or channel2 is done by a difference calculation of the values of four data bits and the values of four LSBs of the channel1 and/or channel2. PSNR values are acceptable and no visual artifacts can be observed from the corresponding stego-images.