Visual Cryptography Scheme for Color Image Using Random Number with Enveloping by Digital Watermarking

Shyamalendu Kandar[1], Arnab Maiti[2], Bibhas Chandra Dhara[3]

[1,2] Computer Sc. & Engineering
Haldia Institute of Technology
Haldia, West Bengal, India

[3]Department of Information Technology
Jadavpur University
Kolkata, West Bengal, India

Abstract

Visual Cryptography is a special type of encryption technique to obscure image-based secret information which can be decrypted by Human Visual System (HVS). This cryptographic system encrypts the secret image by dividing it into n number of shares and decryption is done by superimposing a certain number of shares(k) or more. Simple visual cryptography is insecure because of the decryption process done by human visual system. The secret information can be retrieved by anyone if the person gets at least k number of shares. Watermarking is a technique to put a signature of the owner within the creation.
In this current work we have proposed Visual Cryptographic Scheme for color images where the divided shares are enveloped in other images using invisible digital watermarking. The shares are generated using Random Number.

Keywords: *Visual Cryptography, Digital Watermarking, Random Number.*

1. Introduction

Visual cryptography is a cryptographic technique where visual information (Image, text, etc) gets encrypted in such a way that the decryption can be performed by the human visual system without aid of computers [1].
Like other multimedia components, image is sensed by human. Pixel is the smallest unit constructing a digital image. Each pixel of a 32 bit digital color image are divided into four parts, namely Alpha, Red, Green and Blue; each with 8 bits. Alpha part represents degree of transparency.
A 32 bit sample pixel is represented in the following figure [2] [3].

$$\underbrace{11100111}_{\textbf{Alpha}}\quad\underbrace{11011001}_{\textbf{Red}}\quad\underbrace{11111101}_{\textbf{Green}}\quad\underbrace{00111110}_{\textbf{Blue}}$$

Fig 1: Structure of a 32 bit pixel

Human visual system acts as an OR function. Two transparent objects stacked together, produce transparent object. But changing any of them to non-transparent, final objects will be seen non-transparent. In k-n secret sharing visual cryptography scheme an image is divided into n number of shares such that minimum k number of shares is sufficient to reconstruct the image. The division is done by Random Number generator [4].
This type of visual cryptography technique is insecure as the reconstruction is done by simple OR operation.
To add more security to this scheme we have proposed a technique called digital enveloping. This is nothing but an extended invisible digital watermarking technique. Using this technique, the divided shares produced by k-n secret sharing visual cryptography are embedded into the envelope images by LSB replacement [5]. The color change of the envelope images are not sensed by human eye [6]. (More than 16.7 million i.e.2^{24} different colors are produced by RGB color model. But human eye can discriminate only a few of them.). This technique is known as invisible digital watermarking as human eye can not identify the change in the envelope image and the enveloped (Produced after LSB replacement) image [7].
In the decryption process k number of embedded envelope images are taken and LSB are retrieved from each of them followed by OR operation to generated the original image.
In this paper Section 2 describes the Overall process of Operation, Section 3 describes the process of k-n secret sharing Visual Cryptography scheme on the image, Section 4 describes the enveloping process using invisible digital watermarking, Section 5 describes decryption process, Section 6 describes the experimental result, and Section 7 draws the conclusion.

2. Overall Process

Step I: The source image is divided into n number of shares using k-n secret sharing visual cryptography scheme such that k number of shares is sufficient to reconstruct the encrypted image.
Step II: Each of the n shares generated in Step I is embedded into n number of different envelope images using LSB replacement.

Step III: k number of enveloped images generated in Step II are taken and LSB retrieving with OR operation, the original image is produced.

The process is described by Figure 2

3. k-n Secret Sharing Visual Cryptography Scheme

An image is taken as input. The number of shares the image would be divided (n) and number of shares to reconstruct the image (k) is also taken as input from user. The division is done by the following algorithm.
Step I: Take an image IMG as input and calculate its width (w) and height (h).
Step II: Take the number of shares (n) and minimum number of shares (k) to be taken to reconstruct the image where k must be less than or equal to n. Calculate RECONS = (n-k)+1.
Step III: Create a three dimensional array IMG_SHARE[n][w*h][32] to store the pixels of n number of shares. k-n secret sharing visual cryptographic division is done by the following process.

```
for i = 0 to (w*h-1)
{
  Scan each pixel value of IMG and convert it into 32 bit
binary string let PIX_ST.
      for j = 0 to 31
        {     if (PIX_ST.charAt(i) =1){
              call Random_Place (n, RECONS)
              }
              for k = 0 to (RECONS−1)
              {
                 Set IMG_SHARE [RAND[k]][i][j] = 1
              }
         }
}
```

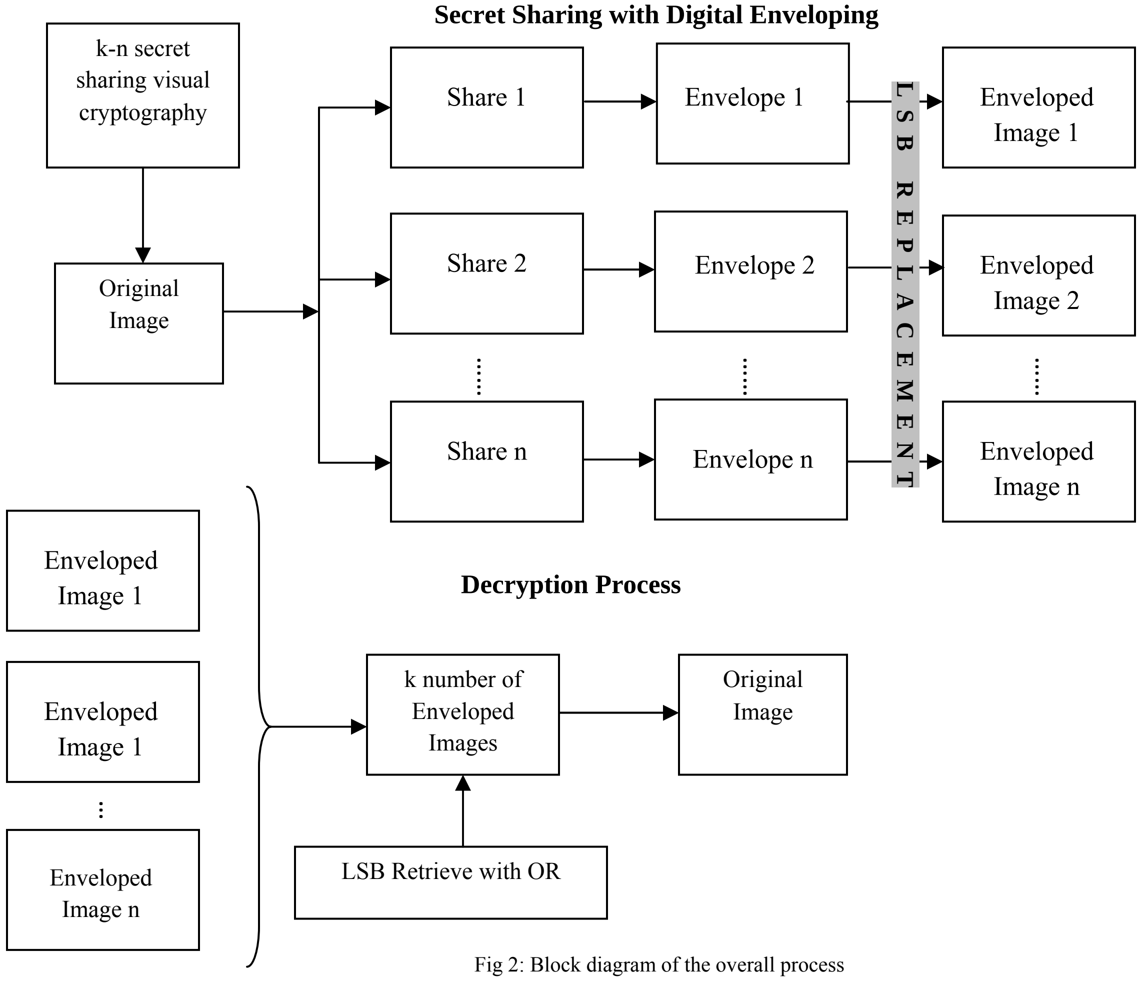


Fig 2: Block diagram of the overall process

Step IV: Create a one dimensional array IMG_CONS[n] to store constructed pixels of each n number of shares by the following process.

```
for k1 = 0 to (n-1)
{  for k2 = 0 to (w*h-1)
  {  String value= “”
     for k3 = 0 to 31   {
          value = value+IMG_SHARE [k1][k2][k3]
        }
      Construct alpha, red, green and blue part of each pixel
      by taking consecutive 8 bit substring starting from 0.
      Construct pixel from these part and store it into
      IMG_CONS[k1] [4].
      }
  Generate image from IMG_CONS [k1]^1 [8].
}
```

subroutine int Random_Place(n, RECONS)

```
{  Create an array RAND[RECONS] to store the
generated random number.
  for i = 0 to (recons-1)
    {
      Generate a random number within n, let rand_int. [9]
      if (rand_int is not in RAND [RECONS])
         RAND [i] = rand_int
    }
   return RAND [RECONS]
}
```

4. Enveloping Using Invisible Digital Watermarking

Using this step the divided shares of the original image are enveloped within other image. Least Significant Bit (LSB) replacement digital watermarking is used for this enveloping process. It is already discussed that a 32 bit digital image pixel is divided into four parts namely alpha, red, green and blue; each with 8 bits. Experiment shows that if the last two bits of each of these parts are changed; the changed color effect is not sensed by human eye[6]. This process is known as invisible digital watermarking [7]. For embedding 32 bits of a pixel of a divided share, 4 pixels of the envelope image is necessary. It means to envelope a share with resolution w X h; we need an envelope image with w X h X 4 pixels. Here we have taken each envelope of size 4w X h.

The following figure describes the replacement process. For replacing 8 bit alpha part, a pixel of the envelope is needed. In the same way red, green and blue part are enveloped in three other pixels of the envelope image.

The enveloping is done using the following algorithm

Step I: Take number of shares (n) as input.
for share = 0 to n-1 follow Step II to Step IV.

Step II: Take the name of the share, let SHARE_NO (NO is from 0 to n-1) and name of the envelope, let ENVELOPE_NO (NO is from 0 to n-1) as input. Let the width and height of each share are w and h. The width of the envelope must be 4 times than that of SHARE_NO.

Step III: Create an array ORG of size w*h*32 to store the binary pixel values of the SHARE_NO using the loop

```
for i = 0 to (w*h-1)
 {  Scan each pixel value of the image and convert it into
32 bit
     binary string let PIX
      for j = 0 to 31
       { ORG [i*32+j] = PIX.charAt(j)
       }
  }
```

Create an array ENV of size 4*w*h*32 to store the binary pixel values of the ENVELOPE_NO using the previous loop but from i = 0 to 4*w*h*32 −1.

Step IV: Take a marker M= −1. Using the following process the SHARE_NO is embedded within ENVELOPE_NO.

```
for i = 0 to 4*w*h −1
{
        ENV [i*32+6] = ORG [++M];
        ENV [i*32+7 ] = ORG [++M];
        ENV [i*32+14] = ORG [++M];
        ENV [i*32+15] = ORG [++M];
        ENV [i*32+22] = ORG [++M];
        ENV [i*32+23] = ORG [++M];
        ENV [i*32+30] = ORG [++M];
        ENV [i*32+31] = ORG [++M];
}
Construct alpha, red, green and blue part of each pixel by
taking consecutive 8 bit substring starting from 0.
Construct pixel from these part and store it into a one
dimensional array let IMG_CONS of size 4*w*h  [4].
      }
  Generate image from IMG_CONS [ ]^1.
```

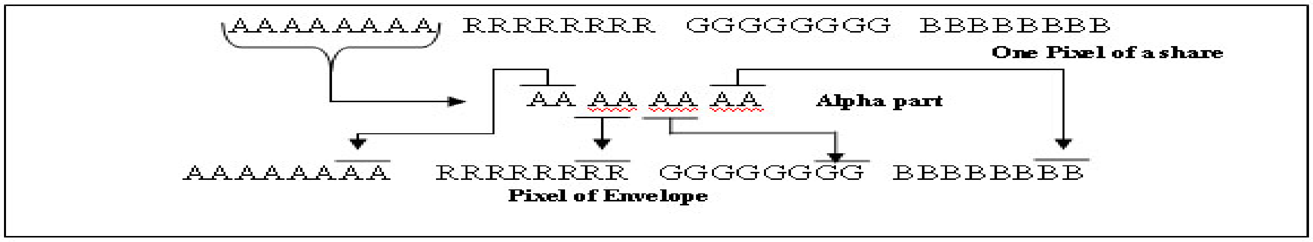

AAAAAAAA RRRRRRRR GGGGGGGG BBBBBBBB
One Pixel of a share
AA AA AA AA
Alpha part
AAAAAAAA RRRRRRRR GGGGGGGG BBBBBBBB
Pixel of Envelope

Fig 3: Enveloping Process

5. Decryption Process

In this step at least k numbers of enveloped images are taken as input. From each of these images for each pixel, the last two bits of alpha, red, green and blue are retrieved and OR operation is performed to generate the original image. It is already discussed that human visual system acts as an OR function. For computer generated process; OR function can be used for the case of stacking k number of enveloped images out of n.

The decryption process is performed by the following algorithm.

Step I: Input the number of enveloped images to be taken (k); height (h) and width (w) of each image.

Step II: Create a two dimensional array STORE[k][w*h*32] to store the pixel values of k number of enveloped images. Create a one dimensional array FINAL[(w/4)*h*32] to store the final pixel values of the image which will be produced by performing bitwise OR operation of the retrieved LSB of each enveloped images.

Step III:

```
for share_no = 0 to k-1
{
Take the name of the enveloped image to be taken and store the pixel values in STORE [share_no][w*h*32] using the following loop.
   for i = 0 to (w*h-1)
     {  Scan each pixel value of the Enveloped image and convert
           it into 32 bit binary string let PIX.
         for j = 0 to 31
          { STORE[share_no][i*32+j] = PIX.charAt(j)
          }
     }
  }
```

Step IV: Take a marker M= −1. Using the following process the last two bits of alpha, red, green and blue of each pixel of each k number of enveloped images are OR ed to produce the pixels of the original image.

```
for i = 0 to w*h
 {
Consider 8 integer values from C0 to C7 and set all of them to 0.
    for SH_NO = 0 to k-1
    {
      c0 = c0 | STORE [SH_NO] [i*32+6];    // | is bitwise OR
      c1 = c1 | STORE [SH_NO] [i*32+7];
      c2 = c2 | STORE [SH_NO] [i*32+14];
      c3 = c3 | STORE [SH_NO] [i*32+15];
      c4 = c4 | STORE [SH_NO] [i*32+22];
      c5 = c5 | STORE [SH_NO] [i*32+23];
      c6 = c6 | STORE [SH_NO] [i*32+30];
      c7 = c7 | STORE [SH_NO] [i*32+31];
    }

FINAL [++M] =c0;
FINAL [++M] = c1;
FINAL [++M] = c2;
FINAL [++M] = c3;
FINAL [++M] = c4;
FINAL [++M] = c5;
FINAL [++M] = c6;
FINAL [++M] = c7;
  }
```

Create a one dimensional array IMG_CONS[ ] of size (w/4)*h to store constructed pixels.
Construct alpha, red, green and blue part of each pixel by taking consecutive 8 bit substring from FINAL[ ] starting from 0.
Construct pixel from these part and store it into IMG_CONS[(w/4)*h]
Generate image from IMG_CONS[ ].

6. Experimental Result

Division using Visual Cryptography:
Source Image: Lena.png
Source image is

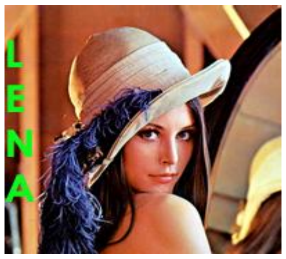
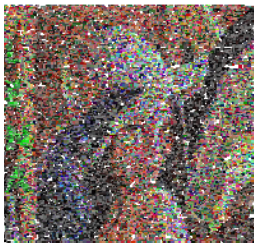
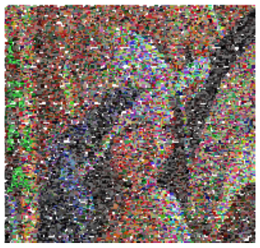


Fig 4: Source Image

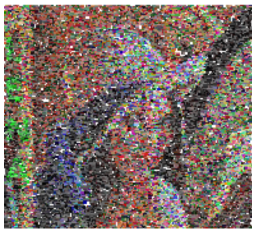
Number of Shares: 4
Numbers of shares to be taken: 3
Image shares produced after applying Visual Cryptography
are:



0img.png



1img.png


2img.png

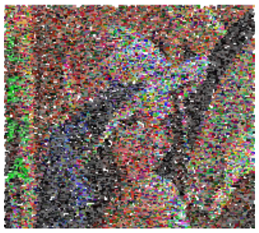
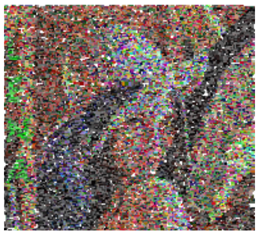

3img.png

Fig 5: Encrypted Shares

Enveloping using Watermarking:


0img.png

+

Envelope0.png

Final0.png

1img.png

+

Envelope1.png

Final1.png

2img.png

+

Envelope2.png

Final2.png

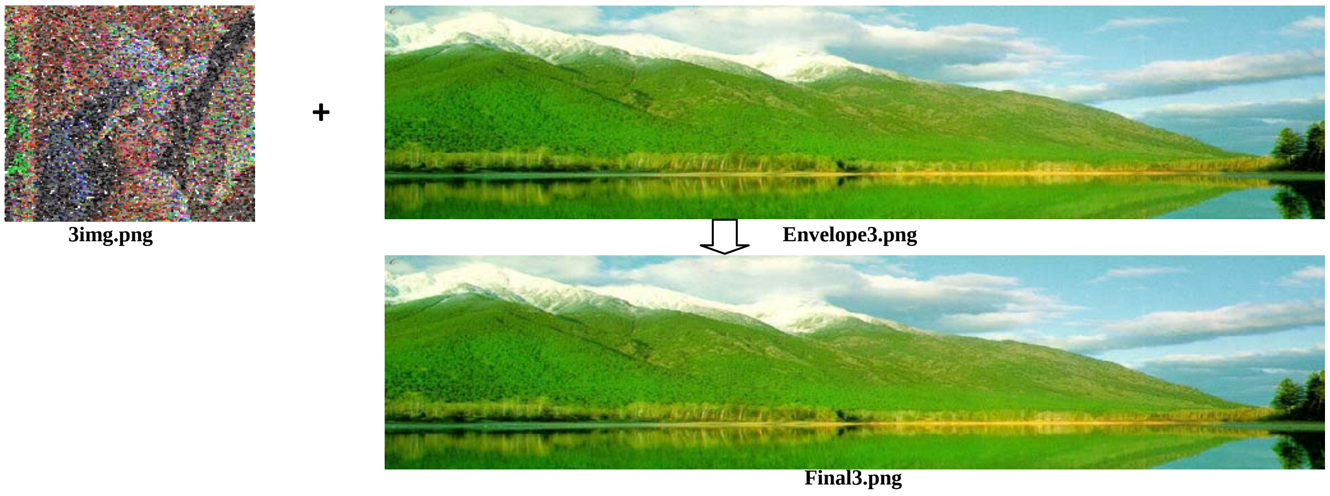


Fig 6: Enveloping shares using Digital Watermarking

Decryption Process:
Number of enveloped images taken: 3
Name of the images: Final0.png, Final2.png, Final3.png

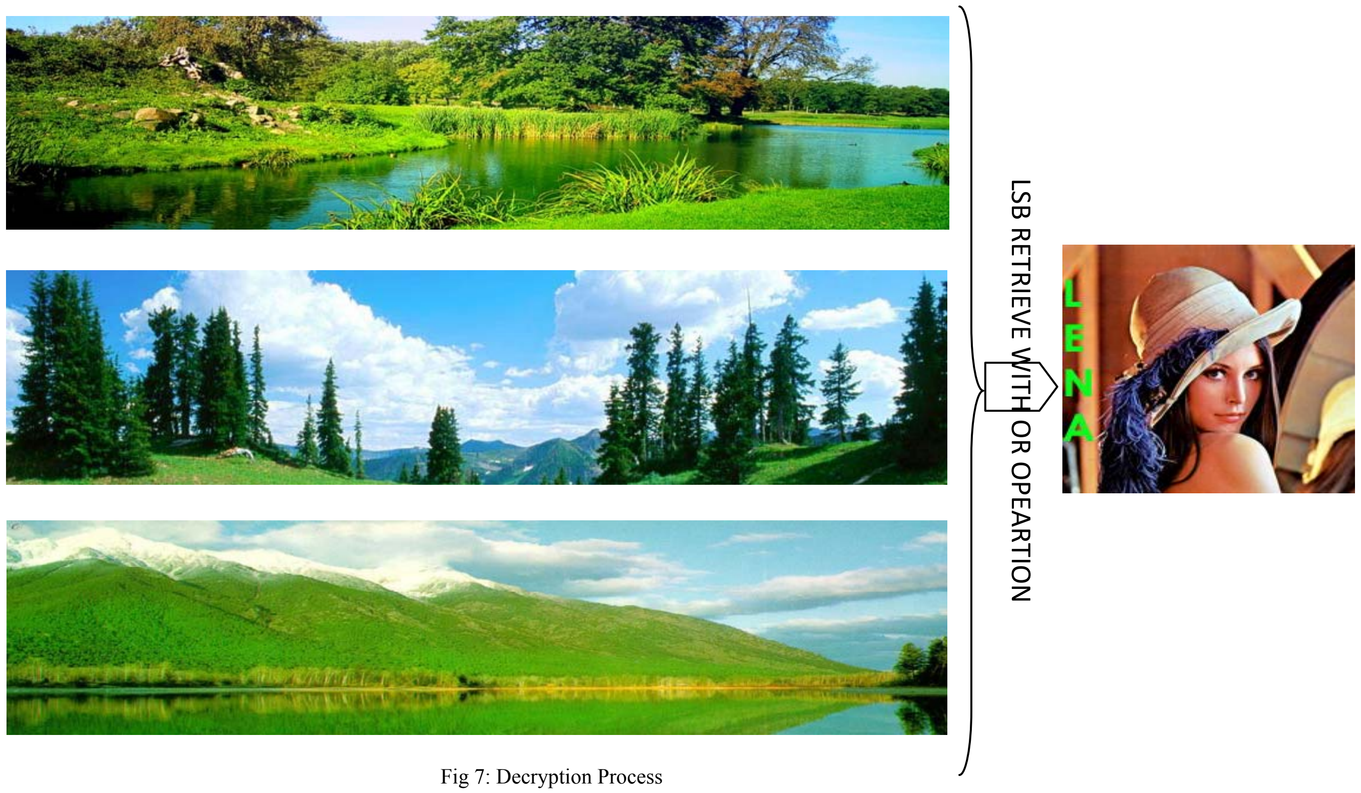


Fig 7: Decryption Process

7. Conclusion

Decryption part of visual cryptography is based on OR operation, so if a person gets sufficient k number of shares; the image can be easily decrypted. In this current work, with well known k-n secret sharing visual cryptography scheme an enveloping technique is proposed where the secret shares are enveloped within apparently innocent covers of digital pictures using LSB replacement digital watermarking. This adds security to visual cryptography technique from illicit attack as it befools the hackers' eye.
The division of an image into n number of shares is done by using random number generator, which is a new technique not available till date. This technique needs very less mathematical calculation compare with other existing techniques of visual cryptography on color images [10][11][12][13]. This technique only checks '1' at the bit position and divide that '1' into (n-k+1) shares using random numbers. A comparison is made with the proposed scheme with some other schemes to prove the novelty of the scheme.

Table 1: Margin specifications

Other Processes	Proposed Scheme
1. k-n secret sharing process is Complex[10][11][12].	1. k-n secret sharing process is simple as random number is used.
2. The shares are sent through different communication channels, which is a concern to security issue [10][11][12][13].	2. The shares are enveloped into apparently innocent cover of digital pictures and can be sent through same or different communication channels. Invisible digital watermarking befools the hacker.

References:

[1] M. Naor and A. Shamir, "Visual cryptography," Advances in Cryptology-Eurocrypt'94, 1995, pp. 1–12.
[2] P. Ranjan, "Principles of Multimedia", Tata McGraw Hill, 2006.
[3] John F Koegel Buford, Multimedia Systems, Addison Wesley, 2000.
[4] Kandar Shyamalendu, Maiti Arnab, "K-N Secret Sharing Visual Cryptography Scheme For Color Image Using Random Number" International Journal of Engineering Science and Technology, Vol 3, No. 3, 2011, pp. 1851-1857.
[5] Naskar P., Chaudhuri A, Chaudhuri Atal, Image Secret Sharing using a Novel Secret Sharing Technique with Steganography, IEEE CASCOM, Jadavpur University, 2010, pp 62-65.
[6] Hartung F., Kuttter M., "Multimedia Watermarking Techniques", IEEE, 1999.
[7] S. Craver, N. Memon, B. L. Yeo, and M. M. Yeung. Resolving Rightful Ownerships with Invisible Watermarking Techniques: Limitations, Attacks and Implications. IEEE Journal on Selected Areas in Communications, Vol16, No.4 May 1998, pp.573–586,.
[8] Schildt, H. The Complete Reference Java 2, Fifth Ed. TMH, Pp 799-839
[9] Krishmoorthy R, Prabhu S, Internet & Java Programming, New Age International, pp 234.
[10] F. Liu1, C.K. Wu1, X.J. Lin, Colour visual cryptography schemes, IET Information Security, July 2008.
[11] Kang InKoo el. at., Color Extended Visual Cryptography using Error Diffusion, IEEE 2010.
[12] SaiChandana B., Anuradha S., A New Visual Cryptography Scheme for Color Images, International Journal of Engineering Science and Technology, Vol 2 (6), 2010.
[13] Li Bai , A Reliable (k,n) Image Secret Sharing Scheme by, IEEE,2006.

Appendix:

[1] Java Language implementation is

```
int c=0;
int a=(Integer.parseInt(value.substring(0,8),2))&0xff;
int r=(Integer.parseInt(value.substring(8,16),2))&0xff;
int g=(Integer.parseInt(value.substring(16,24),2))&0xff;
int b=(Integer.parseInt(value.substring(24,32),2))&0xff;
img_cons[c++]=(a << 24) | (r<< 16) | (g << 8) | b;
```