A Technique for Secret Communication Using a New Block Cipher with Dynamic Steganography

Gandharba Swain[1] and Saroj Kumar Lenka[2]

[1]Research Scholar-CSE, SOA University,Bhubaneswar-751030, Odisha, India
[2]Professor- CSE, MITS University, Lakshmangarh-332311, Rajasthan, India
gswain1234@gmail.com, lenka.sarojkumar@gmail.com

Abstract

This paper presents a technique for secret communication using 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. Each pixel is 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, so it is called as dynamic steganography. The technique is experimented and results are discussed.

Keywords: *dynamic steganography, cryptography, 256 bit key cipher, hill cipher*

1. Introduction

Information and communication technology has grown rapidly. Internet is the most popular communication medium nowadays. But message transmission over the internet is facing some problems such as data security, copyright control, etc. So we need secret communication methods. Cryptography is a well known method in which the secret message is transformed to another form such that it does not make any sense to the intruder. Where as steganography is a method in which the secret message is hidden inside another file like an image such that the image looks innocent as if it is not carrying any message. If these two methods can be combined together to form a hybrid approach, then two levels of security can be achieved.

Steganography can be categorized into four categories. Those are: steganography in image, steganography in audio, steganography in video and steganography in text. The image steganography algorithms can be categorized into two categories, namely, spatial domain and frequency domain. When hiding information inside images usually least significant bit (LSB) method is used. In the LSB method the 8th bit of every byte of the carrier file is substituted by one bit of the secret information. This method works fine in the image carriers because if the least significant bit is changed from 0 to 1 or vice versa, there is hardly any change in the appearance of the color of that pixel. Instead of hiding a fixed number of bits in the LSBs of each pixel, one can also embed different number of bits in LSBs of different pixels based on pixel value range calculation [1].

In image steganography a pixel can carry secret bits by adding one from the gray value or by subtracting one from the gray value. This kind of ±1 steganography can hide a longer message than simple LSB embedding. Zhang and his colleagues proposed a double layered embedding method to further improve the embedding efficiency of ±1 steganography [2]. In [3] a text in image steganography is proposed. This technique presents a way for labeling different colors to identify dark areas of image and then embed the text in those areas. Instead of dark areas darker and brighter pixels can also be used to hide the secret data bits [4]. In [5] a text in image steganography is proposed by mapping the binary values of characters of the text message to various pixels of the image.

In general if the pixels are located in edge areas they can tolerate larger changes than those in smooth areas. The range of changeable pixel value in smooth areas is small, where as in edge areas it is large so that the stego image maintains a good perceptual quality. Wu and Tsai [6] proposed a pixel value differencing method, where a cover image is partitioned into non overlapping blocks of two consecutive pixels. A difference value is calculated from the value of the two pixels in each block. Secret data is embedded into a cover image by replacing the difference values of the two pixel blocks of the cover image with similar ones, in which bits of embedded data are included. Zhang and Wang found that pixel value differencing steganography is vulnerable to histogram based attacks and proposed a modification for enhanced security [7]. Chang and Tseng proposed two sided, three sided and four sided side match methods [8]. These methods find the correlation of a target pixel with its neighboring pixels in order to decide the number of bits to be embedded in that target pixel. A similar technique is also proposed in [9].

Juneja and Sandhu proposed a technique based upon LSB array, in which they have taken all the LSB bits of the different pixels as an array called LSB array, mapped the encrypted message block to this LSB array, where maximum matched, there steganographed [10]. In fact a number of images can be experimented; the image in which the percentage of match is maximum can be used. The rest of the paper is organized as follows. In section 2 the new block cipher, in section 3 the embedding technique, in section 4 the algorithms and in section 5 the experimental results are discussed. Finally the paper is concluded in section 6.

2. The New Block Cipher with 256 Bit Key

The hill cipher can operate over 26 alphabets a through z only [11]. If P is the plain text and K is the key, then the encryption and decryption is defined as follows.

Encryption: cipher text, $C = KP \bmod 26$
Decryption: plain text, $P = K^{-1}C \bmod 26$,

Where K^{-1} is the modular arithmetic inverse of K. Hill cipher has some limitations. It does not work over special characters and digits. It takes the blocks of smaller sizes, so key length is shorter. It is very simple, so vulnerable for exhaustive key search attack and known plain text attack. The proposed block cipher is an extension of hill cipher which addresses the above draw backs. It has a block length of 128 bits (16 characters) and key length of 256 bits (32 characters). The sub key generation, encryption and decryption processes of this block cipher are as discussed in the following sub-sections.

2.1. Sub Key Generation

The key which is 32 characters (256 bits) is represented in a 4 by 8 matrix, say K. The eight sub keys K_1, K_2, K_3, K_4, K_5, K_6, K_7 and K_8 are to be generated. The first sub key K_1 is a 4 by 4 matrix obtained by taking first 4 columns of K and fifth sub key K_5 is a 4 by 4 matrix obtained by taking the remaining 4 columns of K. The second sub key K_2 is obtained by interchanging the first and second column of K_1. The third sub key K_3 is obtained by interchanging the first and third column of K_1. The fourth sub key K_4 is obtained by interchanging the first and fourth column of K_1. The sixth sub key K_6 is obtained by interchanging the first and second column of K_5. The seventh sub key K_7 is obtained by interchanging the first and third column of K_5. The eighth sub key K_8 is obtained by interchanging the first and fourth column of K_5. In this way 8 sub keys namely K_1, K_2, K_3, K_4, K_5, K_6, K_7 and K_8 are generated.

2.2. Encryption

The message will be divided into different blocks, each 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 4 by 4 matrix, say P. The encryption of a message block is done by the following procedure.

```
For i= 1 to 8
    {
    $P_1 = (K_i*P) \bmod (127)$
    $P=P_1$
    }
For i= 1 to 8
    {
    $P_2 = stir (P)$
    $P_3 = XOR (K_i, P_2)$
    $P=P_3$
    }
    $C = P$
```

Where K* P is the matrix multiplication of the two 4 by 4 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.

2.3. Decryption

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

```
For i = 8 to 1
    {
    $C_1 = XOR (K_i, C )$
    $C_2 = stir (C_1)$
    $C = C_2$
    }
For i = 8 to 1
    {
    $C_3 = \{K_{mi}^{-1}*C\} \bmod (127)$
    $C=C_3$
    }
    $P=C$
```

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

2.4. Stir operation

If Matrix A=

$$\begin{pmatrix} 11001100, & 00101010, & 11100110, & 11001100 \\ 11011110, & 10101010, & 00100100, & 01010111 \\ 00011001, & 11101111, & 01111000, & 11000111 \\ 11111100, & 11011011, & 11011100, & 10011001 \end{pmatrix}$$

Then, B= Stir (A) =

$$\begin{pmatrix} 11001111, & 00101000, & 11100111, & 00101000 \\ 11100001, & 01101001, & 11100101, & 10100011 \\ 00110111, & 01101100, & 10111001, & 01110011 \\ 11111110, & 11010101, & 11101110, & 00110001 \end{pmatrix}$$

And Stir (Stir (A)) =

$$\begin{pmatrix} 11001100, & 00101010, & 11100110, & 11001100 \\ 11011110, & 10101010, & 00100100, & 01010111 \\ 00011001, & 11101111, & 01111000, & 11000111 \\ 11111100, & 11011011, & 11011100, & 10011001 \end{pmatrix}$$

This stir operation is defined by the following steps:

1. First two bits (i.e., 1st and 2nd bits) from each byte in a row of A are combined to form the first byte of B in that row.
2. Next two bits (i.e., 3rd and 4th bits) from each byte in a row of A are combined to form next byte of B in that row.
3. Next two bits (i.e., 5th and 6th bits) from each byte in a row of A are combined to form next byte of B in that row.
4. The last two bits (i.e., 7th and 8th bits) from each byte in a row of A are combined to form the last byte of B in that row.

This stir operation is reversible, i.e. Stir(Stir(A))=A.

2.5. 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 sub key generation, stir operation, modular arithmetic inverse and XOR operation together make the encryption and decryption methods strong. As a result the algorithm discussed in section 4 becomes very robust.

3. Embedding and Retrieving Techniques

3.1. 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. Some 3000 pixels at the beginning of the image 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.

Step1: Embed L in the reserved pixels (starting from 3001th pixel upto 3300th pixel) using two least significant bit locations. 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.
Step2: Do any of the four sub steps (a) through (d)
(a) If the two bits of cipher text embedded at the ith pixel of image are 00; then the next two bits of cipher text will be embedded in 7th and 8th bit locations of the (i+1)th pixel of the image.
(b) If the two bits of cipher text embedded at the ith pixel of image are 01; then the next two bits of cipher text will be embedded in 8th and 7th bit locations of the (i+1)th pixel of the image.
(c) If the two bits of cipher text embedded at the ith pixel of image are 10; then the next two bits of cipher text will be embedded in 6th and 7th bit locations of the (i+1)th pixel of the image.
(d) If the two bits of cipher text embedded at the ith pixel of image are 11; then the next two bits of cipher text will be embedded in 7th and 6th bit locations of the (i+1)th pixel of the image.
Step3: Set i= i+1 and L=L-2. If (L=0) goto step4, otherwise take the next two bits of cipher text and go to step2.
Step4: Stop.

Table 1. Embedding at Different Locations in Different Pixels

Cover image pixels (each 1 byte)	Operation	Location
pixel A	embedd 11	7th and 8th
pixel B	embedd 00	7th and 6th
pixel C	embedd 10	7th and 8th
pixel D	embedd 01	6th and 7th
pixel E	embedd 00	8th and 7th
pixel F	embedd 01	7th and 8th
pixel G	embedd 10	8th and 7th
pixel H	embedd 10	6th and 7th
pixel I	embedd 01	6th and 7th
pixel J	embedd 10	8th and 7th
pixel K	embedd 01	6th and 7th
pixel L	embedd 10	8th and 7th
pixel M	embedd 10	6th and 7th
pixel N	embedd 01	6th and 7th
pixel O	embedd 11	8th and 7th
pixel P	embedd 11	7th and 6th
etc...		

For example, let us assume that the data to be embedded is: **11001001 00011010 01100110 10011111.** Suppose the different pixels of image where cipher data 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 table1. 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.

3.2. The Retrieving Technique

Step1: Compute the length of the embedded message L by retrieving bits from the reserved pixels. Retrieve the first two cipher text bits from the 7th and 8th bit location of the 3301th pixel. Initialize the variable CIPHER with these two bits. Set L=L-2. Initialize i=3301.

Step2: 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.

Step3: Append these two retrieved bits to the variable CIPHER. Set L=L-2.

Step4: if (L >0) go to step2, otherwise go to step5.

Step5: Stop

4. The Proposed Algorithm

The algorithm at Sender is represented by the following steps.

1. Convert the carrier image to binary.
2. Divide the secret message into blocks, each block consisting of 16 characters (128 bits).
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, and then we get the stego binary image. Now convert this stego binary image to stego image and then send to receiver.

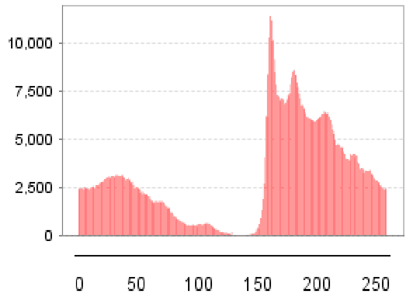
The algorithm at Receiver is as represented by the following steps.

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.
3. Keep them together, convert to text and divide into blocks, each 16 characters.
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.

4. Results and Discussion

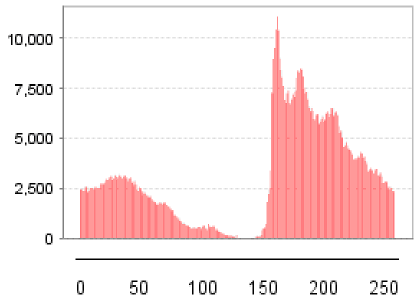
The technique is implemented using java programming language. The experimental results are observed. Five sample observations are shown below. Figure 1 (a) is the original Leena image, (b) is its histogram, (c) is the stego Leena image with 40 kilo bytes of cipher text embedded, and (d) is the histogram of stego Leena image. Figure 2 (a) is the original Fruits image, (b) is its histogram, (c) is the stego Fruits image with 40 kilo bytes of cipher text embedded, and (d) is the histogram of stego Fruits image. Figure 3 (a) is the original Player image, (b) is its histogram, (c) is the stego Player image with 50 kilo bytes of cipher text embedded, and (d) is the histogram of stego player image. Figure 4 (a) is the original Tree image, (b) is its histogram, (c) is the stego Tree image with 60 kilo bytes of cipher text embedded, and (d) is the histogram of stego Tree image. Figure 5 (a) is the original Lady-Man image, (b) is its histogram, (c) is the stego Lady-Man image with 50 kilo bytes of cipher text embedded, and (d) is the histogram of Lady-Man image.

(a)



(b)

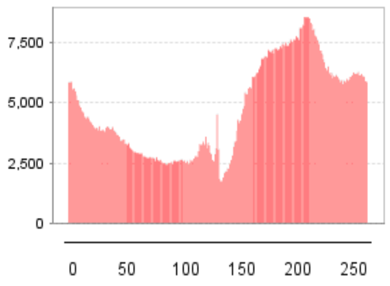
(c)



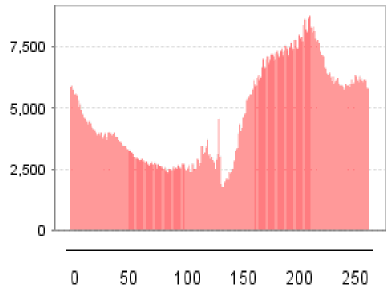
(d)

Figure 1. (a) original Leena image, (b) is its histogram, (c) is the stego Leena image with 40 kilo bytes of cipher text embedded, and (d) is the histogram of stego Leena image.

(a)

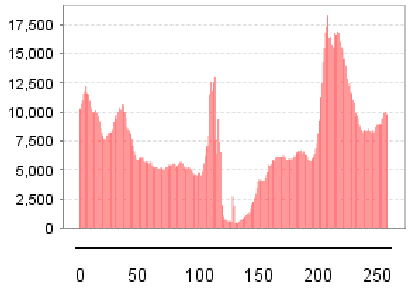


(b)

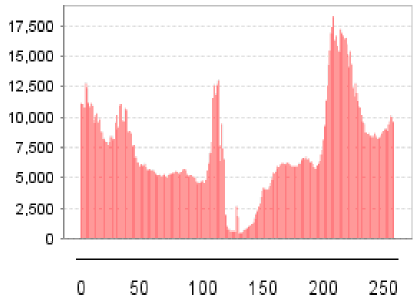


(c) (d)

Figure 2. (a) original Fruits image, (b) is its histogram, (c) is the stego Fruits image with 40 kilo bytes of cipher text embedded, and (d) is the histogram of Fruits image.

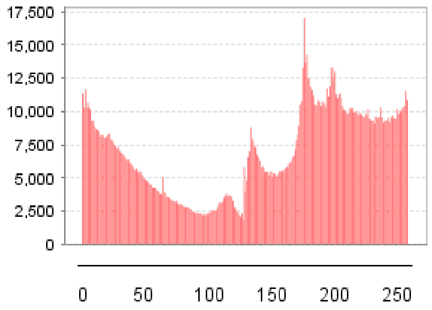


(a) (b)



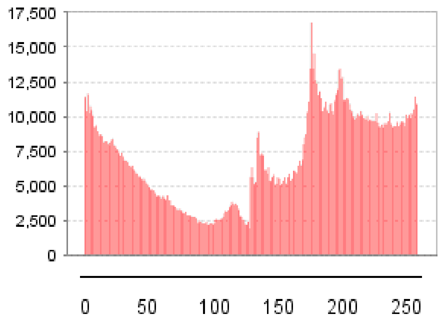
(c) (d)

Figure 3. (a) original Player image, (b) is its histogram, (c) is the stego Player image with 50 kilo bytes of cipher text embedded, and (d) is the histogram of Player image.



(a) (b)

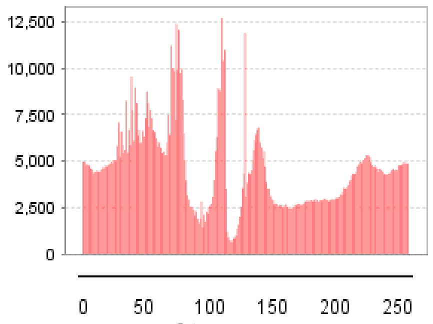
(c)



(d)

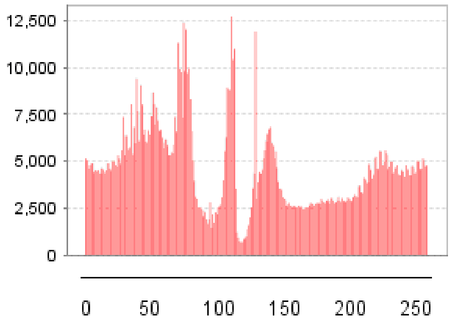
Figure 4. (a) original Tree image, (b) is its histogram, (c) is the stego Tree image with 60 kilo bytes of cipher text embedded, and (d) is the histogram of Tree image.

(a)



(b)

(c)



(d)

Figure 5. (a) original Lady-Man image, (b) is its histogram, (c) is the stego Lady-Man image with 50 kilo bytes of cipher text embedded, and (d) is the histogram of Lady-Man image.

The distortion in the stego image can be computed by peak signal to noise ratio (PSNR). For all the stego images the measured PSNR value is as given in table 2. It is measured in decibels (dB). PSNR values falling below 30 dB indicate a fairly low quality, i.e., distortion caused by embedding can be obvious; however, a high quality stego image should strive for 40 dB and above [12]. The PSNR is as defined below.

$$\text{PSNR}= 10 \log_{10}\left(\frac{C_{\max}^{2}}{\text{MSE}}\right),$$

Where C_{max} = the maximum pixel value, 255 for 8-bit images. MSE called the mean square error is as defined below.

$$\text{MSE} = \frac{1}{MN} \sum_{x=1}^{M} \sum_{y=1}^{N} \left(S_{xy} - C_{xy} \right),$$

Where x and y are the image coordinates, M and N are the dimensions of the image, C_{xy} is the cover image and S_{xy} is the generated stego image.

The capacity i.e., the amount of information hidden in the image is calculated in bits per pixel (bpp). This is defined as the ratio of message length in bits to the image length in pixels. The bpp for all the stego images is as given in Table 2.

The performance of various steganographic methods can be rated by the three parameters: security, capacity and imperceptibility. Steganography may suffer from many active or passive attacks, thus a secured steganographic approach should survive from passive or active attacks. Moreover if the existence of the secret message can only be estimated with a probability not higher than random guessing in the presence of some steganalytic systems, steganography may be considered secure under such steganalytic systems. Otherwise we may claim it to be insecure. Capacity means the amount of message that can be hidden. To be useful in conveying secret message, the hiding capacity should be as high as possible. It can be given in absolute measurement such as the size of secret message, or in relative value such as bits per pixel, or the ratio of the secret message to the cover medium. Stego images should not have severe visual artifacts. Under the same level of security and capacity, the higher the imperceptibility of the stego image, the better is the steganography. If the resultant stego image appears innocuous enough, one can believe this requirement to be satisfied well.

Table 2. Observed PSNR and bpp Values

Image Name	Image Size (kilo bytes)	Amount of cipher text embedded (kilo bytes)	bpp (bits per pixel)	PSNR (decibels)
Leena	768	40	0.42	50.93
Fruits	1186	40	0.27	52.86
Player	1837	50	0.22	53.78
Tree	1749	60	0.28	52.78
Lady-Man	1149	50	0.35	51.70

This proposed algorithm is highly secure as it uses dynamic embedding i.e. message dependent embedding. If you see the histograms in the different figures, there is no difference between the histograms of original images and the histograms 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. By observing the stego images in figures 1 through 5, one can not find any visual artifacts, thus the technique is highly imperceptible.

There are some basic notes that should be observed by a steganographer [12]. Those are: (i) In order to eliminate the attack of comparing the original image with the stego image, we can freshly create an image and destroy it after generating the stego image. The images available in world wide web should not be used. (ii) In order to avoid any human visual perceptual attack, the generated stego image must not have visual artifacts. (iii) Smooth homogeneous areas must be avoided; however chaotic areas with naturally noisy back grounds and salient rigid edges can be targeted. (iv) The secret data must be a composite of balanced bit values, since in general; the expected probabilities of bit 0 and 1 for a typical cover image are the same. Hopefully encryption may provide such a balance.

6. Conclusion

This technique is a unique and stronger approach of doing steganography with images. It provides two levels of security, one at the cryptography level and other at the steganography level. The cryptography algorithm is a block cipher with 256 bit key, thus a stronger one. The steganography follows a cipher text dependent embedding. After the cipher text is embedded, the degradation in image quality is not noticeable by human visual system. The amount of message that can be embedded is also very good. The technique is not susceptible to histogram based attacks.

References

[1] Y. K. Jain and R. R. Ahirwal, "A Novel Image Steganography Method with Adaptive Number of Least Significant Bits Modification Based on Private Stego Keys", International Journal of Computer Science and Security, vol. 4, no. 1, pp. 40-49, **(2010)**.

[2] W. Zhang, X. Zhang and S. Wang, "A Double Layered Plus-Minus One Data Embedding Scheme", IEEE Signal Processing Letters, vol. 14, no. 11, pp. 848-851, **(2007)**.

[3] H. Motameni, M. Norouzi, M. Jahandar and A. Hatami, "Labeling Method in Steganography", Proceedings of World Academy of Science, Engineering and Technology, **(2007)** October, vol. 24, pp. 349-354.

[4] G. Swain and S. K. Lenka, "A Hybrid Approach to Steganography Embedding at Darkest and Brightest Pixels", Proceedings of the International Conference on Communication and Computational Intelligence, **(2010)**, pp.529-534.

[5] M. A. F. Al-Husainy, "Image Steganography by mapping Pixels to Letters", Journal of Computer Science, vol.5, no.1, pp. 33-38, **(2009)**.

[6] D. C. Wu and W. H. Tsai, "A Steganograhic Method for Images by Pixel Value Differencing", Pattern Recognition Letters, vol. 24, no. 9-10, pp.1613-1626, **(2003)**.

[7] X. Zhang and S. Wang, "Vulnerability of Pixel Value Differencing Steganography to Histogram Analysis and Modification for Enhanced Security", Pattern Recognition Letters, vol. 25, pp. 331-339, **(2004)**.

[8] C. C. Chang and H. W. Tseng, "A Steganographic Method for Digital Images Using Side Match", Pattern Recognition Letters, vol. 25, no. 12, pp. 1431-1437, **(2004)**.

[9] K. J. Kim, K. H. Jung and K. Y Yoo, "Image Steganographic Method with Variable Embedding Length", Proceedings of International Symposium on Ubiquitous Computing, **(2008)**, pp. 210-213.

[10] M. Juneja and P. S. Sandhu, "Designing of Robust Steganography Technique Based on LSB Insertion and Encryption", Proceedings of International Conference on Advances in Recent Technologies in Communication and Computing, **(2009)**, pp. 302-305.

[11] W. Stallings, "Cryptography and Network Security Principles and Practices", Low Price Edition, **(2003)**.

[12] A. Cheddad, J. Condell, K. Curran and P. M. Kevitt, "Digital Image Steganography: Survey and Analysis of Current Methods", Signal Processing, vol. 90, pp.727-752, **(2010)**.

Authors

Mr. Gandharba Swain is working as an Associate Professor in the Department of Information Technology, GMR Institute of Technology, Rajam, Andhra Pradesh, India. He received B.Sc(Hons) degree from Berhampur University in 1995, MCA degree from VSS University of Technology (formerly UCE), Burla, in 1999, M.Tech (CSE) degree from NIT, Rourkela, in 2004. He has more than 13 years of teaching experience. He has authored one text book, published several research articles in national and international journals and conferences.

Dr. Saroj Kumar Lenka is working as Professor in the Department of Computer Science and Engineering, Mody Institute of Technology and Science-Deemed University, Rajasthan, India. He has received B.Sc. degree from Berhampur University in 1990, B.E in Computer Science and Engineering from Utkal University in 1994, M.Sc. in Information Technology in 2003 from Allahabad Agricultural Institute Deemed University, M.Tech. in Computer Science in 2005 from Kalinga University and Ph.D. in Computer Science in 2008 from Berhampur University, Orissa. He has more than 18 years experience in teaching and published many research articles in national and international journals and conferences.