International Journal of Engineering Research and General Science Volume 3, Issue 2, Part 2, March-April, 2015
ISSN 2091-2730

Secured Reversible Data Hiding in Encrypted Images by Using Multi Level Data Encryption

Sabeena Shukkoor, Yasim Khan M

Department of Electronics and Communication
College of Engineering Poonjar , CUSAT
Kottayam, India
sabeenashukkoor@gmail.com

Abstract— Now a days, more attention is paid to reversible data hiding (RDH) in encrypted images, since it maintains the important property that the original cover can be losslessly recovered after embedded data is extracted while protecting the image content's properly. All the existing methods embed data by reversibly vacating room from the encrypted images, which may be subject to some errors on data extraction and/or image recovery. This paper presents a novel method by reserving room before encryption with a traditional RDH algorithm which makes much easier for the data hider to reversibly embed data in the encrypted image. This method achieve the property of real reversibility, which means data extraction and image recovery are free of errors. The proposed method also uses an additional data encryption key for improving the security. Experiments show that this novel method can embed more than 10 times as large payloads for the same image quality as the previous methods.

Keywords—Image encryption, Reversible data hiding, Privacy protection, Histogram shift, Additional data encryption, Reserving room before encryption, Peak signal to noise ratio.

INTRODUCTION

Reversible data hiding (RDH) and Non reversible data hiding are the two different data hiding techniques used. In RDH, the original image is recovered with out any loss after extracting the embedded data where as in Non-reversible data hiding once the image is distorted it cannot be reconstructed back. RDH is used where the image and the data have equal importantce. This technique is widely used in medical imaging, military purpose, law forensics etc, where distortion of the original cover is not allowed. The term data hiding comes from the word Stegnography, which is the art of hiding secret data into a carrier to convey secret messages confidentially. Digital images are used as carriers. The original image is the cover image and the image up on which data is embedded is known as stego image. Some distortions may occur in the stego image due to data embedding and these distortions are known as embedding distortions. A good embedding algorithm creates only less embedding distortions.

In Providing secrecy for images, encryption [1] is an effective and popular means since it converts the original and meaningful content to incomprehensible one. Some promising applications can be generated if RDH can be applied to encrypted images. If the data base of a medical image is stored in a data center, these datas can be embedded into the encrypted version of a medical image via RDH technique by using a server. The server [2] can manage the image or verify its integrity by using the notations without having the knowledge of the original content. Here actually patient's privacy is being protected. At the same time, a doctor can decrypt and restore the image for further diagnosing by using the same encryption key.

Many RDH techniques are available now a days based on lossless compression such as histogram modification [3], difference expansion (DE) [4] etc. Among these, histogram based techniques are commonly used. The histogram is modified using .Histogram based methods so that the secret data can be embedded in to the modified histogram. In [5], Ni et al proposed histogram based method in which data is embedded in to the image based on zero/peak pixel value. This method is simple and utilizes short execution time. This technique also have high stego image quality, but embedding capacity is low. This algorithm does not work if the image is having a flat histogram. It also has overflow or underflow problem.

The objective of the proposed method is to develop an RDH scheme with increased security. In order to increase the security an additional data encryption is introduced in the proposed technique.

The rest of this paper is organized as follows. The previous methods of reversible data hiding is explained in section 2, proposed method of data hiding is described in section 3, experimental and theoretical analysis is presented in section 4, and finally conclusions in section 5.

PREVIOUS METHOD

The methods proposed in [6]-[8] can be summarized into the framework, vacating room after encryption (VRAE), In this framework, a content owner encrypts the original image using a standard cipher with an encryption key. After the encrypted image is produced, the content owner hands over it to a data hider and the data hider can embed some additional data into the encrypted image by losslessly vacating some room according to a data hiding key. Then a receiver, sometimes the content owner himself or an authorized third party can extract the embedded data with the data hiding key and then recover the original image from the encrypted version according to the encryption key. This method is illustrated in Figure 1(a).

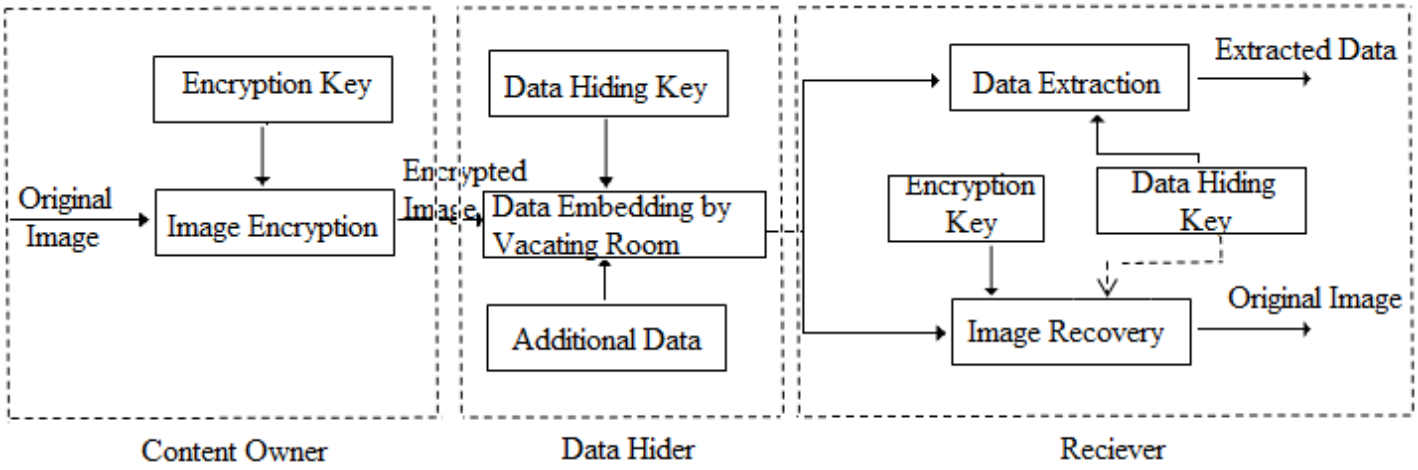

Figure 1(a): Framework of Vacating Room After Encryption

The methods in [6]–[8], encrypts every bit-planes with a stream Cipher, the encrypted 8-bit gray-scale images are generated. The method in [6] segments the encrypted image into a number of non-overlapping blocks sized by $a \times a$; and each block is used to carry one additional bit. The pixels are pseudo-randomly divided into two sets S_1 and S_2 in each block according to a data hiding key. If the extra bit to be embedded is 0, flip 3 LSBs of each encrypted pixel in S_1, otherwise flip the 3 encrypted LSBs of pixels in S_2. For data extraction and image recovery, the receiver flips all 3 LSBs of pixels in S_2 to form a newly decrypted block, and flips all the 3 LSBs of pixels in S_1 to form another new block; one of them will be decrypted to the original block. The original block is assumed to be smoother than interfered block and the embedded bit can be extracted. There is a risk in the bit extraction and image recovery when divided block is relatively small (e.g., $a = 8$) or has much fine-detailed textures.

Hong *et al.* [7] reduced the error rate of Zhang's method [6] by fully exploiting the pixels in calculating the smoothness of each block. The extraction and recovery of blocks are performed according to the descending order of the absolute smoothness difference between two candidate blocks and recovered blocks can further be used to evaluate the smoothness of unrecovered blocks.

PROPOSED METHOD

In the proposed method the order of creating space for data embedding and encryption is reversed. i.e, space to embed data is reserved prior to image encryption and using some RDH technique data is embedded into these specified areas. This method is known as "reserving room before encryption (RRBE)" [9] scheme.

In the proposed framework, reserving room before encryption (RRBE), the content owner first reserve some space on original image and then convert the image into its encrypted version with the encryption key. The data embedding process in encrypted images is reversible, so the data hider only needs to accommodate data into the extra space created. The data extraction and image recovery are free of errors [15], which identical to that of Framework VRAE. The ideal operator for reserving room before encryption are standard RDH algorithms and can be easily applied to Framework RRBE to achieve better performance when compared with Framework VRAE. Due to this new framework, the redundant image content is losslessly compressed and then encrypts it with respect to protecting privacy and is illustrated in Figure 1(b).

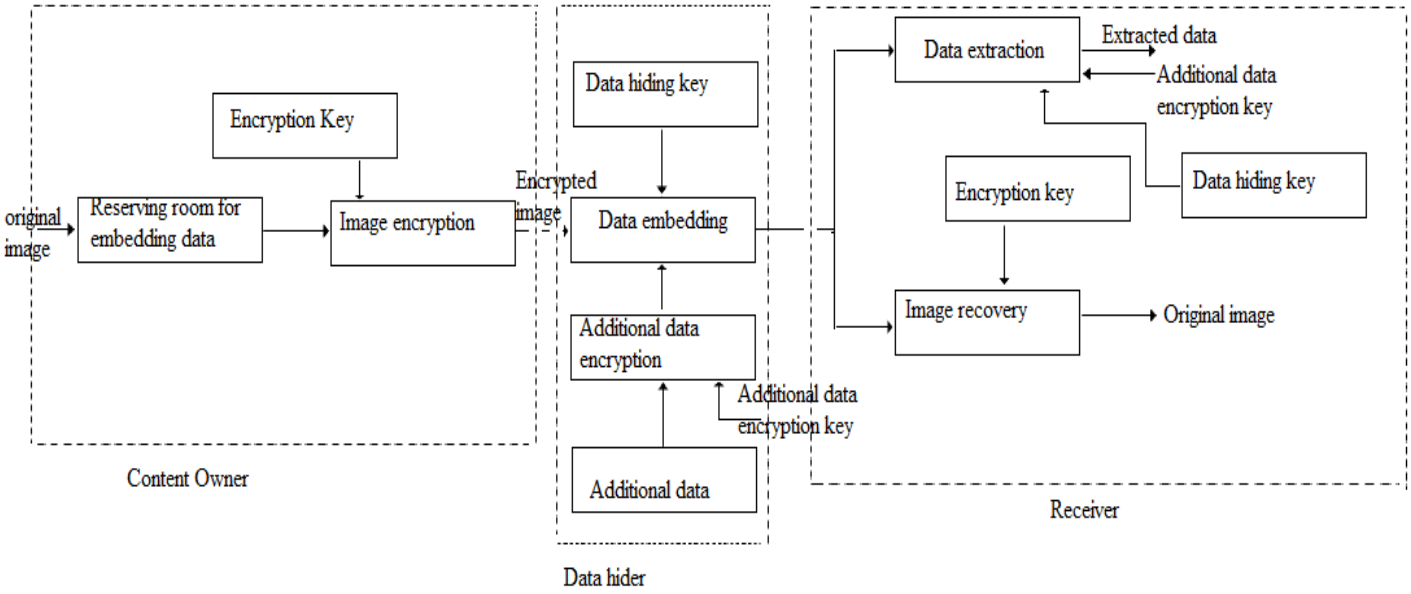

Figure 1(b): Reserving Room Before Encryption with additional data encryption

The proposed method is of four stages:

- Generation of encrypted image
- Additional data Encryption
- Encrypted Data hiding in encrypted image
- Data extraction and Image recovery

A. Generation of Encrypted Images

An encrypted image can be generated using 3 steps : Image partition, Self reversible embedding and Image encryption.

1. Image Partition

The goal of this step is to divide the image into two parts say, A and B using a smoothness function so that a smoother area B is constructed on which standard RDH algorithm [10], [11] can achieve better performance. Consider an original image I of size $M \times N$ and pixel $C_{i,j} \in [0,255], 1 \leq i \leq M, 1 \leq j \leq N$. At first the size of to be embedded message is calculated and denoted as I. The original image is divided into several overlapping blocks along the rows, whose number is determined by I. Each block having m blocks, where $m = \lceil l/N \rceil$ and the number of blocks can be calculated by n= M-m+1.Each block is overlapped by the previous block. A function f is defined to measure the smoothness of each block.

$$f = \sum_{u=2}^{m} \sum_{v=2}^{N-1} \left| C_{u,v} - \frac{C_{u-1,v} + C_{u+1,v} + C_{u,v-1} + C_{u,v+1}}{4} \right| \quad (1)$$

Block containing higher value of f will be the more complex textured area. The content owner selects the particular block with the highest f to be **A** , and puts in front of the image which is concatenated by rest part which is considered as B with fewer textured area, shown in Figure 2.

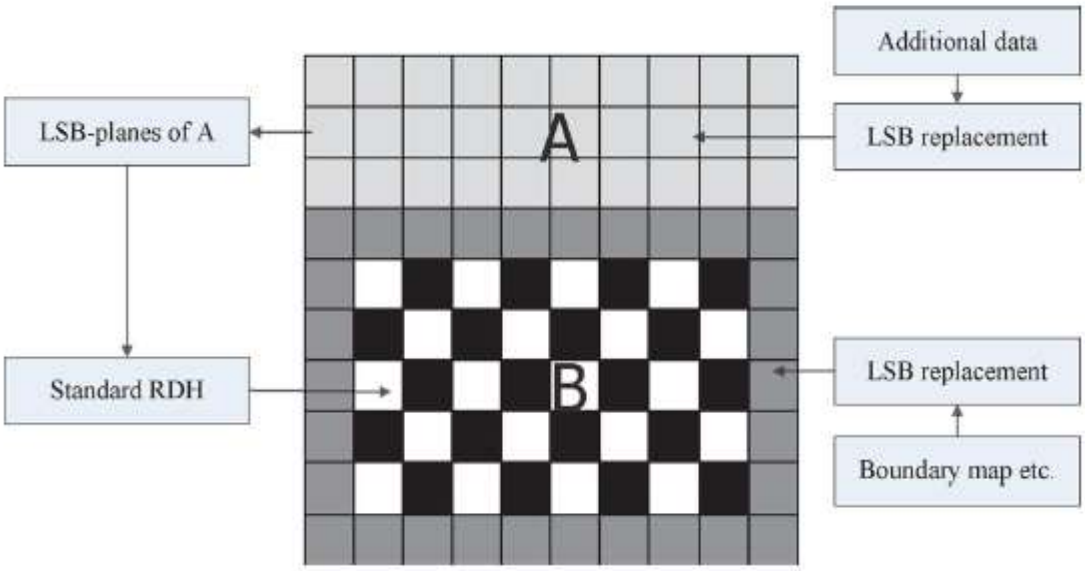

Figure 2: Illustration of image partition and embedding process

2. Self-Reversible Embedding

The main goal of this step is to embed the LSB planes of A into B using any of the RDH algorithm. Interpolation method is one of the commonly used RDH algorithm used in this paper. Pixels in part B are divided in to 2 sets: white and black pixels. White pixels are those whose indices satisfy (i + j) mod 2 = 0 and black pixels are (i +j) mod 2 = 1. Then each value of white pixel is estimated by interpolation value obtained with four black pixels surrounding it using the equation given below:

$$B'_{i,j} = w_1 B_{i-1,j} + w_2 B_{i+1,j} + w_3 B_{i,j-1} + w_4 B_{i,j+1} \quad (2)$$

where the weight w_i ,$1 \le i \le 4$, is determined in the same way proposed in [10]. The estimating error is calculated by, $e_{i,j} = B_{i,j} - B'_{i,j}$and then the data are embedded into estimating error sequence using histogram shift. After calculating all the values of white pixels the estimating error of black pixels are also calculated using modified white pixels and data are embedded into estimated error sequence of white pixels. If more data have to be embedded multilayer embedding can be used. Inorder to embed messages for every single layered embedding two estimating error sequence are required. By using bidirectional histogram shift, the messages are embedded into the error sequence, ie, the estimated error histogram is divided into two, right and left. The highest point in each part is denoted as RM and LM and the zero point in each part denoted as RN and LN. For ideal images RM = 0 and LN = -1. To embed messages into RM all values between RM+1 and RN-1 are shifted towards right by one. The embedding process in the left part is similar, but the shifting direction is left.

The overflow/underflow problem in this method is eliminated by embedding data only on those pixels whose values ranges between 1 and 254. Problem occurs when non-boundary pixels such as 1 is changed to 0 or 254 to 255.These newly generated boundary pixels are known as pseudo boundary pixels. So a boundary map is maintained to identify whether the boundary pixels are pseudo or natural. A binary sequence bit "0" is used to denote natural boundary pixel and "1" for pseudo boundary pixel. The marginal area can be selected to embed boundary map. The parameters such as RN, LN, RM, LM, payload, start row, end row of A in original image are embedded into marginal area.

3. Image Encryption

Image Encryption can be done using any encryption algorithm. After rearranged self-embedded image denoted by X , is generated, we can encrypt X to construct the encrypted image, denoted by E .By using a stream cipher, the encryption version of X can be obtained. For example, a gray value $X_{i,j}$ ranging from 0 to 255 can be represented by 8 bits, $X_{i,j}(0)\ X_{i,j}(1)$,…..,$X_{i,j}(7)$ such that,

$$X_{i,j}(k) = \left\lfloor \frac{X_{i,j}}{2^k} \right\rfloor mod2, \quad k = 0,1,....7 \tag{3}$$

The encrypted bits $E_{i,j}(k)$ can be calculated through exclusive-or operation

$$E_{i,j}(k) = X_{i,j}(k) \oplus r_{i,j}(k), \tag{4}$$

where $r_{i,j}(k)$ is generated via a standard stream cipher determined by the encryption key. Finally, we embed 10 bits information into LSBs of first 10 pixels in encrypted version of A to tell the data hider the number of rows and the number of bit-planes he can embed information into. It is important that after image encryption, the data hider or a third party can not access the content of original image without the encryption key, thus privacy of the content owner being protected.

B. Additional Data Encryption

After encryption the encrypted image X is sent to the data hider. The data hider does not have any access to the real image. The main goal of additional data encryption is to improve the security of the secret/additional data, before embedding it to the encrypted image. Additional data encryption is done with the help of an additional data encryption key. In the reciever section, only the person having this additional data encryption key can retrieve the secret data. This provides higher security to the secret data. Anyone who does not possess the additional data encryption key could not extract the additional data. The simple XOR cipher is the encryption algorithm used for performing the additional data encryption and it operates according to the principles given below:

$A \oplus 0 = A,$
$A \oplus A = 0,$
$(A \oplus B) \oplus C = A \oplus (B \oplus C),$
$(B \oplus A) \oplus A = B \oplus 0 = B,$

where $\oplus$ denotes the exclusive disjunction (XOR) operation. This operation is sometimes called modulus 2 addition (or subtraction, which is identical).With this logic, a string of text can be encrypted by applying the bitwise XOR operator to every character using a given key. To decrypt the output, merely reapplying the XOR function with the key will remove the cipher.

The XOR operator is extremely common as a component in more complex ciphers. Its primary merit is that it is simple to implement, and that the XOR operation is computationally inexpensive. If the key is random and is at least as long as the message, the XOR cipher is much more secure than when there is key repetition within a message[18][19]. When the keystream is generated by a pseudo-random number generator, the result is a stream cipher. With a key that is truly random, the result is a one-time pad, which is unbreakable even in theory.

C. Encrypted Data Hiding in Encrypted Image

After encryption the encrypted data is embedded into the encrypted image. In the encrypted image, the region up on which data to be embedded are already identified and taken into front which is denoted as A_E . After knowing how many bit-planes and rows of pixels he can modify, the data hider simply adopts LSB replacement to substitute the available bit-planes with additional encrypted data. Anyone who does not possess the data hiding key could not extract the additional data.

D. Data Extraction and Image Recovery

Data extraction is the process reverse to data embedding and image decryption is the process reverse to image encryption. If the receiver having the keys i.e. image encryption key and additional data encryption key and data hiding key, he can decrypt the image and extract the data[10][11]. If he has only encryption key he can only decrypt the image, he can't extract the data. If he is having only data hiding key he can extract the data, but cannot decrypt the image. If he is having only additional data encryption key he can only extract the encrypted data but cannot decrypt the data without data hiding key (since using multi data encryption) and also cannot decrypt the image.

EXPERIMENTS AND RESULTS

In this section, the simulation results and testing of performance of the proposed scheme by the additional data encryption and additional data encryption key analyses are presented. All the experiments have been performed on a personal computer with a 2.4 GHz Intel Core2 i3 processor, 2G memory and 250 GB hard disk with a Windows 7 operating system. The proposed method has been tested on standard publically available images such as Lena, Airplane, Barbara, Peppers and Boat and each image is of size 512 x 512.

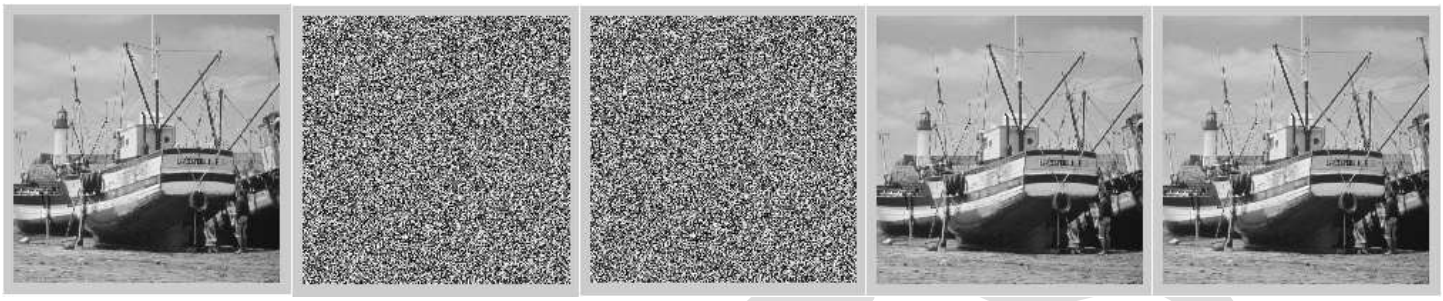

Figure 3: (a) Original Image, (b) Encrypted Image, (c) Encrypted Data, (d) Decrypted Image, (e) Recovered Image

Implementation Issues

The peak signal-to-noise ratio (PSNR) is the objective criteria to find out the quality of the images after decryption. PSNR is the ratio between a signal's maximum power and the power of the signal's noise. Each picture element (pixel) may get changed when an image is modified. Logically, a higher value of PSNR is good because it means that the signal to noise ratio is higher. Signals can have a wide dynamic range, so PSNR is usually expressed in decibels, which is a logarithmic scale. PSNR values of the test images [12] are recorded and evaluated inorder to check the quality of images and to check the efficiency of the proposed algorithm. To achieve high PSNR the following measures have to be taken.

a) Boundary Map

Boundary map is used to distinguish between natural and pseudo boundary pixels. Its size is a criteria to the applicability of the proposed approach. In most cases no boundary map is needed. Table 1 shows the boundary map size of the five standard images. The marginal area of the image must be large enough to record the boundary map. From Table 1 we can find that the images Lena, Airplane, Barbara and Boat are not using any boundary map for various embedding rates. But in the case of pepper image, up to an embedding rate of 0.4 bpp it can hold boundary map. Beyond that embedding rate for pepper, it does not have enough marginal pixels to hold boundary map. So embedding data in image pepper beyond 0.4 bpp is not possible in 1-LSB plane.

Boundary map size (bits)								
Embedding rate (bpp)	0.005	0.01	0.05	0.1	0.2	0.3	0.4	0.5
Lena	0	0	0	0	0	0	0	0
Airplane	0	0	0	0	0	0	0	0
Barbara	0	0	0	0	0	0	0	0
Baboon	0	0	0	0	0	2	18	109
Peppers	0	1	43	92	291	797	1741	-----
Boat	0	0	0	0	0	0	0	0

Table 1: Length of boundary map under various embedding rates.

b) Choice of LSB Plane Number

According to the proposed algorithm at first the image is divided into two parts A and B. The size of A is determined by the size of the message to be embedded and also by the number of LSB planes reversibly embedded in B. The choice of multiple LSB planes increases the size of B with an increase in embedding capacity. Table 2 shows the PSNR comparison between three different choices of LSB planes for five test images under various embedding rates measured by bits per pixel (bpp). From Table 2 we can find that single LSB plane is better at a low embedding rate of less than 0.2 bpp. For an embedding rate of 0.2 bpp and beyond, the choice of multiple LSB planes flips between 2 and 3 for a longer PSNR value.

PSNR Results (dB)									
Embedding Rate(bpp)		0.005	0.01	0.05	0.1	0.2	0.3	0.4	0.5
Lena	1 LSB Plane	67.16	63.44	55.46	52.33	49.07	45.00	40.65	35.84
	2 LSB Plane	66.48	62.65	54.69	51.55	48.39	45.10	42.56	39.46
	3 LSB Plane	64.41	60.94	52.95	49.96	46.79	43.98	41.91	39.53
Airplane	1 LSB Plane	65.94	63.18	57.02	54.20	50.98	48.26	44.67	40.78
	2 LSB Plane	65.48	62.33	55.91	53.05	49.87	48.10	45.05	42.73
	3 LSB Plane	63.47	60.97	53.87	50.79	47.65	45.79	43.88	42.19
Barbara	1 LSB Plane	65.39	62.56	55.56	51.46	47.68	43.56	39.24	34.80
	2 LSB Plane	65.00	61.85	54.72	50.71	47.25	43.70	40.78	37.53
	3 LSB Plane	63.33	60.50	53.16	49.36	45.98	42.81	40.34	37.58
Baboon	1 LSB Plane	57.49	55.71	50.19	46.17	40.68	35.87	31.16	25.92
	2 LSB Plane	57.43	55.47	49.87	45.92	40.41	36.47	33.08	29.85
	3 LSB Plane	57.10	55.13	49.23	45.40	40.09	36.33	32.96	30.19
Peppers	1 LSB Plane	63.77	61.30	54.17	51.02	46.00	42.08	36.91	--------
	2 LSB Plane	63.67	60.53	53.50	50.50	46.16	42.65	39.47	35.76
	3 LSB Plane	62.34	59.54	52.22	49.18	45.43	42.10	39.40	36.87
Boat	1 LSB Plane	67.22	64.13	56.75	52.62	49.10	45.21	41.24	35.99
	2 LSB Plane	66.72	63.26	55.75	51.71	48.40	44.98	42.46	39.98
	3 LSB Plane	64.57	61.34	53.73	50.02	46.71	43.81	41.70	39.46

Table 2: PSNR comparison for three different LSB-plane choices under various embedding rates.

CONCLUSION

Reversible data hiding (RDH) now a days plays an important role because it holds the ability to recover the cover image without any distortion. Encryption is also used along with RDH for privacy protection. All the available RDH techniques is implemented in encrypted images by vacating extra space after encryption. Moreover data hiding is done by simple LSB method in which the data is unsecure. In the proposed method extra space to embed data is reserved before encryption so, all traditional RDH algorithms could achieve better performance by using the proposed system without losing secrecy. More specifically, the system is well secured. The proposed method also proves to be a better solution in saving space for storing secret information. This method also makes use of additional level of encryption while embedding secret data. Furthermore, this novel method can achieve real reversibility, separate data extraction and greatly improvement on the quality of marked decrypted images and it also achieves higher security by the use of additional data encryption. In the present systems data is embedded as plain text. To increase the security some additional encryption level can be used for encrypting the data to be embedded so that the encrypted data can be embedded in the encrypted image. During the first extraction at the receiver side encrypted data is retrieved as output. When the additional data key is used the original data can be retrieved.

REFERENCES:

[1] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL, USA: CRC, 1996.

[2] K. Hwang and D. Li, "Trusted cloud computing with secure resources and data coloring," *IEEE Internet Comput.*, vol. 14, no. 5, pp. 14–22,Sep./Oct. 2010.

[3] J. Hwang , Kim and J. U Choi, "A reversible watermarking based on histogram shifting," Int. Workshop on Digital Watermarking, Lecture Notes in Computer Science, Jeju Island, Korea, 2006, vol. 4283,pp. 348–361, Springer-Verlag.

[4] A. M. Alattar, "Reversible watermark using the difference expansion of a generalized integer transform," IEEE Transactions on Image Processing, Vol. 13, 2004, pp. 1147-1156.

[5] Z. Ni, Y. Shi, N. Ansari, and S. Wei, "Reversible data hiding," IEEE Trans. Circuits Syst. Video Technol., vol. 16, no. 3, pp. 354–362, Mar.2006.

[6] X. Zhang, "Reversible data hiding in encrypted images," IEEE Signal Process. Lett., vol. 18, no. 4, pp. 255–258, Apr. 2011.

[7] W. Hong, T. Chen, and H.Wu, "An improved reversible data hiding in encrypted images using side match," *IEEE Signal Process. Lett.*, vol. 19, no. 4, pp. 199–202, Apr. 2012.

[8] X. Zhang, "Separable reversible data hiding in encrypted image," IEEE Trans. Inf. Forensics Security, vol. 7, no. 2, pp. 826–832, Apr. 2012J.

[9] W. Zhang, K. Ma, X. Zhao, N. Yu, F. Li "Reversible data hiding in encrypted images by reserving room before encryption, "IEEE trans. Info. forensics and security ,vol. 8, no. 3, pp. 553–558, March 2013.

[10] L. Luo*et al.*, "Reversible imagewatermarking using interpolation technique,"*IEEE Trans. Inf. Forensics Security*, vol. 5, no. 1, pp. 187–193,Mar. 2010.

[11] V. Sachnev, H. J. Kim, J. Nam, S. Suresh, and Y.-Q. Shi, "Reversiblewatermarking algorithm using sorting and prediction," *IEEE Trans.*

[12] Miscelaneous Gray Level Images [Online]. Available: http://decsai.ugr.es/cvg/dbimagenes/g512.php

[13] *Circuits Syst. Video Technol.*, vol. 19, no. 7, pp. 989–999, Jul. 2009.

[14] Tian, "Reversible data embedding using a difference expansion" IEEE Trans. Circuit Syst. Video Technol., vol. 13, no. 8, pp. 890–896,Aug. 2003.

[15] D. M Thodi and J. Rodriguez, "Expansion embedding techniques for reversible watermarking," IEEE Trans. Image Process., vol. 16, no. 3,pp. 721–730, Mar. 2007.

[16] Y. Hu, H.K Lee, Chen and J. Li, "Difference Expansion based reversible data hiding using two embedding direction," IEEE trans. multimedia, vol. 10, no. 8, pp. 1500–1512, 2008

[17] K. H Jung and K. Y Yoo, "Data hiding method using image interpolation," Journ. Of Compute standard and interfaces, vol. 31, pp. 465–470, 1996.

[18] Churchhouse, Robert (2002), *Codes and Ciphers: Julius Caesar, the Enigma and the Internet*, Cambridge: Cambridge University Press, ISBN 978-0-521-00890-7

[19] Tutte, W. T. (19 June 1998), *Fish and I*, retrieved 7 October 2010 Transcript of a lecture given by Prof. Tutte at the University of Waterloo