An Efficient Data Hiding Scheme using Hamming Error Correcting Code

Biswapati Jana [*]
Department of Computer Science, Vidyasagar University, Midnapore, Pin-721102, India
biswapatijana@gmail.com

Debasis Giri
Department of Computer Science and Engineering, Haldia Institute of Technology, Haldia Midnapore(East), Pin-721657,India
debasis_giri@hotmail.com

Shyamal Kumar Mondal
Department of Applied Mathematics with Oceanology and Computer Programming, Vidyasagar University, Midnapore, Pin-721102, India
shyamal_260180@yahoo.com

ABSTRACT

In this paper, we propose an efficient secure partially reversible data hiding scheme using hamming error correcting code (PRDHHC). Recently, Kim et al. proposed a data hiding technique in halftone image which hides four secret bits within (4×4) image block through codeword. Lien et al. presented data hiding scheme using hamming code by generating dispersed block through space filling curve decomposition using fifteen bits codeword. Here, we propose data hiding scheme through error creation in three least significant bits of a (7×7) image block using shared secret key. During extraction, the receiver finds the error position within three least significant bits of the image block and extracts secret data with the help of shared secret key. After extraction, we complement the error bits to achieve partial reversible data hiding. Our scheme is compared with other state-of-the-art methods and we obtain reasonably good performance in terms of PSNR and embedding capacity.

Keywords

Data Hiding, Hamming Code, Steganography, Partially Reversible Data Hiding

1. INTRODUCTION

Steganography means the art of hidden communications. The modern age of steganography is implemented computationally, where cover media such as text files, images, audio files, and video files are tweaked in such a way that a secret message can be embedded within them. Popular steganographic methods are based on the least significant bits ($LSBs$) replacement [1] [3] and the modulus operation [2] [9]. Both methods can encode and decode the message successfully but a good method of practice is to keep the message data as short as possible when using steganography. Instead of hiding only one bit into a block of pixels, efficient embedding using cover code were proposed by Zhang et al. [11]. Data hiding using block is commonly used to increase visual quality or to achieve reversibility [10] [6]. Reversible data hiding (RDH) presented by Ni et al. [8] is based on histogram shifting. Multilevel reversible data hiding based on histogram shifting is proposed by Lin et al. [7]. Data hiding using hamming code (DHHC) is recently proposed by Kim et al. [4] to hide information into halftone image. They used codeword to get a syndrome value. Four bits secret message can be embedded into four bits codeword using XOR operation. But changing individual pixels in halftone image may introduce visual distortion. Ma et al. [12] proposed a scheme to improve Kim et al.'s method [4] by changing pair of pixel rather than individually which sacrifice data hiding capacity reduced by half. Lien et al. [5] proposed a dispersed data hiding scheme using hamming code (DDHHC) through space filling curve decomposition. Pixels consist in each block randomly and are distributed uniformly all over the cover image. In this paper, we propose an efficient, Partially Reversible Data Hiding using Hamming Code (PRDHHC) with a shared secret key. Here, secret message is embedded within the cover media by the simple creating error using complement and error is found at the receiver end with the help of hamming error correcting code.

- Our motivation is to send any arbitrary length of secret message with security. Our scheme has been designed in such a manner that receiver can easily find the end of message by finding no error continuously in a stego image.
- Another aim of this work is partial reversibility. To get the secret message, Hamming error correcting code is used to detect the message embedding position by which receiver can complement the bit position to get partial modification image called partial reversible.
- Finally, enhance the security in data hiding using shared secret key which helps us to extract secret message from the stego image. Without this secret key it is hard to extract the secret message from stego image.

The rest of the paper is organized as follows. Section-2 reviews some required literature review. Our propose method

[*]Correspondence Author

ICCCT '15, September 25-27, 2015, Allahabad, India

ACM 978-1-4503-3552-2/15/09 ...$15.00.
http://dx.doi.org/10.1145/2818567.2818674.

is discussed in Section 3. Experimental results and comparisons are given in Section 4 and security analysis of the proposed scheme is presented in Section 5. Finally, some conclusions are given in Section 6.

2. LITERATURE REVIEW

Recently, data hiding becomes a very important role in medical image processing and military communications. Information can be embedded into image which contain ownership identification, authentication and copy right protection. Kim et al. employed hamming code $(15, 11)$ to hide secret data into a halftone image and Lien et al. proposed dispersed data hiding using hamming code with recovery capability. Here, we present a short review of Kim et al.'s [4] and Lien et al.'s [5] hamming code based data hiding schemes.

2.1 Kim et al.'s scheme

Data hiding using hamming code (DHHC) employed hamming code $(15, 11)$ to hide secret data into a halftone image are proposed by Kim et al. [4]. In their scheme, the cover image is divided into blocks of size (4×4). Fifteen bits codeword are used from a block and syndrome is calculated from codeword. Then 4-bit secret message can hide using Exclusive OR operation within the codeword. The halftone image of size $(n \times n)$ which composes n pixels is divided into continuous block of size (4×4). They use code length $n \ \{= (2^{r-1})\}$ and the number of bits that are encoded in each codeword is $k \ \{= (n - r)\}$, where r be a non-negative integer. They considered codeword having minimum hamming distance d as three, so that, one error can be corrected and two errors can be detected. For n bits code $[\log_2 n]$ bits are required. The parity check matrix for the hamming code is

$$H = \begin{vmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{vmatrix} \qquad (1)$$

For a codeword c, they calculate syndrome as $S = H \times c^t$, where H is the parity bit and c is the seven bits sequence binary number. If the syndrome value is non-zero, it denotes the position of the bit error. For example, suppose message $m = 101$, codeword $c = 1101001$. Then using syndrome calculation equation with parity checker H and codeword one can calculate the syndrome $H \times c^t = (000)^t$. To hide the secret message, an Exclusive OR is computed $w = H \times c^t \oplus m$. If w is zero, then no need to flip, otherwise find the w^{th} column of c and flip the w^{th} pixel.

2.2 Lien et al.'s scheme

The dispersed data hiding using hamming code (DDHHC) has been proposed by Lien et al. [5]. First they divide image into sixteen sub images. Using space filling curve, they put the index of each sub image. Those pixels corresponding to the same index in each sub image are gathered as a block of sixteen pixels. To embed $4 \times m$ bits of secret message they randomly select m blocks from the image. Then these m blocks are sorted by the index number and each four bits secret message are embedded into the fifteen bits code word randomly. DDHHC uses two bit toggles with one of its neighboring pixel whereas DHHC flips the value in the selected location. To read the embedded data, the stego image is partitioned into blocks using space filling curve partition.

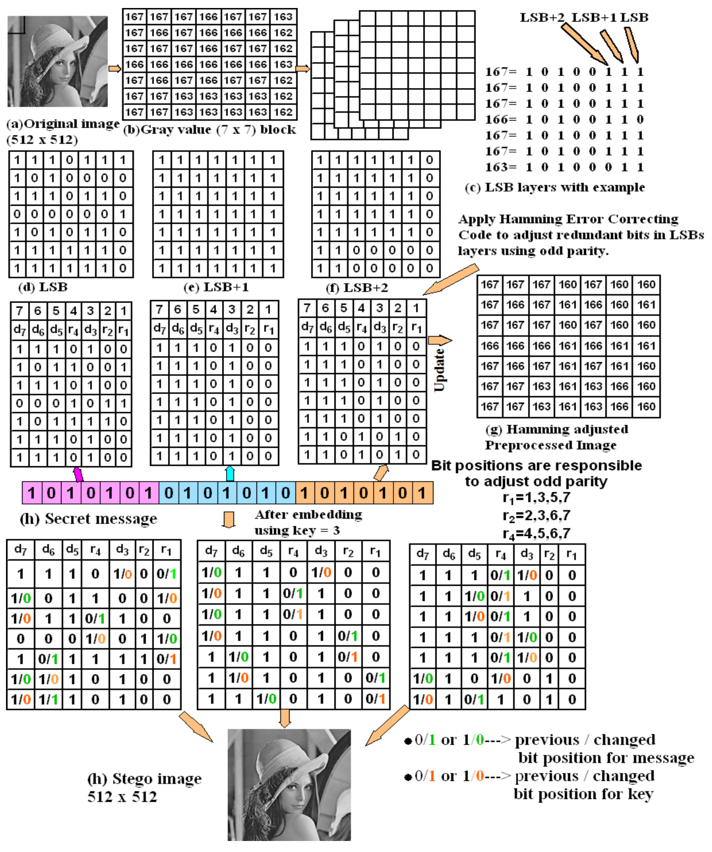

(a)Original image (512 x 512)
(b)Gray value (7 x 7) block
LSB+2 LSB+1 LSB
167= 1 0 1 0 0 1 1 1
166= 1 0 1 0 0 1 1 0
163= 1 0 1 0 0 0 1 1
(c) LSB layers with example
(d) LSB
(e) LSB+1
(f) LSB+2
Apply Hamming Error Correcting Code to adjust redundant bits in LSBs layers using odd parity.
Update
(g) Hamming adjusted Preprocessed Image
Bit positions are responsible to adjust odd parity
r1=1,3,5,7
r2=2,3,6,7
r4=4,5,6,7
(h) Secret message
After embedding using key = 3
(h) Stego image 512 x 512
•0/1 or 1/0---> previous / changed bit position for message
•0/1 or 1/0---> previous / changed bit position for key

Figure 1: Block diagram of message embedding

Then the secret data can be extracted by examining the syndrome computed from the fifteen bits codeword of each block. In this technique, they calculate modified PSNR to measure the visual quality of the image which shows on an average 44 dB for embedding 4096 bits secret data.

3. PROPOSED PARTIALLY REVERSIBLE DATA HIDING SCHEME USING HAMMING CODE (PRDHHC)

3.1 Message embedding stage

Here, we consider a (7×7) pixel block from the gray scale original image. We convert the pixel from gray value to binary. We collect only LSB, $LSB + 1$ and $LSB + 2$ from the selected block. 1^{st}, 2^{nd}, 4^{th} and 8^{th} bits are to be adjusted for error correction using odd parity. We repeat this process for all blocks of the original image. After applying odd parity, a preprocessed image is generated, which is considered as cover image. Now, we use shared secret key (between 0 to 7) to embed data. We can complement one bit at the position of key in the (7×7) block. The same secret key will be used for three LSBs. In the first row of first block we use key, that is, we create error by complement at the key position of LSBs. Then we embed binary data bit by creating error at any bit of the block except the key position. In the second row, we again create error by complement in the error position of the previous row, that is, data embedding position of the previous row will be the key of the next row then embed data by creating error at any position of the second row LSBs except the key position. This process will be continued for the next block by block using LSB, $LSB+1$ and $LSB+2$ bits. As a result, we can embed maximum $1,12,128$ bits, that is, $(73 \times 3 \times 512)$ bits within a (512×512) gray scale image. The block diagram of the message embedding process is depicted in Figure 1. The corresponding algorithm is shown in **Algorithm-1**.

```
Algorithm-1:
Input: Original image $I_{M\times N}$, key K (between 0 to 7), secret data D;
Output: Stego image ($S_{M\times N}$);
Initialization: C =I; sq=7; M=512; N=512 ;
Step-1:
for p=1 to M/sq do
    for q=1 to N/sq do
        $BC_{pq}$ from $I_{M\times N}$ ;
    end
    for i=1 to sq do
        for j=1 to sq do
            $LSB_{(i,j)}$=1st LSB bit of $BC_{pq(i,j)}$ ;
            $LSB+1_{(i,j)}$=2nd LSB bit of $BC_{pq(i,j)}$ ;
            $LSB+2_{(i,j)}$=3rd LSB bit of $BC_{pq(i,j)}$ ;
        end
    end
    for Each LSB Matrix= LSB,LSB+1,LSB+2 do
        apply hamming code row wise;
    end
    for i= $(sq \times (p-1))+1$ to $(sq \times p)$ do
        for j= $(sq \times (q-1))+1$ to $(sq \times q)$ do
            Replace 3 LSBs of $C_{pq(i,j)}$ by LSB+2,LSB+1,LSB;
        end
    end
    for Each LSBs hamming code matrix do
        for i=1 to sq do
            if $i \leq sq$ then
                change LSB(i,key) by (0 to 1 / 1 to 0) ;
            end
        end
        for j=1 to sq do
            if $j \neq key$ and $LSB\ (i,j) \neq d_r$ then
                change LSB(i,j) by (0 to 1 / 1 to 0) ;
                Key is updated to j i.e ($key = j$) ;
                break ;
            end
        end
        if $(r = length(D))$ then
            goto Step-3;
        end
        increase r for selecting next bit $d_{r+1}$ ;
    end
    for i= $(sq \times (p-1))+1$ to $(sq \times p)$ do
        for j= $(sq \times (q-1))+1$ to $(sq \times q)$ do
            Replace LSB 3 bits of $S_{pq(i,j)}$ by LSB+2, LSB+1, LSB;
        end
    end
    if $(r = length(D))$ then
        goto Step 2;
    end
end
Step-2:Get stego image $S_{M\times N}$ ;
Step-3: End.
```

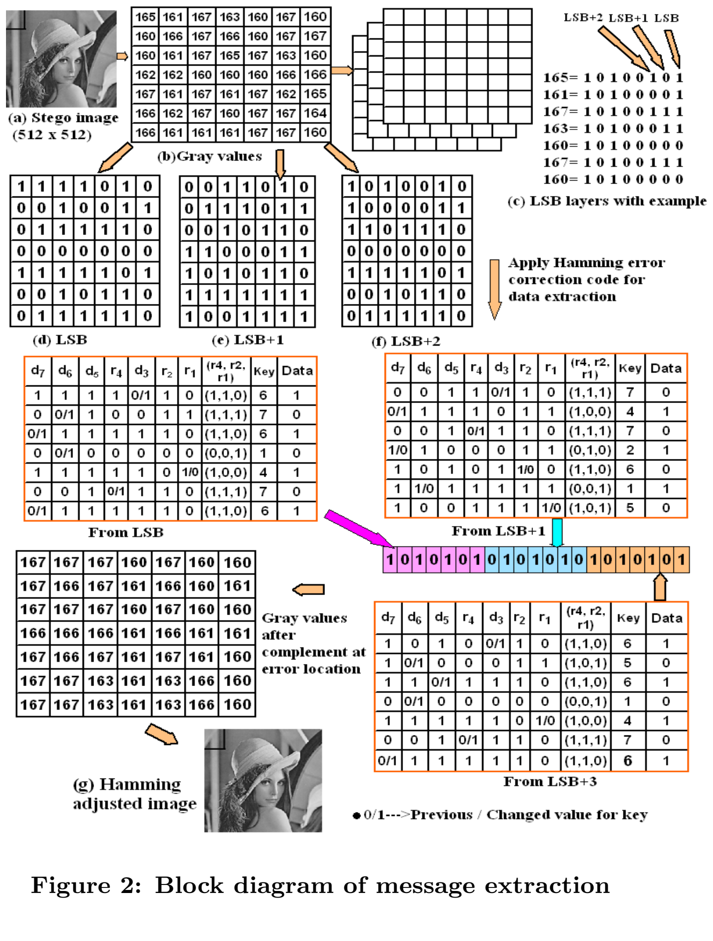
Algorithm 1: Data embedding algorithm in PRDHHC

3.2 Message extraction stage

At the receiver end, we collect LSBs from stego image, complement bit at the position of key in the first row of the first block for three LSBs. Then we apply hamming error correcting code to find error using odd parity. We collect the secret data at the error position. After extraction, we complement that bit position which will produce preprocessed image that is cover image. In the second row, key will be updated by the error position of the previous row. So, we complement that key position and then apply hamming error correction code to find the error location, which is the secret data. After extraction, we complement the bits of LSBs to achieve partially reversibility. We repeat these processes for every block of the stego image until no message has been left. The process will be automatically stopped when no error found situation occur. As a result, no message length is required during extraction in this approach. The extraction

```
Algorithm-2:
Input: Stego image $S_{M\times N}$, key K;
Output: Cover image $C_{M\times N}$ ,secret data stream ($D$);
Initialization: sq=7, M=512, N=512 ;
Step-1:
for p= 1 to M/sq do
    for q= 1 to N/sq do
        $BS_{pq}$ from $S_{M\times N}$
    end
    for i= 1 to sq do
        for j =1 to sq do
            $LSB_{(i,j)}$=1st LSB bit of $BS_{pq(i,j)}$ ;
            $LSB+1_{(i,j)}$=2nd LSB bit of $BS_{pq(i,j)}$ ;
            $LSB+2_{(i,j)}$=3rd LSB bit of $BS_{pq(i,j)}$ ;
        end
    end
    for Each LSBs Stego matrix do
        for i= 1 to sq do
            if $(i \leq sq)$ then
                Change LSB $(i, key)$ by (0 to 1 / 1 to 0) ;
            end
        end
        Apply Hamming in $i^{th}$ row of LSB matrix to ↓nd the error at
        $j^{th}$ column $d_r$=LSB$(i,j)$ ;
        Store Secret data D=$d_r$;
        Change LSB $(i,j)$ by (0 to 1 / 1 to 0) ;
        key=j ;
        if No error found then
            goto Step-2 ;
        end
        increase r to retrieved next bit $d_{r+1}$ ;
    end
    for $i = (sq \times (p-1))+1$ to $(sq \times p)$ do
        for $j = (sq \times (q-1))+1$ to $(sq \times q)$ do
            Replace LSB 3 bit of $C_{pq(i,j)}$ by LSB+2, LSB+1, LSB ;
        end
    end
end
Step-2: Retrieve $C_{M\times N}$ , secret data ($D$) ;
Step-3: End.
```

Algorithm 2: Data extraction algorithm in PRDHHC



(a) Stego image (512 x 512)
(b)Gray values
(c) LSB layers with example
LSB+2 LSB+1 LSB
(d) LSB
(e) LSB+1
(f) LSB+2
Apply Hamming error correction code for data extraction
From LSB
From LSB+1
From LSB+3
Gray values after complement at error location
(g) Hamming adjusted image
● 0/1--->Previous / Changed value for key

Figure 2: Block diagram of message extraction

process is shown in Figure 2. The corresponding algorithm is shown in **Algorithm-2**.

4. EXPERIMENTAL RESULT AND COMPARISON

The distortion is measured by means of two parameters namely, Mean Square Error (MSE) and Peak Signal to Noise Ratio ($PSNR$). The MSE is calculated as follows

$$MSE = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}[X(i,j) - Y(i,j)]^2}{(M \times N)}, \tag{2}$$

where M and N denote the total number of pixels in the horizontal and the vertical dimensions of both the original image and stego image. $X(i,j)$ represents the pixels in the original image and $Y(i,j)$ represents the pixels of the stego image. The difference between the original and stego image are assessed by the Peak Signal to Noise Ratio ($PSNR$). The formula of PSNR is as follows

$$PSNR = 10\ log_{10}\frac{255^2}{MSE} \tag{3}$$

Higher the values of PSNR between two images indicates better the quality of the stego image and very similar to the cover image where as low PSNR demonstrates the opposite.

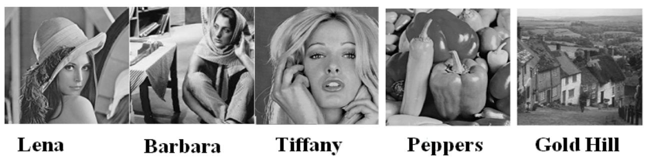


Figure 3: Standard original images (512×512) **pixels**

Here, gray scale (512×512) pixels images are used as original images, which are shown in Figure 3. The stego images, after embedding secret data are shown in Figure 4. The PRDHHC algorithms: Data embedding and Data extraction are implemented in MATLAB Version 7.6.0.324 (R2008a).

Table 1: PSNR of Original Image (OI), Stego Image (SI) and Cover Image (CI) after embedding 4096 **and** 16384 **bits secret data**

	4096 bits			16384 bits		
Image	OI & SI	OI & CI	CI & SI	OI & SI	CI & OI	CI & SI
Lena	53.52	54.45	54.58	46.89	47.94	48.03
Barbara	51.67	51.52	53.58	44.88	44.69	46.94
Ti↑any	53.29	54.48	54.57	46.87	47.98	48.04
Pepper	52.31	51.97	53.43	45.85	45.68	47.04
Gold hill	49.76	48.82	52.23	44.98	42.21	45.67

Table 1 shows the PSNR of original image (OI), stego image (SI) and cover image (CI) after embedding 4096 and

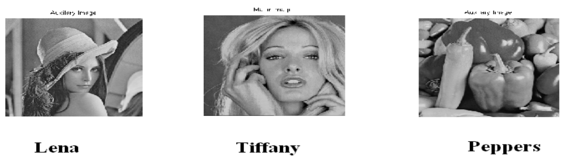


Figure 4: Stego images (512×512) **pixels.**

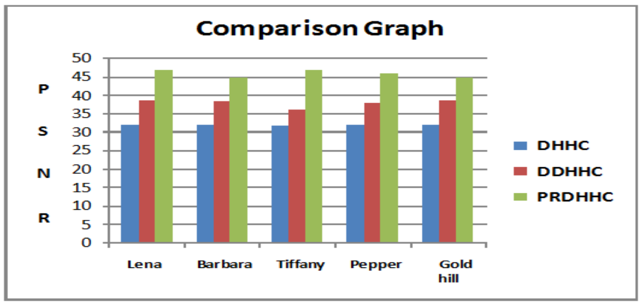


Figure 5: PSNR comparison graph with DHHC, DDHHC and PRDHHC after embedding 4096 **bits**

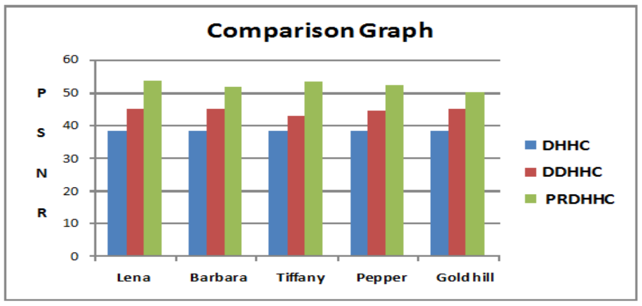


Figure 6: PSNR Comparison graph with DHHC, DDHHC and PRDHHC scheme after embedding 16384 bits

16384 bits. We get cover image (CI) after updating redundant bits of original image (OI) for hamming error correcting code. The stego image (SI) is generated after embedding the secret bits within cover image (CI). In our experiment, we found minimum PSNR changes among OI, CI and SI. The {PSNR of (OI & CI) - PSNR of (OI & SI)} is for adjustment of hamming error correcting code and {PSNR of (CI & SI)-PSNR of (OI & SI)} is for data embedding. The quality of stego image is not much degraded after embedding secret data.

Table 2: Comparison of PRDHHC with DHHC and DDHHC in terms of PSNR

	16384 bits			4096 bits		
Scheme	DHHC	DDHHC	PRDHHC	DHHC	DDHHC	PRDHHC
Lena	32.03	38.60	46.89	38.12	44.71	53.52
Barbara	32.03	38.57	44.88	38.14	44.77	51.67
Ti↑any	31.72	36.24	46.87	38.04	42.91	53.29
Pepper	31.98	38.02	45.85	38.10	44.19	52.31
Gold hill	32.02	38.66	44.98	38.13	44.96	49.76
Average	31.96	38.02	45.89	38.11	44.31	52.11

Table 2 shows the comparison of PRDHHC with Kim et al.'s (DHHC) scheme [4] and Lien et al.'s (DDHHC) scheme [5]. In (DHHC) scheme, the PSNR of lena image is 32.03 dB when embed 16384 bits and 38.12 dB when embed 4096 bits. In Lien et al.'s scheme, the PSNR of lena image is 38.60 dB when embed 16384 bits and 44.71 dB when embed 4096 bits, but in our scheme PSNR of lena image is 46.89 dB and 53.52 dB when embed 16384 and 4096 bits respectively. In this scheme, we achieve higher PSNR than other existing schemes. The comparison graph of DHHC, DDHHC, and proposed PRDHHC scheme are shown in Figure 5 and Figure 6 when 4096 bits and 16384 bits are embedded. The PSNR of our scheme is better than Kim et al.'s (DHHC) scheme [4] and Lien et al.'s (DDHHC) scheme [5]. We can measure embedding capacity (payload) using following equation

$$B = \frac{|\delta|}{m \times n}(bpp) \tag{4}$$

In our experiment $|\delta|$ represents number of bits in secret message and B denotes bits per pixel (*bpp*) which is the payload. In a (512 × 512) image, the embedding capacity is 1,12,128 bits. So the B = 0.43 bpp.

5. SECURITY ANALYSIS

5.1 RS Analysis

We analyze our stego images by RS analysis. Let us assume that we have a cover image of size (M × N). In RS analysis method, first the stego image is divide into disjoint groups G of n adjacent pixels $(x_1, \ldots, x_n)$. Each pixel value is in a set P that is $P = \{0, 1, \ldots, 255\}$. Here, each group consists of 4 consecutive pixels in a row. We define a discrimination function f that returns a real number $f(x_1, \ldots, x_n) \in R$ to each pixel group $G = (x_1, \ldots, x_n)$. The main goal of use of discrimination function is to identify the "Smoothness" or "Regularity" of each group of pixels G. The discrimination function f is define as

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{n-1} |x_{i+1} - x_i| \tag{5}$$

An invertible function F is defined which operates on P, called "flipping". Flipping consists of two-cycles which permutes the pixels value. So, F^2= Identity or $F(F(x)) = x$ for all x belongs to P. Flipping the LSB of each pixel value and the corresponding permutation F_1 is: $0 \leftrightarrow 1, 2 \leftrightarrow 3, \ldots, 254 \leftrightarrow 255$. We define another function, named shift LSB flipping and treated as F_{-1}. So the permutation F_{-1}: $-1 \leftrightarrow 0, 1 \leftrightarrow 2, \ldots, 255 \leftrightarrow 256$. In the other word, F_{-1} flipping can be define as

$$F_{-1}(x) = F_1(x + 1) - 1, \tag{6}$$

for all x belong to P. There are three types of groups, Regular (R), Singular (S) and Unusable (U) groups, which are defined depending on the discrimination function f and the flipping operation F. Depending on the condition, groups are define below.

$$\begin{cases} G \in R & \text{if } f(F(G)) > f(G) \\ G \in S & \text{if } f(F(G)) < f(G) \\ G \in U & \text{if } f(F(G)) = f(G) \end{cases} \tag{7}$$

where $F(G) = F(x_1), \ldots, F(x_n)$.
The flipping operation will be executed with the help of a mask value M, which is a n tuples with values -1, 0, and 1. The flipped group $F_M(G)$ is defined as $(F_M(1)(x1), F_M(2)(x2), \ldots, F_M(n)(xn))$. Then we calculate the value of RS analysis as follows

$$(|R_M - R_{-M}| + |S_M - S_{-M}|)/(R_M + S_M) \tag{8}$$

where R_M and R_{-M} are the total number of regular group with mask M and $-M$ respectively. S_M and S_{-M} are the total number of singular group with mask M and $-M$ respectively. When the value of RS analysis is closed to zero means the scheme is secure. It is observed from Table-3 that the values of R_M and R_{-M}, S_M and S_{-M} are nearly equal for stego image. Thus rule $R_M \cong R_{-M}$ and $S_M \cong S_{-M}$ is satisfied for the stego image in our scheme, that means the difference is nearly zero. So, the proposed method is secure against RS attack.

Table 3: RS analysis of stego image

Image	Data	Stego image				
		R_M	R_{-M}	S_M	S_{-M}	RS value
Cameraman	20000	7118	7107	3551	3594	0.0051
	50000	6768	6851	3944	3895	0.0123
	75000	6304	5947	4943	5279	0.0616
	112128	6207	6035	4997	5173	0.0311
Lena	20000	5617	5607	4067	4068	0.0011
	50000	5563	5476	4291	4337	0.0135
	75000	5636	5539	4517	4589	0.0166
	112128	5641	5387	4509	4709	0.0447
Baboon	20000	5893	5815	4960	5105	0.0205
	50000	5897	5875	5076	5131	0.0070
	75000	6018	5813	5107	5313	0.0369
	112128	5844	5986	5256	5123	0.0248

Table 4: Relative entropy between I and Stego image

Image	Data	Entropy I	Entropy S	Difference
Lena	20000	7.4451	7.4452	0.0105
	20249	7.4451	7.4453	0.0131
Barbara	20000	7.0480	7.0485	0.0112
	20249	7.0480	7.0486	0.0134
Tiffany	20000	7.2925	7.2926	0.0122
	20249	7.2925	7.2926	0.0129
Pepper	20000	7.2767	7.2770	0.0142
	20249	7.2767	7.2771	0.0169
Gold hill	20000	7.2367	7.2375	0.0112
	20249	7.2367	7.2379	0.0143

5.2 Relative Entropy

To measure the divergence of stego image (S) from original image (I) is defined as (D) between the probability distributions of the original image (I) and the stego image (S) are calculated by

$$D(I||S) = \sum_{x=0}^{255} I(x) log \frac{I(x)}{S(x)}, \tag{9}$$

When the relative entropy (D) lies between two probability distribution functions is zero then the system is perfectly secure. D($I||S$) is a nonnegative continuous function and equals to zero if and only if $I(x)$ and $S(x)$ are coincide. Thus D($I||S$) can be normally considered as a distance between the measures $I(x)$ and $S(x)$. The experiment results are shown in Table 4. It is shown that when the number of bits in the secret message increases, the relative entropy in stego image also increases. The difference of relative entropy nearer to zero, which implies the proposed scheme provides secure hidden communication.

Table 5: Standard Deviation (SD) and Correlation Coefficient (CC)

Image	SD		CC
	Image (I)	Stego image (S)	(I) and (S)
Baboon	38.3719	37.8500	0.9820
Cameraman	61.5978	61.1221	0.9913
Lena	47.8385	47.4358	0.9864

5.3 Statistical Attack

The proposed scheme is also assessed based on statistical distortion analysis by some image parameters like Standard Deviation (SD) and Correlation Coefficient (CC) to check the impact on image after data embedding. The SD before and after data embedding and CC of original and stego image are summarized in Table 5. Minimizing parameter difference is one of the primary aims in order to get rid of statistical attacks. From Table 5 it is seen that there is no substantial divergence between the standard deviation of the cover-image and the stego-image. The SD of original image is 47.8385 and the SD of stego image is 47.4358, and the difference is 0.4027 for lena image. The CC between the image I and S is 0.9864 for lena image which implies the change in the original image will predict a change in the same direction in the stego image. So it is hard to locate the embedding position in the stego image. This study shows that the magnitude of change in stego-image based on image parameters is small from a original image. Since the image parameters have not changed much, the method offers a good concealment of data and reduces the chance of the secret data being detected. Thus, it indicates a perfectly secure steganographic system.

6. CONCLUSION

In this paper, a partially reversible data hiding scheme using hamming code (PRDHHC) has been proposed where only least three significant bits of every pixel block are used for data embedding. The PSNR is 53.52 dB in PRDHHC for Lena image when embed 4096 bits and 46.89 dB when embed 16384 bits. Relative entropy of the original image and the stego image is vary depending upon the number of message bits of secret message. In our experiment, it is shown that when the number of message bits in the secret message is increasing then the relative entropy in stego image is also increasing. A shared secret key is used by which receiver can extract the secret message. Without key it is hard to extract secret message. In this approach, message length is not required to send to the receiver, that is without knowing the message length, receiver can successfully retrieve the secret message. This scheme recovers the cover image but does not recover the original image, that is why this scheme is partially reversible scheme. Further, we have compared our proposed PRDHHC scheme with existing schemes and it shows better result than the previous schemes. One can use dual image to recover original image.

7. REFERENCES

[1] C.-K. Chan and L.-M. Cheng. Hiding data in images by simple lsb substitution. *Pattern recognition*, 37(3):469–474, 2004.

[2] C.-C. Chang, C.-S. Chan, and Y.-H. Fan. Image hiding scheme with modulus function and dynamic programming strategy on partitioned pixels. *Pattern Recognition*, 39(6):1155–1167, 2006.

[3] C.-C. Chang, J.-Y. Hsiao, and C.-S. Chan. Finding optimal least-significant-bit substitution in image hiding by dynamic programming strategy. *Pattern Recognition*, 36(7):1583–1595, 2003.

[4] C. Kim, D. Shin, and D. Shin. Data hiding in a halftone image using hamming code (15, 11). In *Intelligent Information and Database Systems*, pages 372–381. Springer, 2011.

[5] B. K. Lien, S.-K. Chen, W.-S. Wang, and K.-P. King. Dispersed data hiding using hamming code with recovery capability. In *Genetic and Evolutionary Computing*, pages 179–187. Springer, 2015.

[6] B. K. Lien and Y.-m. Lin. High-capacity reversible data hiding by maximum-span pairing. *Multimedia Tools and Applications*, 52(2-3):499–511, 2011.

[7] C.-C. Lin, W.-L. Tai, and C.-C. Chang. Multilevel reversible data hiding based on histogram modification of difference images. *Pattern Recognition*, 41(12):3582–3591, 2008.

[8] Z. Ni, Y.-Q. Shi, N. Ansari, and W. Su. Reversible data hiding. *Circuits and Systems for Video Technology, IEEE Transactions on*, 16(3):354–362, 2006.

[9] C.-C. Thien and J.-C. Lin. A simple and high-hiding capacity method for hiding digit-by-digit data in images based on modulus function. *Pattern recognition*, 36(12):2875–2881, 2003.

[10] F.-X. Yu, H. Luo, and S.-C. Chu. Lossless data hiding for halftone images. In *Information Hiding and Applications*, pages 181–203. Springer, 2009.

[11] W. Zhang, S. Wang, and X. Zhang. Improving embedding efficiency of covering codes for applications in steganography. *Communications Letters, IEEE*, 11(8):680–682, 2007.

[12] M. Zhipeng, wind Li Yong, and Z. Xinpeng. Based on hamming and subordinate pixel compensation halftone image information hiding. *Journal of Shanghai University (Natural Science)*, 19(2):111–115, 2013.