Image Steganography By Closest Pixel-pair Mapping

Adnaan Ahmed, Nitesh Agarwal, Sabyasachee Banerjee
Department of Computer Science and Engineering
Heritage Institute of Technology
West Bengal, Kolkata 700007
Email: adnaan.1703@gmail.com
niteshagarwal002@gmail.com
sabyasachi.banerjee@heritageit.edu

***Abstract*—Steganography is one of the important and elegant tools used to securely transfer secret message in an imperceptible manner. Visual Steganography is another added feature of it. It is the steganographic method involving multimedia files like image, video etc. to hide a secret message. However this method may result in the distortion of the colour frequencies of the cover image which is predictable by some analysis. Here in this paper we have proposed a method for steganography which results in absolutely no distortion of the cover image. The proposed image is independent of the size of the cover image and the secret image i.e. a larger image can be hidden in a smaller image. The proposed method also uses AES Encryption for secure transfer of the stego-key. The nexus of this cover image and the encrypted data serves the purpose of secure transfer of secret data.**

Keywords: *Reference hash-table (RHT), Image Steganography, AES Encryption, PSNR value, Histogram, Memoization lookup-table (MLT).*

I. Introduction

The practice or technique of concealing some significant and crucial data within other trivial data is called Steganography. Steganography basically works on the principle of Security through Obscurity [14] [11]. The most extensively practiced form of steganography is Visual steganography and is usually achieved through image files. Our focus will be on image files to accomplish visual steganography. A blend or combination of various regions consisting of pixels constitute an image. Every pixel itself is an amalgamation of three basic colors G (green), R (red), and B (blue). Now these RGB values of various pixels can be manipulated to some extent to hide data in images. There is no significant difference observed between the original and resultant image if there's a marginal deviation. There's just a slight shade difference in the altered region and that too is not visible in normal conditions.

Many cryptography algorithms have been designed till date all with the primary objective of converting information into unreadable ciphers [12]. Here we have used joint key cryptography that employs a common key to both, the encryption and decryption of the message. This key is transmitted confidentially by the sender to the receiver [3]. We intend to use the vastly secure type of Block Cipher, namely the AES-Encryption (Advanced Encryption Standard) [9]. We will use 128- bit key for encrypting the main data (which is nearly 8 times faster than Triple DES [1]) and the key for this encrypted data will be further encrypted using AES-Encryption that uses a 256-bit key [3] [8].

II. Related Work

Least significant bit of each pixel of the cover image is substituted by the secret bits sequentially [4]. The image which serves as a cover for the information so as to achieve steganography, is called as the cover image. The edited image, called as the Stego image or Encrypted image or embedded image that can be transmitted to the receiver via the original image. The receiver then can decode the data from the image by pixel based image comparison [4]. Since only LSB is being substituted, there will be very less visible change in the cover image. But as the relative size of the secret message increases with respect to the cover image, quality of the stego-image is compromised [8] [4].

The simple LSB substitution can be modified to improve the quality of the stego-image. This improvement is done by the Optimal LSB Method which applies Optimal Pixel Adjustment Process (OPA). In this method three pixels are picked and one whose pixel value is closest to the original pixel value with the secret data embedded is chosen to conceal the secret data [4].

LSB replacement (LSBR) [6] is an embedding scheme, where only the LSB plane of the cover image is replaced with the secret image according to a pseudorandom number generator. As a result, some structural asymmetry is introduced, and thus it is very easy to detect the existence of hidden message.

LSB matching (LSBM) [6] is a minor alteration to LSB replacement. +1 or -1 is randomly added to the corresponding pixel value, if the secret bit does not match the LSB of the cover image. The obvious asymmetry artefacts introduced by LSB replacement is easily avoided by using LSBM.

LSB matching revisited (LSBMR) [6] [7] uses a pixel pairs as an embedding unit, in which one bit of secret message is there in LSB of the first pixel, and another bit of secret message is carried by the relationship (odd-even combination) of the two pixel values.

But the biggest drawback of LSB substitution, LSBR, LSBM, LSBMR and many such related methods lies in the fact that the histogram comparison of the two images in consideration give away the difference hence leading to an end of the obscurity(fig-1). Hence, we decided not to actually embed any data in the cover image and rather map data to be transmitted with respect to the cover image.

978-1-4799-3080-7/14/$31.00

III. Proposed Work

Steganographic algorithms usually use a reference data, which are processed along with the stego-image to get the secret data. Here in this paper we have proposed a method to structure this reference data in such a way so that both the stego-image and secret image suffers absolutely no distortion. In order to provide high security we encrypt this reference data with AES encryption method for safe transmission of data. On the receivers end the reference data can be decrypted and can be used to extract the secret data *(NOTE: In our entire approach we have assumed the image to be column major matrix.)*.

A. Encryption algorithm

First we create a reference hash table *(RHT)* with its size being that of the secret image. Size of the secret image is given as $R_s * C_s * P_s$ (where R_s = number of rows in each plane of the secret image, C_s = number of columns in each plane of the secret image and P_s = page width i.e. number of image planes of the secret image). It will have two attributes.

i) Pixel position.

ii) Pixel difference.

Pixel position specifies the closest matched pixel obtained from the cover image. Pixel difference is the difference between the value of pixel position and the current pixel value of the stego image.

TABLE I: Representation of the reference hash table

INDEX	1	2	.	.	S_s
PIXEL POSITION					
PIXEL DIFFERENCE					

This *RHT* will map corresponding pixel positions of the secret image to the pixel position of the cover image with least pixel value difference. If there exists a pixel value in the cover image which is same as that of the pixel value of secret image then the position of that pixel value in the cover image will be recorded in the *RHT* with the index being the pixel position of the secret image. Since the value is same the pixel difference attribute will have the value 0.

If exact same pixel value doesn't exist in the cover image then the pixel position of such pixel is recorded whose value is closest to the pixel of the secret image. Then the pixel difference attribute will store the difference in the pixel. This helps to get the exact pixel value later for all three color pixels (RGB for color image).

To make this process faster we have introduced a memoization lookup table *(MLT)* which acts like a virtual cache memory. We know that in an image every pixel has 8 bits; hence it can have values in the range of 0 to 255. Pixel position for each pixel value which has already been searched is entered in the *MLT* so that when the same position is again required it can be obtained from the *MLT* in constant time. Hence by this technique of space time trade-off we have drastically reduced the processing time.

1) Pseudo-code For Encryption Algorithm: / cvrIm, secIm denotes the Cover Image and Secret Image respectively. mlt and rht denotes the Memory lookup table and Reference Hash Table respectively.*/*

TABLE II: Representation of the memoization lookup table

INDEX	0	1	2	3	.	.	.	255
PIXEL POSITION	0	0	0	.	.	.	.	0
PIXEL DIFFERENCE	0	0	0	.	.	.	.	0

ImageEncryptor (cvrIm [ ][ ][ ], secIm [ ][ ][ ])
1. Store the rows, columns and page-width of cvrIm in (r1,c1,p1) Store the rows, columns and page-width of secIm in (r2,c2,p2) Calculate the total no. of pixels of cvrIm(s1<- r1*c1*p1)
2. Calculate the total no. of pixels of secIm(s2<- r2*c2*p2) let mlt [1.256][12] be a new array
3. Initialize mlt [ ][ ] with -1
4. let rht [1.s2][12] be a new array
5. for i=1 to s2
  5.1. low = ∞
  5.2. flag = 0
/ This section checks whether there is any memoization look-up table match */*
  5.3. if mlt [secIm[i]][1] > -1
    5.3.1. rht [i][1] = mlt [secIm[i]][1]
    5.3.2. rht [i][2] = mlt [secIm[i]][2]
    5.3.3. continue
  5.4. end if
/ This section checks if there is any exact match in the cover image */*
  5.5. for j=1 to s1
    5.5.1. if secIm[i] == cvrIm[j]
      5.5.1.1. rht [i][1] = j
      5.5.1.1. rht [i][2] = 0
      5.5.1.3. mlt [secIm [i]][1] = j
      5.5.1.4. mlt [secIm[i]][2] = 0
      5.5.1.5. flag = 1
      5.5.1.5. break
*/*This section keeps track of the closest value in the cover image along with its pixel difference*/*
    5.5.2. end if else
    5.5.3. if abs (low) > abs (cvrIm [j]- secIm[i])
      5.5.3.1. low = secIm [i]- cvrIm [j]
      5.5.3.2. temp = j
    5.5.4. end if
  5.6. end for
  5.7. if flag == 0
    5.7.1. rht [i][1]=temp rht [i][2]=low
    5.7.2. mlt [secIm [i]+1][1] = temp
    5.7.3. mlt [secIm [i]+1][2] = low
  5.8. end if
6. end for
*/*This section stores the secret image properties in the RHT*/*
7. rht[end+1][1] = r2
8. rht[end][2] = c2
9. rht[end+1][1] = p2
10. rht[end][2] = s2
11. let enc be a new file for storing the encrypted reference String enc=AES-Encryption(rht)
end

2) Time complexity analysis For Encryption Algorithm:
Let the total size of the cover image be $r1 * c1 = n$.
Let the total size of the secret image be $r2 * c2 = m$.
Time taken to fill the *memoization lookup table* will be $k1 * 256 * n$, where $k1$ is constant.
Time taken to complete the rest of the process $k2 * (m - 256)$, where $k2$ is constant.
Hence total time taken is:

$$t = k1 * 256 * n + k2 * (m - 256)$$

Therefore required time will be,

$$O(k1 * 256 * n + k2 * (m - 256))$$

Or, $O(t) = O(n + m)$ (Eliminating constants).
Hence the entire process is done in linear time.

B. Decryption Algorithm

From the encrypted string first we obtain the reference hash table *(RHT)* using its respective AES-decryption algorithm. The last four entries of RHT will contain R_s, C_s, P_s and S_s of the secret image. These data will later be needed to extract the original secret image form the final image. Now from the *RHT* we can extract undistorted secret image by extracting the exact pixel values from the stored pixel positions in the *RHT* and if there exists some difference we can set the exact pixel value of the secret image since we know the difference between pixels.
To get the image in correct dimensions we use R_s, C_s and P_s which we obtained it earlier.

1) Pseudo-code For Decryption Algorithm: / Here cvrIm is the embedded cover image which we have used and enc is the encrypted file for the reference hash table */*

```
ImageDecryptor (cvrIm [ ][ ][ ], enc)
1. let rht be a new String
2. rht = AES-Decryption(enc)
3. s = rht [end]
4. rht = reshape(rht, s+2, 2)
/* This section extracts the secret image properties from RHT
*/
5. r = rht [end-1][1]
6. c = rht [end-1][2]
7. p = rht [end][1]
8. let extIm [1.s] be a new array for storing extracted Image
/* This section fills the extIm from RHT to get the secret
image*/
9. for i = 1 to s2
   9.1. extIm [i] = cvrIm[ rht [i][1] + rht [i][2] ]
10. end for
11. extIm = reshape(extIm,r,c,p)
// reshaping the matrix to get the image of three planes (RGB)
end
```

2) Time complexity analysis For Decryption Algorithm:
Let the size of the secret image be $r * c = n$. Time taken to get the secret image from the reference hash table $k * n$, where k is constant.

$$t = k * n$$

Therefore required time will be

$$O(k * n)$$

Or, $O(n)$ (Eliminating constants). Hence the entire process is done in linear time.

C. Dual Layer Encryption and Transmission

The RHT obtained after Encryption is first encrypted through a 128-bit AES Encryption [9]. The key obtained (primary key) is further encrypted through a 256-bit AES encryption [2]. The key hence obtained is called the secondary key. The RHT is encrypted using 128 bit AES encryption and not 256-bit because as the size of RHT increases the process becomes extremely slow. Hence an equivalent security component can be achieved through encrypting the resultant key of 128-bit AES encryption with 256-bit AES encryption.

Finally, the cover image, 128-bit AES encrypted RHT data, the 256-bit AES encrypted primary key and the secondary key are all transmitted through separate channels.

IV. Explanation

To understand the process let us assume that there is a 4x4 color secret image. *(Here the main text is the pixel value and super-scripted text is the pixel position.)*

TABLE III: Red Plane of 4x4 secret image

0^{0}	5^{4}	0^{8}	25^{12}
125^{1}	69^{5}	169^{9}	181^{13}
0^{2}	203^{6}	5^{10}	202^{14}
255^{3}	250^{7}	250^{11}	250^{15}

TABLE IV: Green Plane of 4x4 secret image

250^{16}	201^{20}	8^{24}	18^{28}
5^{17}	8^{21}	169^{25}	181^{29}
69^{18}	202^{22}	201^{26}	255^{30}
169^{19}	181^{23}	8^{27}	125^{31}

TABLE V: Blue Plane of 4x4 secret image

8^{32}	69^{36}	69^{40}	25^{44}
125^{33}	5^{37}	6^{41}	203^{45}
169^{34}	181^{38}	169^{42}	0^{46}
202^{35}	255^{39}	18^{43}	8^{47}

Let there be 3x4 RGB cover image, whose pixel representation is given below in the table VI

TABLE VI: Red Plane of 3x4 cover image

69^{0}	6^{3}	8^{6}	18^{9}
25^{1}	69^{4}	125^{7}	150^{10}
75^{2}	169^{5}	181^{8}	202^{11}

Here the size of our secret image is $3 * 4 * 4 = 48$ and the size of the cover image is $3 * 3 * 4 = 36$. Our intent is to map all the pixel values of the secret image with the cover image and store the pixel position of the closest pixel value in the cover image in Reference Hash Table *(RHT)*. Where the index of the *RHT* will be the pixel position of the secret image. To speed up the process we have introduced memoization look-up table *(MLT)*. It is cache-like structure where, the position of a particular pixel once searched in the cover image, it stores this position corresponding to the pixel value. Hence if later

TABLE VII: Green Plane of 3x4 cover image

250^{12}	255^{15}	69^{18}	202^{21}
75^{13}	25^{16}	125^{19}	250^{22}
0^{14}	18^{17}	8^{20}	150^{23}

TABLE VIII: Blue Plane of 3x4 cover image

6^{24}	8^{27}	6^{30}	69^{33}
75^{25}	150^{28}	202^{31}	250^{34}
18^{26}	25^{29}	169^{32}	0^{35}

same pixel value needs to be searched, its pixel position can be returned in constant time. Following example will make it clear. Our initial Reference Hash Table *(RHT)* will be like this presented in table IX

TABLE IX: Initial reference hash table

INDEX	0	1	.	.	47
PIXEL POSITION					
PIXEL DIFFERENCE					

Our initial Memoization Look-up Table will look like this following table X

TABLE X: Initial memoization lookup table

INDEX	0	1	2	3	.	.	.	.	255
PIXEL POSITION	0	0	0	.	.	.	.	.	0
PIXEL DIFFERENCE	0	0	0	.	.	.	.	.	0

Now we will match each pixel of the secret image to the closest value in the cover image. 1^{st} pixel of the secret image (1^{st} pixel in red plane) has pixel value of 0. We can see that this exists in 15^{th} pixel of the cover image (3^{rd} pixel in the green plane). Hence in the *RHT* for index 0 we store the value 14. (as each matrix is 0 indexed) with the pixel difference being 0 hence storing 0 in the pixel difference row. At the same time in *MLT* for index 0 we store the value 14 and in the pixel difference we store 0 as from now on any occurrences of the pixel value 0 can be mapped to cover image in constant time.

Now the 5^{th} pixel of the secret image has a pixel of 5. In the cover image there isn't any pixel of the same value but the closest value to it is 6 in the 4^{th} pixel. So in the *RHT* for index 4 we store 3 in the pixel position and in the pixel difference we store -1 $(5-6)$. Again in the *MLT* we store 3 in the pixel position and -1 in the pixel difference as from now on any occurrence of the same pixel can be mapped in constant time.

As we can see from the example we are successfully mapping the exact value of the secret image to the cover image without causing any distortion in the cover image or data loss of the secret image. We can also observe that this procedure is independent of the relative size of the images. Here we have mapped secret image of size 48 $(3*4*4)$ to the cover image of size 36 $(3*3*4)$.

So after the complete process our *RHT* will look like the table XI

And our final *MLT* will look like the table XII

TABLE XI: Final reference hash table

INDEX	0	1	2	.	.	.	45	46	47
PIXEL POSITION	14	7	14	.	.	.	11	14	6
PIXEL DIFFERENCE	0	0	0	.	.	.	1	0	0

TABLE XII: Final memoization lookup table

INDEX	0	1	.	.	201	.	.	254	255
PIXEL POSITION	14	-1	.	.	11	.	.	-1	15
PIXEL DIFFERENCE	0	-1	.	.	-1	.	.	-1	0

V. Experimental Results

For comparing our work with the other related works (as described in II) we have shown below an analysis of LSB planes of the original image, and LSBM, LSBMR embedded images shown in fig-2.

Fig-3 has been used to explain our scheme. The standard images used are Lena [13] as Cover Image and YF-17 [5] as Secret Image. The Secret Image is mapped with respect to the Cover and the RHT is obtained. This RHT is secured through a Dual Layer Encryption method and transmitted through separate channels as described in III-C. The Secret Image is obtained from the Cover Image and RHT using the decryption algorithm. The Extracted Secret Image obtained is also shown in fig-3. Beside the pictures are the histograms of their red, green and blue planes, which clearly depicts the independence of the secret image on the cover image.

Another major drawback of the LSB related techniques is that the data to be embedded has to be always less than the size of the LSB plane of the cover image. This drawback has been successfully eliminated by our approach which is clearly depicted when the YF-17 [5] image is mapped with respect to Lena [13].

*(**NOTE:** Image Dimensions: Lena [13] 128x128, YF-17 [5] 286x240)*

VI. Conclusion

The above experimental results indicate that the hybrid system combining steganography and cryptography establishes a secure method of data transfer. It is not entirely dependent on Security through Obscurity. The Dual Layer Encryption increases the data's security many folds. Furthermore, data transmission through separate channels add to the security. Hence our proposed method can be very effective in day to day data transfer and communication or could be clubbed with any other technology to make it more secure.

VII. Future Work

We would like to try and embed the encrypted Reference Hash Table itself into the cover image without much distortion in our future works. We would also try to extend this to text on text steganography, where we will try to hide a secret text of any length into a cover image of any length. Also we would build up a user friendly software that would directly implement our proposed method.

References

[1] William C Barker and Elaine B Barker. Sp 800-67 rev. 1. recommendation for the triple data encryption algorithm (tdea) block cipher. 2012.

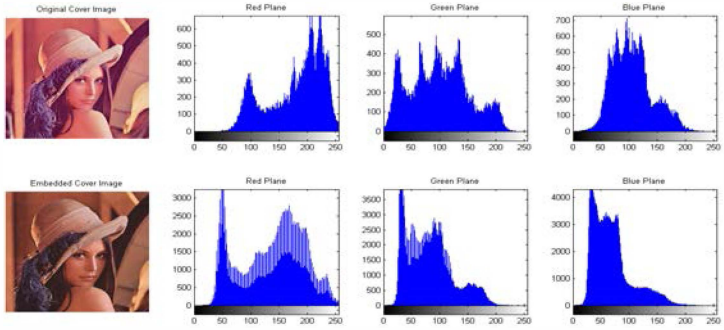

Fig. 1: RGB planes Histogram Comparison of LSB substituted Macbeth play [10] in the standard image Lena [13](512x512) before and after LSB substitution

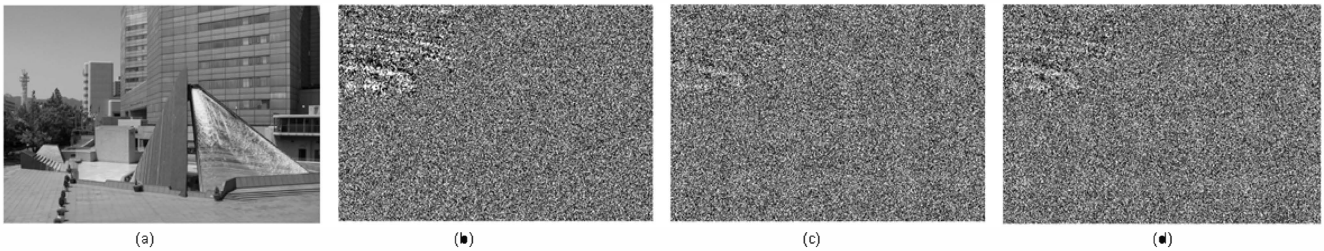

Fig. 2: (a) Cover Image [6] (b) LSB of Cover Image [6] (c) LSB of LSBM stego. [6] (d) LSB of LSBMR stego. [6]

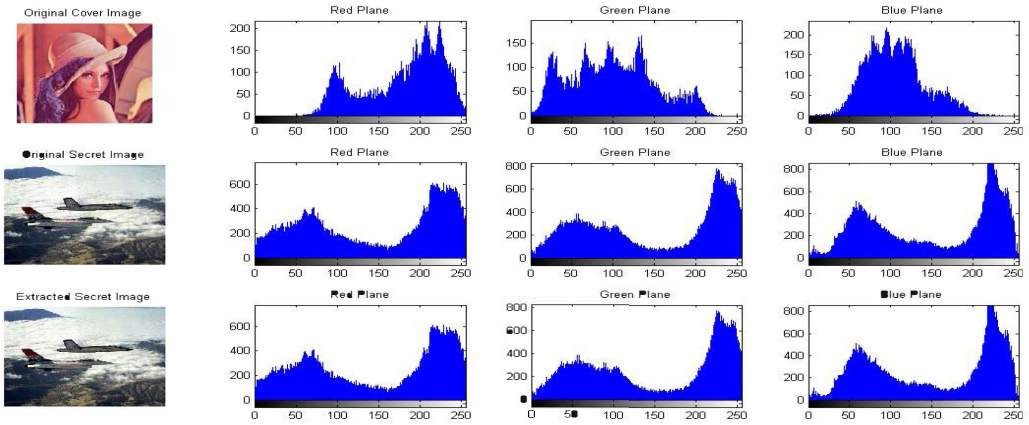

Fig. 3: RGB Histogram: Lena [13] v/s YF-17 [5]

[2] Alex Biryukov and Dmitry Khovratovich. Related-key cryptanalysis of the full aes-192 and aes-256. Cryptology ePrint Archive, Report 2009/317, 2009.

[3] Dan Boneh. Crypto 2003. 2003.

[4] Chi-Kwong Chan and Lee-Ming Cheng. Hiding data in images by simple lsb substitution. *Pattern recognition*, 37(3):469–474, 2004.

[5] Joseph W Lee, James F Meyers, Angelo A Cavone, and Karen E Suzuki. Do p p i e r gi o ba i ve loc i m et ry measurements of the vortical flow above an f/a-18. 1993.

[6] Weiqi Luo, Fangjun Huang, and Jiwu Huang. Edge adaptive image steganography based on lsb matching revisited. *Information Forensics and Security, IEEE Transactions on*, 5(2):201–214, 2010.

[7] Jarno Mielikainen. Lsb matching revisited. *Signal Processing Letters, IEEE*, 13(5):285–287, 2006.

[8] Goutam Paul and Imon Mukherjee. Image sterilization to prevent lsb-based steganographic transmission. *arXiv preprint arXiv:1012.5573*, 2010.

[9] NIST FIPS Pub. 197. *Announcing the Advanced Encryption Standard (AES)*, 2001.

[10] William Shakespeare. *Macbeth (Dover Thrift Editions)*. Dover Publications, unabridged edition, 12 1993.

[11] J. Michael Stewart. *Network Security, Firewalls And Vpns*.

[12] John R. Vacca, editor. *Network and System Security*. Syngress, 1 edition, 3 2010.

[13] Zhou Wang, Alan C Bovik, and BL Evan. Blind measurement of blocking artifacts in images. In *Image Processing, 2000. Proceedings. 2000 International Conference on*, volume 3, pages 981–984. Ieee, 2000.

[14] Elizabeth D. Zwicky, Simon Cooper, and D. Brent Chapman. *Building Internet Firewalls*. O'Reilly Media, second edition edition, 7 2000.