International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 6 (2018) pp. 3852-3860

A Novel Approach for Implementing the Bit Inversion Technique to Increase the PSNR of Stego Images and to Remove its Issues

Waseem Akram[#], Puneet Kumar[*]

[#] *Department of CSE, UIE, Chandigarh University, Mohali, Punjab, 140413, India.*

* *Department of CSE, UIE, Chandigarh University, Mohali, Punjab, 140413, India.*

Abstract

Bit Inversion Technique is applied to remove the modifications in the stego images that occur during the steganographic process. Its application results in the high-quality stego images with a lesser number of modifications. In this paper, the problems with the application of the Bit Inversion Technique are discussed and a potential solution is proposed to improve the implementation or the applicability of the bit inversion technique and also to increase the quality (PSNR) of the stego images. The proposed solution improves the implementation process of the bit inversion technique by introducing a preprocessing step that is applied to the stego images before the application of Bit Inversion Technique. In this preprocessing step, stego images are partitioned into a number of parts and then the Bit Inversion Technique is applied to each part. The introduction of this step not only increases the applicability of the bit inversion technique but also produces the high-quality stego images than the regular bit inversion application and this can be easily seen in the results provided in this paper. The results provided in this paper also confirm with our hypothesis which states that the quality (or PSNR) of the stego images is directly proportional to the partitions done to the stego images before applying the Bit Inversion Technique.

Keywords: Image Steganography; Bit Inversion; LSB Substitution; Stego Image; PSNR

INTRODUCTION

After the advent of the Internet as an information sharing and communication platform, the main concern of the researchers has been on providing the security to the data that moves across the internet. Thus, cryptography was brought into existence to provide the security to the data involved in communication through the internet. Different methods were invented to convert the data into a non-intelligent and non-readable form (encryption) so that the secrecy of the communication is confined only to the valid authorities, which know the methods to convert this non-intelligent data back into the sensible data (decryption). This encryption and decryption process was created to keep the shared information in secret form [1] [2]. But the advancements in the study of cryptography gave birth to a new type of study called cryptanalysis. Many cryptanalytic attacks were done by experts on the cryptosystems and on the encrypted data itself. Many of them were actually successful in breaking the cryptosystems and in intercepting the encrypted data [3]. Therefore, sometimes it is better to conceal the very existence of the communication in the first place. The technique that realizes this concept of keeping secret the communication of messages itself is called the steganography [2] [4].

Steganography is the science of concealing the one kind of data (video, or image, or message, or a file) within the other or same kind of data (video, or image, or message, or a file). In other words, it is the art and science of obscure communication. The literal meaning of steganography is "*cover writing*" which is derived from the two root words from Greek "*steganos*" which means, "*covered, concealed, obscured or protected*" and "*graphein*" which means, "*writing*". In image steganography, the data is concealed inside the different types of digital images [4] [5] [6] [7].

Steganography is not a novel concept it has been in practice since distant past. For example, the first recorded application of steganography technique can be found around 440 BC when *Histiaeus* tattooed the message onto the scalp of the trusted servant after shaving his head and then after his hair had grown back he sent him out to deliver the message to his vassal, *Aristagoras*. Other types of techniques used in ancient times include writing hidden messages within wax tablets, use of secret inks, embedding of messages inside poems etc. [2] [6] [7]

Image Steganography is a technique through which we can embed the secret messages inside the images by adjusting the pixels values or changing some other parameters of the image. Various terms that are used in Image Steganography are [2]:

- **Cover Image:** An image that is a carrier of secret information.
- **Stego Image:** When the secret message is embedded into the cover image, the resulting image is called a stego image.
- **Message:** Message represents the original data that is to be hidden inside the cover image.
- **Stego Key:** Some steganographic algorithms may require you to use a key for embedding the data inside the images and the same key is used for retrieval.

Image steganography techniques are divided into two categories depending on the type of data of a cover image that is manipulated to hide the secret information [2] [5] [8].

Image Domain or Spatial Domain: In this technique, data or information is embedded directly into the pixel values (or pixel intensities). We directly manipulate the pixel values. These techniques are typically dependent on the image formats to be used as a cover. These techniques are easy to apply due to their simplicity and are often characterized as "*simple systems"*. In image or spatial domain, data in the cover images can be lost due to the exposure to cropping, or compression.

Transform Domain or Frequency domain: In this technique, a cover image is transformed into the frequency domain using different transforming techniques and then secret information is embedded or inserted in the significant areas of the transformed image. Transform domain is better than spatial domain because transform domain hides the information into the frequency (or transform) coefficients of the cover images that are less exposed to cropping, or compression.

LITERATURE SURVEY

In this section, we shall be presenting the literature survey of LSB substitution techniques.

Simple LSB substitution: In this method, the message bits are directly inserted into the LSBs of the pixels of the cover image in a sequential fashion. This technique provides the large hiding capacity and it is very easy to implement [6] [9] [10] [11].

Random LSB substitution: Simple LSB substitution hides the information in the cover pixels in a sequential manner. This makes the information easy to detect and extract. The Random LSB substitution techniques use the PRNGs (Pseudo Random Number Generators) to embed the information into the pixels in a random fashion. The PRNG should be seeded with some key that is shared between the encoder and decoder. In this way, the message bits are spread randomly over the stego-image. Thus, it provides more security than the simple LSB substitution technique. [10] [11]

PVD Technique: In this technique, the cover image is partitioned into non-overlapping blocks (or pairs) of two consecutive pixels. The difference of the pixel values in each pair is calculated. On the basis of that difference, each pixel pair is associated with (or categorized into) a different number of ranges (or categories) of smoothness and contrast properties. Then, the data is embedded in each pixel pair and the amount of data embedded depends on the width of the range that the difference of pixels in a pair belongs to. [12]

Edge-Based Technique: The changes made inside the smooth areas of the images may be visible to open eye. To overcome this problem, edge-based techniques were proposed by the researchers. In this, maximum modifications are done on edge pixels and lesser on non-edge pixels. In [13], authors have used a hybrid edge detector (combination of canny and fuzzy detectors) to find the edge pixels to embed the data. Three bits are embedded in edge pixels and only one bit is embedded in non-edge pixels. All operations are done on grayscale images. Similarly, in [14] authors have used two different hybrid detectors, first is Sobel ORed Fuzzy and second is Laplacian ORed Fuzzy detector. They have compared the results with that of [13] but they are using the RGB images in their method.

LSB Matching Technique: The LSB substitution methods create randomness in the cover images, this changes the statistical properties of the cover images and causes asymmetry imbalance i.e. even values are always increased and odd values are always decreased. These changes become the footprint for the detection of the hidden information. Steganalysis methods like RS-analysis [15] and Histogram attack [16] take advantage of these problems. The LSB matching technique was introduced to eliminate or reduce the asymmetry imbalance thus improving undetectability. In this technique, if the message bit does not match the LSB of cover image pixel then one is added or subtracted randomly from the cover pixel. This ensures that the modified pixels will always have the same probability of increasing or decreasing. [17]

Bit Inversion technique: In [18] [19], authors proposed a novel Bit Inversion scheme which is a general scheme that can be applied after any LSB substitution technique. In this scheme, a stego image is analyzed for particular patterns in some LSBs and then modified and unmodified bits (or pixels) are counted. And based on some rules, some of the LSBs (or pixels) that were modified are inverted back to the original. This scheme improves the quality and the security of a stego image. It also reduces the asymmetry imbalance in the stego images that occurs because of the LSB substitution. This, in turn, decreases the detectability of the stegno image by the statistical steganalysis algorithms like [15] and [16].

In this paper, the problems with the application of the Bit Inversion Technique are discussed and a potential solution is proposed to improve the implementation or the applicability of the bit inversion technique to the stego images and also to increase the PSNR (Peak Signal to Noise Ratio) values of stego images. The stego images are preprocessed before applying the bit inversion technique. As a result of the preprocessing step that is applied before implementing the bit inversion technique, the quality (PSNR) and the security of the stegno image is further improved in addition to the increased applicability.

BIT INVERSION TECHNIQUE

To understand the bit inversion algorithm that is depicted in Fig. 1, the following example is considered.

Let us consider that eight message bits, 01001011, that are to be hidden in the following image pixels (cover pixels):

Pixel 1-3 11000101, 10010111, 10010101

Pixel 4-6 10110011, 10011100, 10100100

Pixel 7-8 00001100, 10010100

After hiding the eight message bits using simple LSB substitution, the updated pixels (stego pixels) would be:

Pixel 1-3 1100010**0**, 10010111, 1001010**0**

Pixel 4-6 1011001**0**, 1001110**1**, 10100100

Pixel 7-8 0000110**1**, 1001010**1**

The pixels with the bold LSBs are the modified pixels. These modifications decrease the PSNR of the stego image and this is where the bit inversion technique comes to rescue. It decreases the number of modification in the stego image.

When we follow the first two steps of the algorithm, we get the results as shown in Table I. Here, *bitPat* represents the 2nd and 3rd LSBs of stego pixels (or cover pixels) which are shown underlined. And the number of modified and unmodified pixels is calculated for each *bitPat* by comparing the stego pixels with cover (or original) pixels.

And after applying the 3rd step, we get the results, as shown in Table II, by inverting the LSBs of all the stego pixels which contain the *bitPats* with *Mod* > *Unmod.* The updated stego pixels after the 3rd step would be:

Pixel 1-3 11000101, 10010111, 10010101

Pixel 4-6 10110011, 10011100, 1010010**1**

Pixel 7-8 00001100, 10010100

So, we can see how the results get reversed after applying the bit inversion technique. Initially, we had six modified pixels but after the application of bit inversion, we have just one modified pixel. Thus, this results in a less modified and high-quality stego image.

Table I. Bit Inversion Decisions

bitPat	Modified Pixels (Mod)	Unmodified Pixels (Unmod)	Mod > Unmod
‘00’	0	0	No
‘01’	1	0	Yes
‘10’	5	1	Yes
‘11’	0	1	No
Total	**6**	**2**	**--**

Table II. Results after Bit Inversion

bitPat	Modified Pixels (Mod)	Unmodified Pixels (Unmod)
‘00’	0	0
‘01’	0	1
‘10’	1	5
‘11’	0	1
Total	**1**	**7**

Algorithm: Bit Inversion

Input: Stego-Image *S*, Cover Image *C*.
Pattern: {00, 01, 10, 11}
Mod: Modified bits (or pixels)
Unmod: Unmodified bits/pixels
Output: Improved Stego-Image *S'*
BEGIN:
1: Check 2nd and 3rd LSB pattern of every pixel in stego image, *S.* Let this be called as *bitPat.*
2: **for each** *bitPat* ∈ *Pattern* **do**
*/*calculate the number of modified and unmodified bits (or pixels) for each bitPat in the stego image, S, by comparing with the original cover image pixels*/*
Mod=calculateMod (*bitPat*)
Unmod=calculateUnmod (*bitPat*)
end for
3: **for each** *bitPat* ∈ *Pattern* **do**
if *Mod* > *Unmod* **then**
invert the LSB of every pixel of stego image, *S*, with the *bitPat* fulfilling the above condition.
end if
end for
END

Note: *For simplicity, we have separated the steps 2 and 3. But they can be combined as a one step.*

Figure 1. Bit Inversion Algorithm

PROBLEMS IN APPLICABILITY OF BIT INVERSION AND THE PROPOSED SOLUTION

The problem with the application of the bit inversion technique is the fact that *bigger the stego image, lesser the chances of modified pixels being greater than unmodified pixels*. This phenomenon sometimes *causes the inapplicability* of the bit inversion technique to some of the stego images. And when it is applicable, it *does not give* the best results.

A potential solution to the above problems is proposed and implemented in this paper. It comprises of three steps. In 1st step, we divide or partition the stego images into a number of parts. In 2nd step, we apply the bit inversion technique to each part of the stego image independently. In 3rd step, we combine the parts to get the final improved stego image. Smaller parts have higher chances of “modified pixels being greater than unmodified pixels” therefore higher chances of applicability of bit inversion technique and also the higher pixel benefit. *It can be seen in the results presented in this paper that the pixel benefit (hence PSNR) is directly proportional to the number of partitions made to the stego images.*

To understand this, let us take the example of a 10x10 stego image (Fig. 2 (a)) that is produced by hiding some data in it. In this image, let us consider black spots as the modified pixels. This example is just for the understanding purpose.

If we consider the whole image for bit inversion then we have 40 modified pixels and 60 unmodified pixels (assuming all pixels have same 2nd and 3rd LSB pattern). Then, according to the 3rd step of bit inversion algorithm presented in previous section, we cannot invert the pixels. Therefore, we will not get any benefit by using bit inversion technique.

Now if we partition the above image into two 5x10 parts (Fig. 2 (b)) and apply the bit inversion on both of them we will get better results. For part one, we have 30 modified and 20 unmodified pixels thus bit inversion can be applied and we can get the benefit of 10 pixels. For part two, we have 10 modified and 40 unmodified pixels thus we cannot apply bit inversion on this part.

Similarly, if we partition the image into four 5x5 parts then we will have the parts as shown in Fig. 2 (c) Now apply the bit inversion to every part. For part one, we have 23 modified and 2 unmodified pixels. For part two, we have 7 modified and 18 unmodified pixels. For part three, we have 5 modified and 20 unmodified pixels.

And for part four, we have 5 modified and 20 unmodified pixels. So, we can apply bit inversion only on part one but we will get total benefit of 21 pixels. This benefit is two times more than the benefit provided by dividing the stego image into two parts. Thus, it can be induced that more partitions result in more benefit, see Table III. The flow diagram of the proposed solution is depicted in Fig. 3.

Table III. Applicability of Bit Inversion Technique

Image	Bit Inversion	Benefit (In Pixels)
Full	Not Applicable	0
Partitioned into 2	Applicable	10
Partitioned into 4	Applicable	21

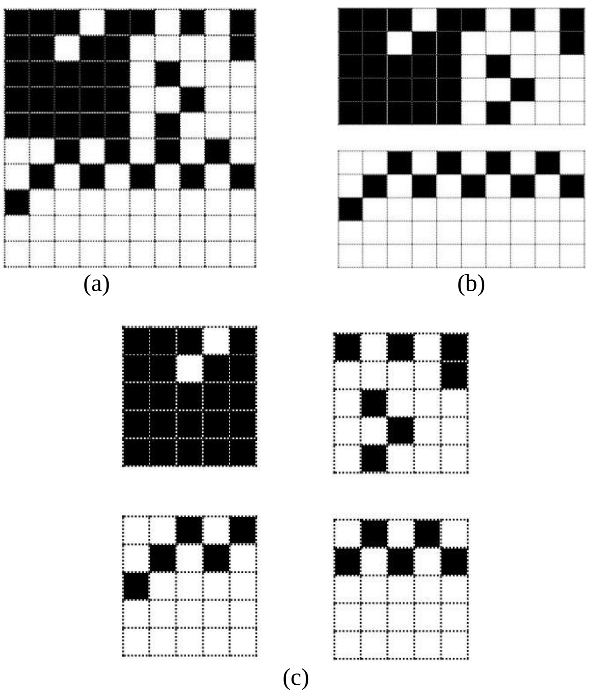

Figure 2 (a) 10x10 Stego Image (b) Two 5x10 Parts (c) Four 5x5 Parts

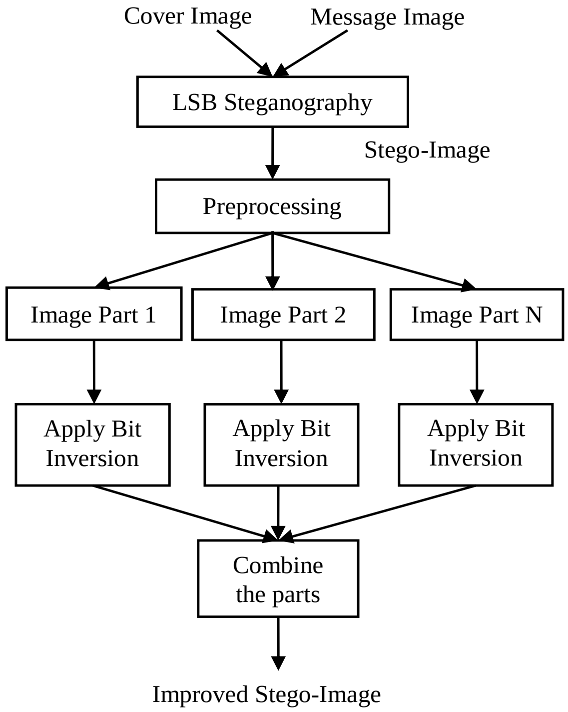

Figure 3. Flow Diagram of the Proposed Solution

EXPERIMENTAL SETUP

We use MATLAB R2016a for the implementation of our proposed work. We use four cover images and seven message images to test the performance of our technique. Message images are embedded into the cover images to generate the stego images that are to be tested for improvements. Images are taken from the UCI image database and the image database of a book named "*Digital Image Processing, 3rd ed"*, by Gonzalez and Woods. Four cover images are Test Pattern (1024*1024) [20], Building (1114*834) [21], Katrina (1281*1153) [22], and Baboon (512*512) [20]. They are shown in Fig. 4 (a-d) respectively. Seven message images are Clock (256*256) [20], Plane (256*256) [20], Plane (180*180) [20] Pattern (256*256) [20], Pattern (180*180) [20], Cygnus Loop Xray (300*300) [22], and Crab Pulsar Xray [22]. They are shown in Fig. 4 (e-i) respectively. Grayscale Baboon (512*512) is produced from 24-bit colour Baboon using Matlab function rgb2gray. And Pattern (180*180) is produced from Pattern (256*256) using the Matlab function, imresize.

In our implementation, we first convert the message image into a bitstream and use the simple LSB substitution (SLSBS) technique to hide the bits into the cover image. Then, we use various partitioning schemes to partition the stego images into a number of parts. The partitioning schemes we use are 1-part, 4-part, 16-part, and 64-part schemes and they partition the stego-image into 2 parts, 4 parts, 16 parts, and 64 parts respectively. Each partitioning scheme is implemented independently of others. And bit inversion is applied to each part in each scheme. And then, the results from each scheme are calculated independently.

In the 1-part scheme, we partition the stego-image into two parts by dividing its row-size by 2. Then we take its floor value as the last row of the first part and the floor value plus one as the first row of the second part of the stego-image. And if we consider only first part for bit inversion then we have a 1-part scheme. In the 4-part scheme, we partition the stego-image into four parts by dividing both row-size and column-size by 2. And we take their floor values as the last row and the last column of the first part of a stego image. And similarly, we take other parts of the image from these values. For partitioning into sixteen parts, we first partition the image into four parts as explained above then each part is further partitioned into four parts. Thus, it gives a total of sixteen parts. We further partition the sixteen parts into four parts each thus giving a total of sixty-four parts of the image.

Also, we use first twenty bits of stego-image as a header to store the dimensions of the message image that is hidden inside it. First ten bits are for the row-size and next ten bits for the column-size. We also need to store the bit inversion decisions for each part of the stego image and for each pattern (00, 01, 10, and 11). This header information is extracted during the decoding process to correctly retrieve the message image from the stego image.

Thus, for a single part of the image, we use four bits to store the bit inversion decisions for four-bit patterns ('00', '01', '10', '11') of 2nd and 3rd LSBs of pixels. The first bit will tell whether the bit inversion has been done on pixels with the pattern '00', second bit will tell about '01', the third bit will tell about '10' and the fourth bit will tell about '11'. This information is stored in the end pixels of the stego-image as a trailer. So, for one partition we need a trailer of four bits that are stored in the LSBs of the last four pixels of the stego-image, one bit per pixel. For sixteen parts, we need a trailer of 64 bits that is stored in last 64 pixels of the stego-image. And similarly, for sixty-four parts, we need a trailer of 256 bits that is stored in the last 256 pixels of the stego image. This trailer information is extracted during the decoding process to revert back the changes done to the stego images during the encoding process to correctly retrieve the message image from the stego image.

RESULTS AND ANALYSIS

In Table IV and Table V, "0-parts" means that the stego images are not partitioned and bit inversion is applied to the whole image. And it is clear from the results that bit inversion was not applicable and was not able to produce any pixel benefit when applied to the unpartitioned stego images. Our solution takes into account the smaller parts of the image for the improvements that can be easily seen in the results.

Table IV shows the results produced by simple LSB substitution (SLSBS) and all partitioning schemes when a message image, "pattern", is embedded inside the cover images, "test pattern" and "house". For the cover image, "test pattern", and the message image, "pattern", the number of modified pixels after Simple LSB Substitution (SLSBS) is 3,21,805 and the PSNR value is 53.2069. But after applying the bit inversion to the upper part of the stego image (i.e. 1-part scheme), the number of modified pixels get reduced to 1,75,086. Thus, giving a benefit of 1,46,719 pixels and an improvement in PSNR by 2.6434 units when compared to SLSBS method. As we keep partitioning the stego image into a number of parts and applying bit inversion to each part, the number of modified pixels gets further reduced. We can see in a 64-part scheme that the modified pixels are further reduced to 1,04,520. Therefore, it gives a benefit of 2,17,285 pixels and an increase in PSNR by 4.8839 units when compared to SLSBS method. And when compared to the 1-part scheme, we have a benefit of 70,566 pixels and an increase in PSNR by 2.2405 units. Same can be observed when the cover image used is "House" as shown in Table IV.

Similarly, Table V shows the results produced when message image, "Cygnus loop xray", is embedded into the cover images, "test pattern" and "Katrina". With the cover image "Katrina", we can notice that the number of modified pixels by the 1-part scheme is 3,19,325 and by the 4-part scheme, it is 3,19,329. It can be seen that the modified pixels increase after 4-part scheme that normally should not happen. This anomaly can be explained by the fact that 1-part scheme requires a 4-bit information trailer at the end of the stego image and 4-part scheme requires a 16-bit information trailer. And since the 4-part scheme was unable to produce more benefit than the 1-part scheme, it should have produced the same benefit as 1-part scheme. But due to this 16-bit trailer that is inserted into the last pixels of the stego image, 4 more pixels were modified by 4-part scheme than 1-part scheme.

Table VI shows the pixel benefits and PSNRs produced by simple LSB substitution, 1-Part, and 64-Part bit inversion schemes for the number of cover images and messages images that are shown in Fig. 4.

The results provided in table IV, V, VI confirm with our hypothesis which states that the quality (or PSNR) of the stego images is directly proportional to the partitions done to the stego images before applying the Bit Inversion Technique all partitions. As we can readily see from the results that the stego images produced by 64-part scheme have higher PSNR values than any other scheme. Also, stego images produced by the 64-part scheme are closer to their respective cover images than any other scheme hence more secure. (Fig. 5 and Fig. 6)

Fig. 5 and Fig. 6 shows the graphical comparison of histograms of stego images, produced by various schemes, with their cover images. We can readily see that the stego images produced by the 64-Part scheme are better than 1-Part scheme and 64-Part scheme produces stego-images with the histogram that are closer to their cover images. Therefore, it can be induced that the applicability of the bit inversion technique and the quality of stego images increases as the partitions increase.

CONCLUSION

The proposed solution is basically a post-steganographic process applied to the stego-images to improve the applicability of the bit inversion technique to the stego images and also to increase their quality (PSNR). And with the help of the results obtained by extensive experiments, we conclude that our hypothesis is indeed true. Which states that the quality (or PSNR) of the stego images is directly proportional to the partitions done to the stego images before applying the Bit Inversion Technique all partitions.

The main idea of the proposed work is to partition the stego images into a number of parts and applying Bit-Inversion to each part independently and then combining the parts back together to get a final improved stego image. This technique leads to the improved applicability of the bit inversion technique, lesser number of modified pixels and better PSNR values of stego images.

The proposed work considers only the grayscale images as cover images and message images and it considers simple LSB substitution for the steganographic process. In future, the work presented in this paper can be extended to the 24-bit colour images as well as to other steganographic methods. Also, the future work can be based on finding the clusters of modified pixels in the stego-images and applying the Bit-Inversion to those clusters rather than dividing the whole stego-image into a number of partitions.

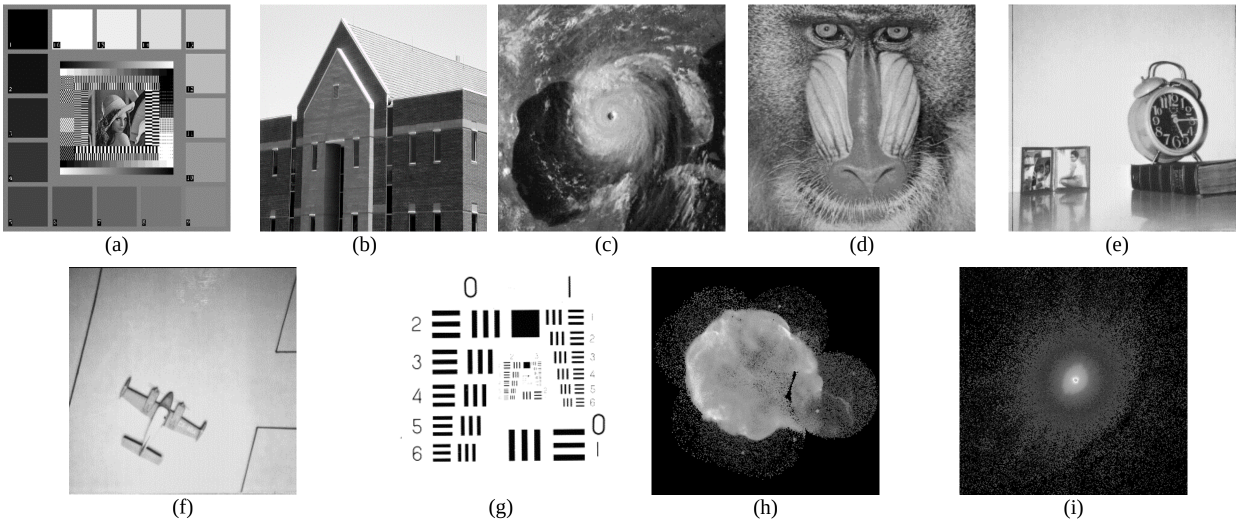

Figure 4. Cover Images: (a) Test Pattern (b) House/building (c) Katrina (d) Baboon/Mandrill, Message Images: (e) Clock (f) Plane (g) Pattern (h) Cygnus Loop Xray (i) Crab Pulsar Xray.

Table IV. Statistics For 'Pattern' Message Image

Message Image: Pattern (256*256)				
Cover Image: Test Pattern (1024*1024) **After SLSBS:** Modified Pixels = 321805, Unmodified Pixels = 726771, PSNR = 53.2609				
After Bit Inversion				
Parts	**Modified Pixels**	**Unmodified Pixels**	**Pixel Benefit**	**PSNR**
0	Same as SLSBS and Bit Inversion Inapplicable			
1	175086	873490	146719	55.9043
4	160412	888164	161393	56.2844
16	133547	915029	188258	57.0805
64	104520	944056	217285	58.1448
Cover Image: House (1114*834) **After SLSB:** Modified Pixels = 293243, Unmodified Pixels = 635833, PSNR = 53.1390				
0	Same as SLSBS and Bit Inversion Inapplicable			
1	209740	719336	83503	54.5945
4	208577	720499	84666	54.6187
16	196722	732354	96521	54.8728
64	190132	738944	103111	55.0208

Table V. Statistics For 'Cygnus Loop' Message Image

Message Image: Cygnus Loop Xray (300*300)				
Cover Image: Test Pattern (1024*1024) **After SLSBS:** Modified Pixels = 303961, Unmodified Pixels = 744615, PSNR = 53.5086				
After Bit Inversion				
Parts	**Modified Pixels**	**Unmodified Pixels**	**Pixel Benefit**	**PSNR**
0	Same as SLSBS and Bit Inversion Inapplicable			
1	286455	762121	17506	53.7662
4	274444	774132	29517	53.9523
16	236794	811782	67167	54.5931
64	213126	835450	90835	55.0504
Cover Image: Katrina (1281*1153) **After SLSB:** Modified Pixels = 378717, Unmodified Pixels = 1098276, PSNR = 54.0414				
0	Same as SLSBS and Bit Inversion Inapplicable			
1	319325	1157668	59392	54.7823
4	319329	1157664	59388	54.7822
16	305562	1171431	73155	54.9736
64	287471	1189522	91246	55.2386

Table VI. Pixel Benefit And PSNR For Message Images

Cover	**Message**	**SLSBS**	**1-Part Scheme**		**64-Part Scheme**	
		PSNR	**Pixel Benefit**	**PSNR**	**Pixel Benefit**	**PSNR**
Test Pattern (1024 * 1024)	Pattern (256*256)	53.2609	146719	55.9043	217285	58.1448
	Clock (256*256)	53.8570	41304	54.5488	72712	55.1600
	Plane (256*256)	53.9689	24705	54.3802	64788	55.1435
	Cygnus Loop Xray (300 * 300)	53.5086	17506	53.7662	90835	55.0504
	Crab Pulsar Xray (300 *300)	53.5292	18027	53.7960	90530	55.0736
House/Building (1114 * 834)	Pattern (256*256)	53.1390	83503	54.5945	103111	55.0208
	Clock (256*256)	53.3300	38074	53.9633	45313	54.0949
	Plane (256*256)	53.3303	38218	53.9662	46516	54.1174
	Cygnus Loop Xray (300 * 300)	52.6222	14885	52.8225	37578	53.1467
	Crab Pulsar Xray (300 * 300)	52.6872	14629	52.8870	36471	53.2035
Katrina (1281 * 1153)	Pattern (256*256)	55.7399	15618	56.0131	52739	56.7411
	Clock (256*256)	55.6943	0	55.6943	17190	55.9928
	Plane (256*256)	55.6975	0	55.6975	17227	55.9977
	Cygnus Loop Xray (300 * 300)	54.0414	59392	54.7823	91246	55.2386
	Crab Pulsar Xray (300 * 300)	54.0380	59950	54.7858	91386	55.2363
Baboon (512*512)	Pattern (180*180)	51.1869	434	51.2015	2844	51.2832
	Plane (180*180)	51.1872	568	51.2062	2865	51.2842

International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 6 (2018) pp. 3852-3860
© Research India Publications. http://www.ripublication.com

	Cover Image: Baboon; **Message Image:** Pattern (180*180)	**Cover Image:** Baboon; **Message Image:** Plane (180*180)
1-Part-Scheme		
64-Part-Scheme		

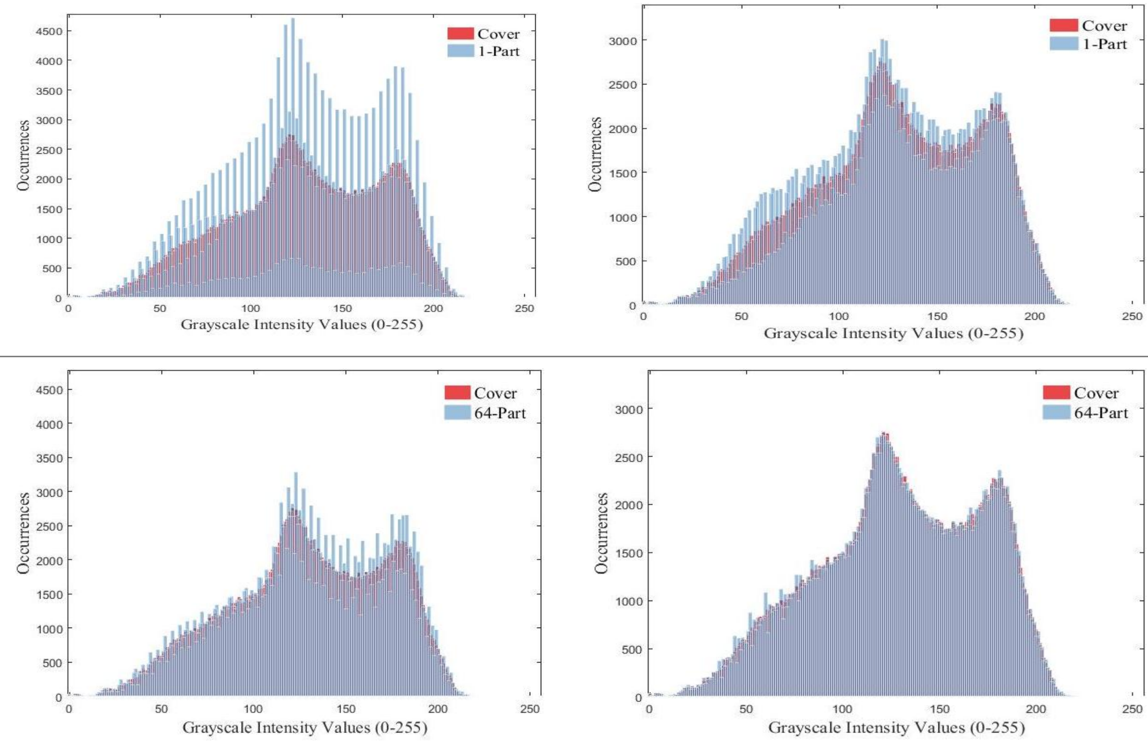

Figure 5. Histograms of Stego Images Produced by Various Schemes

	Cover Image: House; **Message Image:** Pattern (256*256)	**Cover Image:** Test Pattern; **Message Image:** Pattern (256*256)
1-Part-Scheme		
64-Part-Scheme		

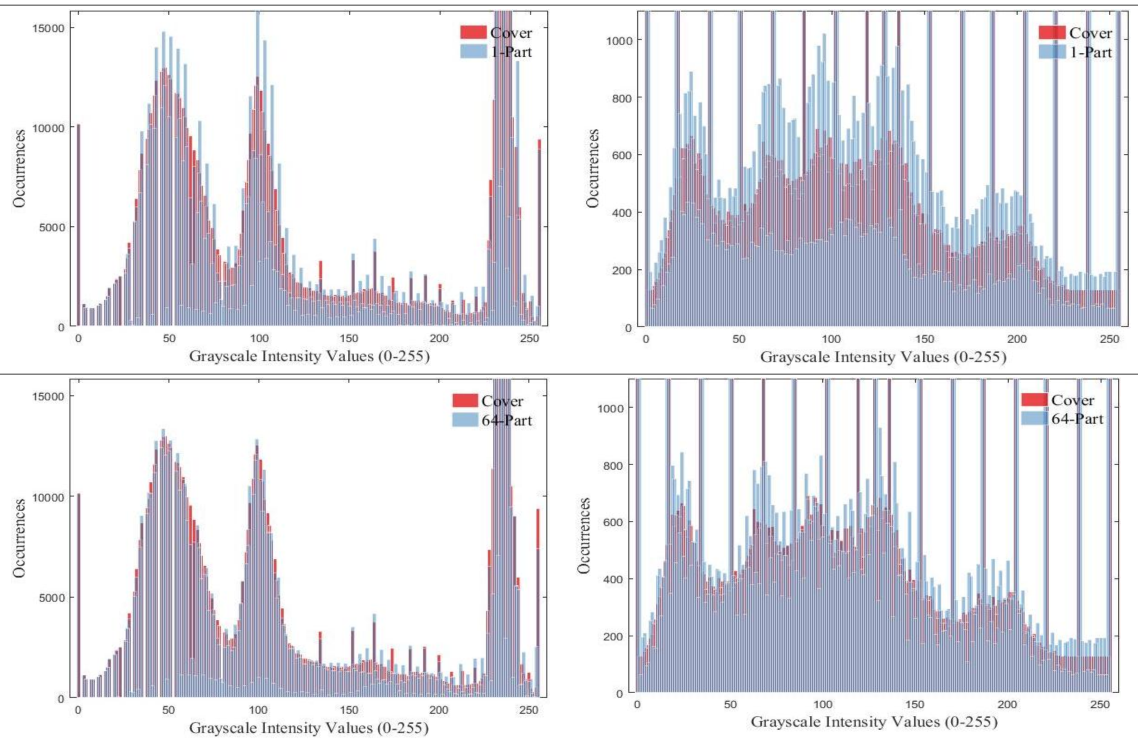

Figure 6. Histograms of Stego Images Produced by Various Schemes

REFERENCES

[1] J. S. Coron, “What is cryptography?”, *IEEE Security & Privacy*, vol. 4, no. 1, pp. 70-73, Jan.-Feb. 2006.

[2] Nagham Hamid, Abid Yahya, R Badlishah Ahmad, Osamah M Al-Qershi, “Image steganography techniques: an overview”, *International Journal of Computer Science and Security (IJCSS)*, vol. 6, no. 3, pp. 168-187, 2012.

[3] E. F. Brickell and A. M. Odlyzko, “Cryptanalysis: a survey of recent results”, *Proceedings of the IEEE*, vol. 76, no. 5, pp. 578-593, May 1988.

[4] J. Fridrich, R. Du, L. Meng, “Steganalysis of LSB Encoding in Color Images”, *Proc. IEEE Int'l Conf. Multimedia and Expo*, 2000.

[5] Cheddad, J. Condell, K. Curran, P. M. Kevitt, “Digital image steganography: Survey and analysis of current methods”, *Signal Process.*, vol. 90, no. 3, pp. 727-752, Mar. 2010.

[6] N. Johnson, S. Jajodia, “Exploring Steganography: Seeing the unseen”, *IEEE Computer*, vol. 31, pp. 26-34, 1998.

[7] “Steganography”, *En.wikipedia.org*, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Steganography. [Accessed: 11- Oct- 2017].

[8] T. Morkel, J.H.P. Eloff, M.S. Olivier, “An overview of image steganography”, http://mo.co.za/open /stegoverview.pdf, Accessed on October 2017.

[9] C. K. Chan, L. M. Chen, "Hiding data in images by simple LSB substitution", *Pattern Recognition.*, vol. 37, no. 3, pp. 469-474, 2004.

[10] S. Venkatraman, A. Abraham, M. Paprzycki, “Significance of Steganography on Data Security”, *International Conference on Information Technology: Coding and Computing. ITCC 2004*, vol. 2, pp. 347-351, 5–7 Apr 2004.

[11] R. Jain and J. Boaddh, “Advances in digital image steganography”, *2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH)*, Noida, 2016, pp. 163-171. doi: 10.1109/ICICCS.2016.7542298

[12] D. C. Wu and W. H. Tsai, “A steganographic method for images by pixel-value differencing,” Pattern Recognition Letter, vol. 24, no. 9-10, pp. 1613-1626, 2003.

[13] W.-J. Chen, C.-C. Chang, T. H. N. Le, “High payload steganography mechanism using hybrid edge detector”, *Expert Syst. Appl.*, vol. 37, no. 4, pp. 3292-3301, Apr. 2010.

[14] Anastasia Ioannidoua, Spyros T. Halkidisb, George Stephanidesb, "A novel technique for image steganography based on a high payload method and edge detection", Expert Systems with Applications, Vol. 39, no. 14, pp. 11517-11524, 2012.

[15] J. Fridrich, M. Goljan, R. Du, “Reliable detection of LSB steganography in color and grayscale images”, *Proc. ACM Workshop Multimedia Security*, pp. 27-30, 2001-Oct.-5.

[16] X. Zhang, S. Wang, “Vulnerability of pixel-value differencing steganography to histogram analysis and modification for enhanced security”, *Pattern Recognit. Lett.*, vol. 25, pp. 331-339, 2004.

[17] G. L. Smitha, E. Baburaj, “A survey on image steganography based on Least Significant bit Matched Revisited (LSBMR) algorithm”, 2016 International Conference on Emerging Technological Trends [ICETT], 2016.

[18] N Akhtar, Shahbaaz Khan, Pragati Johri, “An Improved Inverted LSB Image Steganography”, IEEE International Conference on Issues and challenges in Intelligent Computing techniques (ICICT), 7-8 February 2014, India.

[19] N Akhtar, “An LSB Substitution with Bit Inversion Steganography Method”, 3rd International Conference on Advanced Computing, Networking, and Informatics (ICACNI - 2015), 23 - 25 June 2015, Bhubneswar, Orissa, India.

[20] The USC-SIPI Image database http://sipi.usc.edu/database/

[21] *Imageprocessingplace.com*. [Online]. Available: http://www.imageprocessingplace.com/downloads_V3/dip3e_downloas/dip3e_book_images/DIP3E_CH10_Original_Images.zip. [Accessed: 25- Nov- 2017].

[22] *Imageprocessingplace.com*, 2017. [Online]. Available: http://www.imageprocessingplace.com /downloads_V3/dip3e_downloas/dip3e_book_image s/DIP3E_CH01_Original_Images.zip. [Accessed: 25- Nov- 2017].