ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology
Volume 2, Issue 2, February 2013

A Comparative Study On Some New Steganographic Techniques

H. Faheem Ahmed and U. Rizwan

Abstract — In this paper, we present some new methods and develop techniques with algorithms for hiding text behind the gray scale image. Steganographic technique allows the sender to communicate information to the receiver without a third party even knowing that some communication is occurring. First, we embed a 512-bytes of text message in a 512 by 512 gray level image in each of the bit planes from first to eighth bit. We have also given some new techniques of embedding a text message in the entire gray level of the image itself, instead of bit levels, like hiding the text message in memory locations of image pixel values where the gray level is most repeated in column wise, row wise, abundant and odd abundant memory locations, equal / near equal to the gray level column wise, and entire image and at random locations. A comparison study is also carried out.

Keywords: Steganography, Abundant Numbers,

1. INTRODUCTION

Information in the form of files becomes the norm in the current era of globalization. Many of these files are confidential and very important for other parties unknown. Steganography is the art and science which studies how withholding information on a medium in such a way that its presence is not detected by other parties who are not entitled to that information. Views of human needs on a picture or image that cannot be separated from human life, steganography is made to conceal the message in the form of text in the picture. Because humans are usually less sensitive to the form of text messages that are not easily visible, steganography cannot be easily detected. Recipient can extract the secret information in it.

Steganography is the hiding of information in such a way that onlookers do not suspect information is being hidden. In contrast, cryptography is explicitly hiding information known to be present and even typically publishing the encrypted message and the entire method apart from the key. Steganography can consist of altering a photograph or sound or other such item in such a way that casual observance of the item will not even arouse suspicion. So many documents, pictures, sound files etc. are posted over the internet or sent over email, that any one of them could easily contain a hidden message.

2. LSB STEGANOGRAPHIC TECHNIQUES IN DIFFERENT PLANES

One of the techniques used in steganography to hide data behind images is called the least-significant bit (LSB) insertion wherein the LSB of each byte of the pixel of the image's data is replaced with the single bit of the data to be hidden. This is based on the premise that the total number of bit-changes in the image's data will be so small that the resulting stego-image will be indistinguishable to the human eye from the original image. LSB steganography that replaces the least significant bits of the host medium is a widely used technique with low computational complexity and high insertion capacity. The last few bits in a color byte, however, do not hold as much significance as the first few. This is to say that two bytes that only differ in the last few bits can represent two colors that are virtually indistinguishable to the human eye. For example, 00100110 and 00100111 can be two different shades of red, but since it is only the last bit that differs between the two, it is impossible to see the color difference. LSB steganography, then, alters these last bits by hiding a message within them.

A large number of commercial steganographic programs use the least significant bit embedding as the method of choice for message hiding in 24-bit, 8-bit color images, and gray scale images respectively. It is commonly believed that changes to the LSBs of colors cannot be detected due to noise that is always present in digital images.

The least significant bit, that is, the eighth bit inside an image is changed to a bit of the secret message. When using a 24-bit image, one can store 3 bits in each pixel by changing a bit of each of the red, green and blue color components, since they are each represented by a byte.

The Peak Signal to Noise Ratio (PSNR), Mean Square Error (MSE) are performance parameters to measure the quality of image.

- MSE: It is defined as square of error between cover stego-image. The error indicates the distortion in an image.MSE can be calculated by using two dimensional mathematical equation described as follows:

$$MSE = \left(\frac{1}{N}\right)^2 \sum_{i=1}^{M} \sum_{j=1}^{N} \left(X_{ij} - \bar{X}_{ij}\right)^2$$

  where Xij = the value of pixel in cover image and $\bar{X}_{ij}$ =the value of pixel in stego-image and N is the size of image.

- PSNR: It is a measure of quality of image. PSNR can be calculated by using the mathematical formula given below:

$$PSNR = 10 \times \log \frac{255^2}{MSE} \text{ db}$$

Here we compute and compare PSNR, MSE values, when a message text 'hello' is embedded in a 5x8 gray level image in each pixel level.

Let this is be the sample image.

112	150	238	164	34	175	107	116
86	30	67	246	208	88	218	114
80	9	40	169	109	42	125	105
93	116	222	221	227	39	208	229
100	221	60	2	187	48	117	1

Let the text to be embedded is 'hello'. In decimal

[104, 101, 108, 108, 111].

In Binary

[1101000, 1100101, 1101100, 1101100, 1101111].

The values of gray level stego-image in decimal and their binary equivalent along with the calculated values of PSNR and MSE are given in table 1. The shading shows the specific bits that have been chosen for hiding the image.

From table 1, extracting the shaded bits from each byte, in each of the above techniques, gives back the embedded text as shown below.

1101000 = h
1100101 = e
1101100 = l
1101100 = l
1101111 = o

We implement the above method for an actual image lena512.bmp with the following 512 bytes of text message

Steganographic techniques have been used with success for centuries already. However, since secret information usually has a value to the ones who are not allowed to know it, there will be people or organisations who will try to decode encrypted information or find information that is hidden from them. Governments want to know what civilians or other governments are doing, companies want to be sure that trade secrets will not be sold to competitors and most persons are naturally curious. Many different motives End

and calculate the PSNR, MSE values. The stego-image and the calculated values of PSNR and MSE for each of the LSB levels are summarized in table 2.

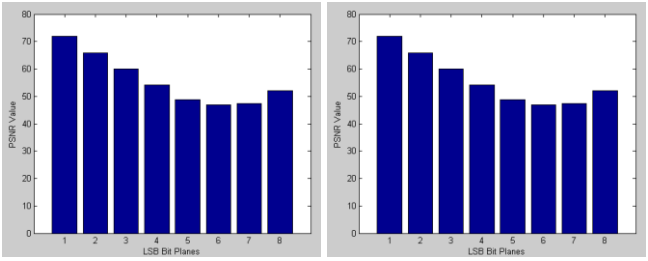


Fig 1. Histograms of the calculated values of PSNR and MSE

In figure 1, the histograms of the calculated values of PSNR and MSE indices for each LSB Bit planes are given.

3. GRAY LEVEL REPLACEMENT STEGANOGRAPHIC TECHNIQUES

We introduce some new technique by taking a simple 5x5 image matrix, and a text 'hello' which is embedded in various ways. Then this technique is applied on an actual image of lena512.bmp a 512 x 512 gray scale image and add the same 512 bytes of data and recover the same text along with actual and stego-image with their corresponding PSNR and MSE computations.

3.1 Embedding Text in Every Modal Value Column Wise

In this technique, we embed the text 'hello' in every modal value locations of each column. The mode is the most repeated value. If there is no repetition of values in a column of the matrix of the cover image, the first smallest value in that column is taken as the modal value.

Algorithm:

```
Input : message m, cover image c, stego
Image s;
Output: Stego image s
s[] = c[];
Find mode of each column in an array
mode []
for i=1 to length(m)
       s[mode(i)]=m[i]
next i
;To extract the embedded message
For each location in mode array
       Read  s[mode(i)]
```

ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology
Volume 2, Issue 2, February 2013

Table 1. Decimal and binary values of stego-image with corresponding PSNR and MSE values

LSB Level	Image after insertion	Binary equivalent	PSNR	MSE
1st Bit	112 151 239 164 35 174 106 116 86 31 67 246 208 89 218 115 80 9 41 168 109 43 124 104 92 117 223 220 227 39 208 228 100 221 61 2 187 49 117 1	01110000 10010111 11101111 10100100 00100011 10101110 01101010 01110100 01010110 00011111 01000011 11110110 11010000 01011001 11011010 01110011 01010000 00001001 00101001 10101000 01101101 00101011 01111100 01101000 01011100 01110101 11011111 11011100 11100011 00100111 11010000 11100100 01100100 11011101 00111101 00000010 10111011 00110001 01110101 00000001	51.1411	0.500
2nd Bit	112 150 238 164 34 173 105 116 84 30 67 244 208 90 216 114 80 11 42 169 111 42 125 105 93 118 222 221 227 39 208 229 100 223 62 0 187 50 119 3	01110000 10010110 11101110 10100100 00100010 10101101 01101001 01110100 01010100 00011110 01000011 11110100 11010000 01011010 11011000 01110010 01010000 00001011 00101010 10101001 01101111 00101010 01111101 01101001 01011101 01110110 11011110 11011101 11100011 00100111 11010000 11100101 01100100 11011111 00111110 00000000 10111011 00110010 01110111 00000011	46.0896	1.600
3rd Bit	112 150 238 160 38 171 107 112 82 30 71 242 208 92 218 118 80 13 44 169 109 46 121 105 89 116 222 217 231 39 208 225 96 221 60 2 191 52 117 5	01110000 10010110 11101110 10100000 00100110 10101011 01101011 01110000 01010010 00011110 01000111 11110010 11010000 01011100 11011010 01110110 01010000 00001101 00101100 10101001 01101101 00101110 01111001 01101001 01011001 01110100 11011110 11011001 11100111 00100111 11010000 11100001 01100000 11011101 00111100 00000010 10111111 00110100 01110101 00000101	38.8880	8.400
4th Bit	112 158 238 164 42 167 99 116 86 30 75 246 208 88 210 122 80 9 40 161 109 42 117 97 85 124 222 213 235 47 208 229 100 221 60 2 187 56 125 9	01110000 10011110 11101110 10100100 00101010 10100111 01100011 01110100 01010110 00011110 01001011 11110110 11010000 01011000 11010010 01111010 01010000 00001001 00101000 10100001 01101101 00101010 01110101 01100001 01010101 01111100 11011110 11010101 11101011 00101111 11010000 11100101 01100100 11011101 00111100 00000010 10111011 00111000 01111101 00001001	33.5368	28.800
5th Bit	96 150 254 164 50 175 107 100 70 30 83 230 192 88 202 114 64 25 56 169 125 58 109 105 77 116 222 205 243 55 192 229 100 221 60 2 187 48 117 17	01100000 10010110 11111110 10100100 00110010 10101111 01101011 01100100 01000110 00011110 01010011 11100110 11000000 01011000 11001010 01110010 01000000 00011001 00111000 10101001 01111101 00111010 01101101 01101001 01001101 01110100 11011110 11001101 11110011 00110111 11000000 11100101 01100100 11011101 00111100 00000010 10111011 00110000 01110101 00010001	26.8468	134.40
6th Bit	80 182 238 132 34 143 75 84 86 62 99 214 208 120 218 114 80 41 40 137 109 42 93 73 93 116 254 221 227 39 208 197 68 253 60 2 187 48 117 33	01010000 10110110 11101110 10000100 00100010 10001111 01001011 01010100 01010110 00111110 01100011 11010110 11010000 01111000 11011010 01110010 01010000 00101001 00101000 10001001 01101101 00101010 01011101 01001001 01011101 01110100 11111110 11011101 11100011 00100111 11010000 11000101 01000100 11111101 00111100 00000010 10111011 00110000 01110101 00100001	21.2608	486.40
7th Bit	48 214 238 164 98 175 43 52 22 94 67 182 144 88 154 114 16 73 104 169 109 106 61 41 29 116 222 157 227 103 144 165 36 221 124 2 251 112 117 65	00110000 11010110 11101110 10100100 01100010 10101111 00101011 00110100 00010110 01011110 01000011 10110110 10010000 01011000 10011010 01110010 00010000 01001001 01101000 10101001 01101101 01101010 00111101 00101001 00011101 01110100 11011110 10011101 11100011 01100111 10010000 10100101 00100100 11011101 01111100 00000010 11111011 01110000 01110101 01000001	13.8780	2662.4
8th Bit	112 150 238 36 162 47 107 116 86 158 195 118 80 216 90 242 80 137 168 41 237 170 125 105 93 244 222 93 227 167 80 101 100 221 188 2 187 176 245 129	01110000 10010110 11101110 00100100 10100010 00101111 01101011 01110100 01010110 10011110 11000011 01110110 01010000 11011000 01011010 11110010 01010000 10001001 10101000 00101001 11101101 10101010 01111101 01101001 01011101 11110100 11011110 01011101 11100011 10100111 01010000 01100101 01100100 11011101 10111100 00000010 10111011 10110000 11110101 10000001	8.20509	9830.4

Table 2. The stego-image and the computed values of PSNR and MSE

LSB Level	Stego image	PSNR	MSE
1st Bit		71.90228	0.00420
2nd Bit		65.85413	0.01689
3rd Bit		60.05485	0.06421
4th Bit		54.19831	0.24731
5th Bit		48.84066	0.84921
6th Bit		47.01555	1.29278
7th Bit		47.40147	1.18286
8th Bit		51.93636	0.41634

Example.

Let cover image be

10	8	6	4	1
2	5	8	9	4
6	0	9	9	8
5	8	7	4	0
9	4	2	9	1

The mode in each column is 2 8 2 9 1

Let the message be 'hello' ie., the value to be embedded is

[104 101 108 108 111]

In the cover image,

104(h) is inserted in first column 2nd location,
101(e) is inserted at second column 1st location,
108(l) is inserted in third column 5th location,
108(l) is inserted at fourth column 2nd location and
111(o) is inserted in 5th column at 1st location.

Now the image becomes

10	101	6	4	111
104	5	8	108	4
6	0	9	9	8
5	8	7	4	0
9	4	108	9	1

Now extracting the shaded location, which represents the mode locations, we get

[104 101 108 108 111]

which is 'hello'.

The procedure detailed above is implemented in the lena512.bmp gray scale image, for the 512 bytes of text presented in section 2. The original image and its stego-image embedded in column wise mode memory locations are presented in Fig. 2.

MSE = 6.09623 PSNR = 40.28019

Fig. 2 Original and Stego-mage

3.2 Embedding Text In Every Modal Value Row Wise

In this technique, we embed the text 'hello' in every modal value locations of each row. If there is no repetition of values in a row of the matrix of the cover image, the first smallest value in that row is taken as the modal value.

For the cover image given in section 3.1, the mode in each row is 1 2 9 0 9.

Let the message be 'hello', that is, the value to be embedded is

[104 101 108 108 111]

Now the image becomes

10	8	6	4	104
101	5	8	9	4
6	0	108	9	8
5	8	7	4	108
111	4	2	9	1

Now extracting the shaded location, which represents the mode locations, we get

[104 101 108 108 111]

which is 'hello'.

The procedure detailed above is implemented in the lena512.bmp gray scale image, for the 512 bytes of text presented in section 2. The original image and its stego-image embedded in column wise mode memory locations are presented in Fig. 3.

MSE= 6.21456 PSNR =40.19670

On close observation, one can find noise at the left edge of the stego-image as they are abundant memory

Fig. 3 Original and Stego-image

3.3 Embedding Text in Every Abundant Number Memory Location

A positive integer is *abundant*, if the sum of its proper divisors is greater than the number itself. For instance

$$18 < 1 + 2 + 3 + 6 + 9.$$

Since the length of the text message considered in this article consists of 512 bytes, we create the first 512 abundant numbers, some of which are listed below.

{12, 18, 20, 24, 30, 36, 40, 42, 48, 54, 56, 60, 66, 70, 72, 78, 80, 84, 88, 90, 96, 100, 102, 104, 108, 112, 114, 120,126, 132, 138, 140, 144, 150, 156, 160, 162,168, 174, 176, 180, 186, 92, 196, 198, 200, 204, 208, 210, 216, 220, 222, 224, 228, 234, 240, 246, 252, 258, 260, 264, 270, 272, 276, 280, 282, 288, 294, ……. }

and we replace each 12^{th}, 18^{th}, 20^{th}, 24^{th} , ……. pixel locations of the image with the text, to get the stego-image. We retrieve from 12^{th}, 18^{th}, 20^{th}, 24^{th} , ……. pixel locations of the stego-image, to get back the embedded message. The figure 4 below shows the original image (left) and 512 bytes of text embedded in each abundant number address column wise on the right.

The original image and its stego image embedded in column wise mode memory locations are presented in Fig. 4.

MSE= 0.16587 PSNR =55.93311

Fig. 4 Original and Stego-image

3.4 Embedding Text In Every Odd Abundant Number Memory Location

First we generate odd abundant numbers. The first odd abundant number is 945. We generate odd abundant numbers as much as the size of the embedding text. In this example, since the text consists of 512 bytes, we find the first 512 odd abundant numbers, which are listed in table 3. Then replace each of the following odd abundant number image pixel locations with one byte of text.

The original image and its stego image embedded in column wise mode memory locations are presented in Fig. 5.

MSE= 0.31371 PSNR =53.16553

Fig. 5 Original and Stego Image

3.5 Replacing Gray Level (In Each Column) That Is Equal / Closest To Text Value

In this method, we find gray level that is equal or closest to each text value in each column, and replace that gray level with the text value.

Table 3. The first 512 Odd Abundant Numbers

945	1575	2205	2835	3465	4095	4725	5355	5775	5985	6435	6615
6825	7245	7425	7875	8085	8415	8505	8925	9135	9555	9765	10395
11025	11655	12285	12705	12915	13545	14175	14805	15015	15435	16065	16695
17325	17955	18585	19215	19305	19635	19845	20475	21105	21735	21945	22275
22365	22995	23205	23625	24255	24885	25245	25515	25935	26145	26565	26775
27405	28035	28215	28665	28875	29295	29835	29925	30555	31185	31395	31815
32175	32445	33075	33345	33495	33915	34125	34155	34965	35805	36225	36855
37125	37485	38115	38745	39375	39585	40425	40635	41055	41895	42075	42315
42525	42735	43065	44415	44625	45045	45675	45885	46035	46305	47025	47355
47775	48195	48825	49665	49725	49875	50085	50505	50715	51765	51975	53235
53865	54285	55125	55575	55755	55965	56595	56925	57645	57915	58275	58695
58905	59535	61215	61425	62475	63315	63525	63945	64155	64575	65205	65835
66825	66885	67095	67275	67725	68145	68355	68985	69615	69825	70455	70785
70875	71775	72345	72765	74025	74655	75075	75735	76545	76725	77175	77385
77805	78435	78975	79695	80325	80535	81081	81585	81675	82005	82215	83265
83475	83655	84105	84315	84525	84645	85995	86625	87885	88725	88935	89505
89775	90405	91035	91245	91575	91665	92565	92925	93555	94185	94815	95445
95865	96075	96525	97335	98175	99225	100035	100485	101115	101475	101745	102375
102465	102795	103005	103455	103635	104895	105105	105525	106785	107415	108045	108675
109395	109725	110565	111375	111825	112035	112455	113715	114345	114975	116025	116235
116655	116865	118125	118755	118965	120015	121095	121275	121905	122265	123165	123585
123795	124215	124425	125685	125895	126225	126945	127575	128205	129195	129465	129675
130095	130515	130725	131355	131625	132825	133245	133875	134505	135135	137025	137445
137655	138105	138915	139755	140175	140805	141075	142065	142695	143055	143325	144375
144585	146475	146685	147735	148005	148365	148995	149175	149625	150255	151305	151515
151725	152145	152775	153153	153615	154035	155295	155925	156555	156975	157815	158235
159075	159705	159885	160545	160875	160965	161595	162225	162435	162855	163485	164835
165165	165375	166005	166635	166725	167265	167475	167895	168525	169155	169575	169785
170625	170775	171045	171171	171675	172095	172935	173565	173745	174195	174405	174825
176085	176715	177975	178605	179025	180495	181125	181335	181545	182325	182385	182655
183015	183645	184275	185535	185625	185955	186165	186615	187425	188055	188265	188955
189189	189945	190575	191565	191835	192465	192885	193545	193725	195195	195615	196245
196875	197505	197925	198135	199395	199485	199815	200025	200475	200655	201285	201825
202125	203175	203775	204435	205065	205275	206325	206745	206955	207207	208845	209055
209475	210105	210375	210735	211365	211575	212355	212625	213675	213885	214515	215325
215775	215985	216315	216405	217035	218025	218295	218925	219555	219765	220185	220605
221445	222075	222705	222915	223125	223839	223965	224595	225225	225855	227115	227205
227535	227745	228375	228735	229425	229635	229845	230175	230265	231525	232155	233415
234465	234675	235125	235305	235935	236775	236925	237195	237405	237825	238095	238875
239085	240345	240975	241395	241605	242865	243243	243705	244035	244125	244755	245025
245385	246015	246645	246675	247275	248325	248535	248625	249165	249375	249795	250425
250635	250965	251685	252315	252525	252945	253575	253935	254205	254475	255255	255645
256095	256725	257355	257565	257985	258825	259875	260865				

For example, let the cover image be

10	8	6	4	1
2	5	8	9	4
6	0	9	9	8
5	8	7	4	0
9	4	2	9	1

and let the text to be embedded is

4 9 3 6 2

Then the above text is embedded in the cover image as

10	9	6	6	2
2	5	8	9	4
6	0	9	9	8
4	8	7	4	0
9	4	3	9	1

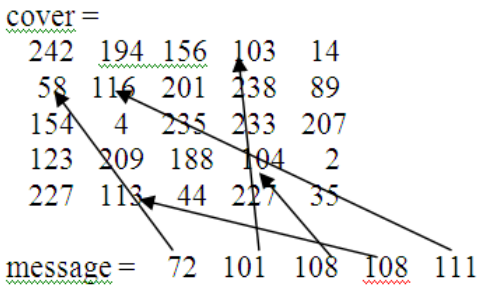
With the above technique, we get the lowest MSE. The original image and its stego image embedded equal / closest pixel value in each column are presented in Fig. 6.

MSE= 0.16389 PSNR =55.98526

Fig. 5 Original and Stego Image

3.6 Replacing Gray Level that is equal /closest to text value in entire image

The first character of message 72 is embedded at the nearest gray level 58, the second character 101 is embedded at 103 and so on as shown in the following example.



The image with embedded text is

242	194	156	101	14
72	111	201	238	89
154	4	235	233	207
123	209	188	108	2
227	108	44	227	35

The original image and its stego image embedded equal / closest pixel value in entire image are presented in Fig. 7.

MSE= 0.33418 PSNR =52.89096

Fig. 7 Original and Stego Image

3.7 Embedding Text at Random Locations in Image

Let

Cover =

129	84	90	159	72
24	154	97	175	169
145	163	116	38	191
157	44	78	234	27
180	94	181	161	87

message = 'hello' = [72 101 108 108 111]

The following random numbers are generated

4 1 3 2 2

Inserting the above text values at the memory locations of random numbers,

cover1 =

129	84	90	72	72
101	154	97	175	169
145	163	108	38	191
157	108	78	234	27
180	111	181	161	87

The original image and its stego image embedded at random generated pixel locations, in entire image are presented in Fig. 8.

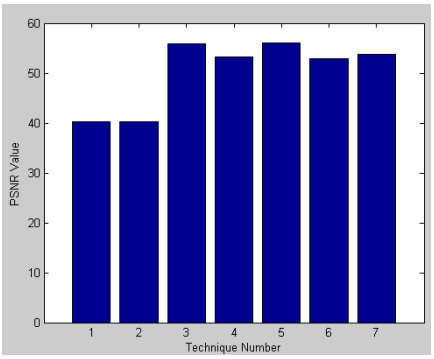
MSE= 0.27424 PSNR =53.74952

Fig. 8 Original and Stego Image

The MSE and PSNR values for the various techniques calculated in this article are presented in table 4.

Table 4. Computed values of PSNR and indices

No	Embedding technique	PSNR	MSE
1	Mode Column wise	40.28019	6.09623
2	Mode Row wise	40.19670	6.21456
3	Abundant Number location	55.93311	0.16587
4	Odd Abundant Number location	53.16553	0.31371
5	Equal/Closest Value column wise	55.98526	0.16389
6	Equal/Closest Value in entire image	52.89096	0.33418
7	Random Locations	53.74952	0.27424



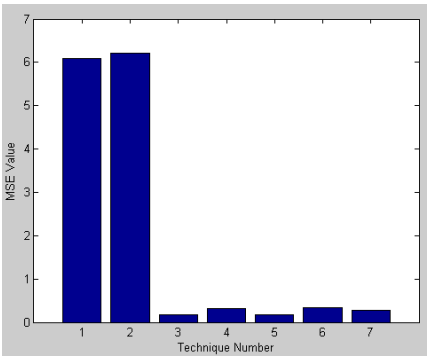


Fig. 7 Computed PSNR and MSE

The retrieved text from all the above methods is :

Steganographic techniques have been used with success for centuries already. However, since secret information usually has a value to the ones who are not allowed to know it, there will be people or organizations who will try to decode encrypted information or find information that is hidden from them. Governments want to know what civilians or other governments are doing, companies want to be sure that trade secrets will not be sold to competitors and most persons are naturally curious. Many different motives End

4. CONCLUSION

In this paper, we have discussed seven new methods, for embedding text in a gray level image. We find that embedding the text, at gray level that is equal/closest to text value gives lowest MSE.

REFERENCES

Avcibas, I.; Memon, N.; Sankur, B. *Image steganalysis with binary similarity measures*

Guillermito, *Steganography: A Few Tools to Discover Hidden Data*, 2006.

Johnson. N. F. , Z. Duric, S. Jajodia, *Information Hiding: Steganography and Watermarking – Attacks and Countermeasures*, Kluwer Academic Publishers, 2001

Misiti M., Misiti Y., Oppenheim G., and Poggi J., *Wavelet Toolbox for Use with MATLAB*, User Guide MathWorks Inc., 2000

Rizwan. U and Faheem Ahmed. H, *An Alternative Technique in Data Embedding*, Advanced Materials in Physics, pp 233 – 242, 2012

Rizwan. U and Faheem Ahmed. H, Comprehensive study on various types of steganographic schemes and possible steganalysis methods for various cover carrier like image, text, audio and video, *International Journal of Scientific and Engineering Research*, Volume **3,** Issue 11, November 2012, pp 151 – 154.

Rizwan. U and Faheem Ahmed. H, A New Approach in Steganography using different Algorithms and Applying Randomization Concept, *International Journal of Advanced Research in Computer and Communication Engineering*, Volume 1, Issue 9, November 2012, pp 233 – 242.

Matlab 7.0 Coding for embedding text in mode location

```
clc
cover=imread('lena512.bmp');
cover1=double(cover);
m=mode(cover1);
temp=1:512;
temp(:)=0;
m1='Steganographic techniques have been
used with success for centuries already.
However,';
m2='since secret information usually has
a value to the ones who are not allowed
to';
m3='know it, there will be people or
organisations who will try to decode
encrypted information';
```

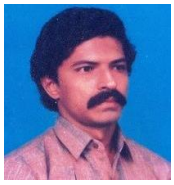
```
m4='or find information that is hidden
from them. Governments want to know
what';
m5='civilians or other governments are
doing, companies want to be sure that
trade secrets';
m6='will not be sold to competitors and
most persons are naturally curious. Many
different';
m7='motivesEnd'
message=strcat(m1,m2,m3,m4,m5,m6,m7);
msgnum=double(message);
l=length(message);
for i=1:l
    k=find( m(i) == cover1(:,i) );
    temp(i)=k(1);
    cover1(k(1),i)=msgnum(i);
end

for i=1:l
    text(i)= cover1(temp(i),i);
end
disp(char(text));
subplot(1,2,1),imshow(cover)
subplot(1,2,2),imshow(cover1,[0,255])
[rows columns] = size(cover);
% Calculate mean square error .
mseImage = (double(cover) - cover1) .^ 2;
mse = sum(sum(mseImage)) / (rows *
columns);
fprintf('\nMSE=%8.5f',mse);
% Calculate PSNR (Peak Signal to noise
ratio).
PSNR_Value = 10 * log10( 255^2 / mse);

fprintf('\nPSNR =%8.5f',PSNR_Value);
```

AUTHORS PROFILE

U. Rizwan earned his Ph.D. degree in Mathematics from the University of Madras. He is currently the Head of the Department of Mathematics, Islamiah College, Vaniyambadi and is serving the Institution for the past 26 years. He has published 41 research articles in journals of international repute. He has authored 7 books and is also the editor of two international journals. He has guided 30 M.Phil. in Mathematics scholars and one M.Tech. (IT) candidate. Presently, he is guiding Ph.D. research scholars in Mathematics and Computer Science. His research interest includes Image Processing, Hacking algorithms, Stochastic Processes, Fuzzy Logic, etc. He is the member of the board of studies in PG Mathematics of Thiruvalluvar University, Vellore. He is also an academic auditor.



H. Faheem Ahmed earned his M.Tech. degree in Information Technology from Punjabi University and M.Phil. degree in Computer Science from Manonmaniam Sundaranar University. He is pursuing Ph.D. in Computer Science. He has guided 50 M.Phil. research scholars in Computer Science. He is currently the Head of the Department of Computer Science and Applications, Islamiah College, Vaniyambadi and is serving the institution for the past 28 years. His research interest includes Steganography and Image processing. He has published 8 research articles and authored one book.