Volume 1, No. 5, December 2010
Journal of Global Research in Computer Science
ISSN-2229-371X
RESEARCH PAPER
Available Online at www.jgrcs.info

A NEW METHOD FOR EMBEDDING DATA WITHIN AN IMAGE

Prof. Samir Kumar Bandyopadhyay*[1], Biswajita Datta[2], Debjit Chakrabarty[3],Aditi Majumdar[3], Srimanti Bhowmick[3] and Nilanjana Ghosh[3]
[1]Dept. of Computer Sc. & Engineering,
University of Calcutta, Kolkata, India
e-mail:skb1@vsnl.com
[2]Lecturer, Department of Computer Sc. & Engineering,
St. Thomas College of Engineering and Technology
Kolkata, India
[3]Students, Department of Information Technology,
St. Thomas College of Engineering and Technology
Kolkata, India

Abstract: The growth of high speed computer networks and the Internet, in particular, has increased the ease of Information Communication. The cause for the development is also of the apprehension - use of digital formatted data. In comparison with Analog media, Digital media offers several distinct advantages such as high quality, easy editing, high fidelity copying, compression etc. But this type advancement in the field of data communication in other sense has hiked the fear of getting the data snooped at the time of sending it from the sender to the receiver. Information Security is becoming an inseparable part of Data Communication. In order to address this Information Security, Steganography plays an important role. Steganography is the art and science of writing hidden messages in such a way that no one apart from the sender and intended recipient even realizes there is a hidden message. This paper proposed a new method for embedding data within an image so that the image will look unchanged to human visual systems.

***Keywords*:** Network, image, security, encryption, hiding, and HIV.

INTRODUCTION

Steganography is the art of covered or hidden writing [1]. The purpose of steganography is covert communication to hide a message from a third party. Steganography comes from the Greek words Steganós (Covered) and Graptos (Writing). The origin of steganography is biological and physiological. The term "steganography" came into use in 1500's after the appearance of Trithemius' book on the subject "Steganographia". A short overview in this field can be divided into three parts and they are Past, Present and Future [2].

Steganography in the modern day sense of the word usually refers to information or a file that has been concealed inside a digital Picture, Video or Audio file. What Steganography essentially does is exploit human perception; human senses are not trained to look for files that have information hidden inside of them. Generally, in steganography, the actual information is not maintained in its original format and thereby it is converted into an alternative equivalent multimedia file like image, video or audio which in turn is being hidden within another object. This apparent message (known as cover text in usual terms) is sent through the network to the recipient, where the actual message is separated from it.

The majority of today's steganographic systems uses multimedia objects like image, audio, video etc. as cover media because people often transmit digital pictures over email and other Internet communication [3]. In modern approach, depending on the nature of cover object, steganography can be divided into five types:

- Text Steganography
- Image Steganography
- Audio Steganography
- Video Steganography
- Protocol Steganography

So, in the modern age so many steganographic techniques have been designed which works with the above concerned objects. More often in today's security advancement, we sometimes come across certain cases in which a combination of Cryptography and Steganography are used to achieve data privacy over secrecy. In this paper, we proposed a new method for embedding data within an image so that the image will look unchanged to human visual systems (HVS).

The word "Steganography" technically means "covered or hidden writing". Its ancient origins can be traced back to 440 BC. Although the term steganography was only coined at the end of the 15th century, the use of steganography dates back several millennia. In ancient times, messages were

hidden on the back of wax writing tables, written on the stomachs of rabbits, or tattooed on the scalp of slaves.

In today's world, we often listen a popular term "Hacking". Hacking is nothing but an unauthorized access of data which can be collected at the time of data transmission. With respect to steganography this problem is often taken as Steganalysis.

Information can be hidden inside a multimedia object using many suitable techniques. As a cover object, we can select image, audio or video file. Depending on the type of the cover object, definite and appropriate technique is followed in order to obtain security.

Since everyone can read, encoding text in neutral sentences is doubtfully effective. But taking the first letter of each word of the previous sentence, you will see that it is possible and not very difficult. Hiding information in plain text can be done in many different ways [4].

Many techniques involve the modification of the layout of a text, rules like using every n-th character or the altering of the amount of white space after lines or between words [5]. The last technique was successfully used in practice and even after a text has been printed and copied on paper for ten times, the secret message could still be retrieved. Another possible way of storing a secret inside a text is using a publicly available cover source, a book or a newspaper, and using a code which consists for example of a combination of a page number, a line number and a character number. This way, no information stored inside the cover source will lead to the hidden message. Discovering it relies solely on gaining knowledge of the secret key.

To hide information, straight message insertion may encode every bit of information in the image or selectively embed the message in "noisy" areas that draw less attention—those areas where there is a great deal of natural colour variation. The message may also be scattered randomly throughout the image. A number of ways exist to hide information in digital media. Common approaches include:

- Least significant bit insertion
- Masking and filtering
- Redundant Pattern Encoding
- Encrypt and Scatter
- Algorithms and transformations

Each of these techniques can be applied, with varying degrees of success.

Least significant bit (LSB) insertion is a common and simple approach to embed information in an image file. In this method the LSB of a byte is replaced with an M's bit. This technique works good for image, audio and video steganography. To the human eye, the resulting image will look identical to the cover object [1].

For example, if we consider image steganography then the letter A can be hidden in three pixels (assuming no compression). The original raster data for 3 pixels (9 bytes) may be

(00100111 11101001 11001000)
(00100111 11001000 11101001)
(11001000 00100111 11101001)

The binary value for A is 10000001. Inserting the binary value for A in the three pixels would result in

(00100111 11101000 11001000)
(00100110 11001000 11101000)
(11001000 00100111 11101001)

The underlined bits are the only three actually changed in the 8 bytes used. On average, LSB requires that only half the bits in an image be changed. You can hide data in the least and second least significant bits and still the human eye would not be able to discern it. The resultant image for the above data insertion and the original cover image are given below.

Fig. 1: The cover image

Fig. 2: The stego-image (after A is inserted)

Masking and filtering techniques are mostly used on 24 bit and grey scale images. They hide info in a way similar to watermarks on actual paper and are sometimes used as digital watermarks. Masking images entails changing the luminance of the masked area. The smaller the luminance change, the less of a chance that it can be detected [1, 4-5].

Patchwork and other similar tools do redundant pattern encoding, which is a sort of spread spectrum technique. It works by scattering the message throughout the picture. This makes the image more resistant to cropping and rotation. Smaller secret images work better to increase the redundancy embedded in the cover image, and thus make it easier to recover if the stego-image is manipulated [1, 4].

The Encrypt and Scatter technique tries to emulate white noise. It is mostly used in image steganography. White Noise Storm is one such program that employs spread

spectrum and frequency hopping. It does this by scattering the message throughout an image on eight channels within a random number that is generated by the previous window size and data channel. The channels then swap rotate, and interlace amongst each other. Each channel represents one bit and as a result there are many unaffected bits in each channel. This technique is a lot harder to extract a message out of than an LSB scheme because to decode first detect that a hidden image exists and extract the bit pattern from the file. While that is true for any stego-image you will also need the algorithm and stego key to decode the bit pattern, both of which are not required to recover a message from LSB. Some people prefer this method due to the considerable amount of extra effort that someone without the algorithm and stego-key would have to go through to extract the message. Even though White Noise Storm provides extra security against message extraction it is just as susceptible as straight LSB to image degradation due to image processing [1, 5].

LSB modification technique for images does hold good if any kind of compression is done on the resultant stego-image e.g. JPEG, GIF etc [20]. JPEG images use the discrete cosine transform to achieve compression. DCT is a lossy compression transform because the cosine values cannot be calculated exactly, and repeated calculations using limited precision numbers introduce rounding errors into the final result. Variances between original data values and restored data values depend on the method used to calculate DCT [6, 7, 8].

Like LSB our proposed method is efficient instead of that it's not easy to analysis, however, standard LSB is not effective in term of the data hidden quantity, all researchers agreed the fact that the size of data hidden is a problem in that particular area, the other problem that faced there, in fact if we try to increase the quantity of data in the image there will be a suspect changes which become clear to human eyes. Our approach will face a challenge that high rate data hidden without affecting the images quality.

PROPOSED METHOD

Here our main aim is to hide some information (text) within an image. We call the text to be hidden as target text and the image under which they are to be hidden as cover image.

Here we consider 24 bit colour BMP images as cover image. Each colour used has a 24-bit RGB value. In such images each 24 bits pixel thought of as a collection of 3bytes where the first byte (first 8 bits) represents the Gray level of Red component, the next byte represents the Gray level of Green component and the last byte represents the Gray level of Blue component.

Here we proposed a new method for embedding data within an image so that the image will look unchanged to HVS.

Algorithm for Encoding (hiding) the data

Step 1: Start
Step 2: Read the Cover Image and the target Text massage
Step 3: Convert the target massage into upper Case
Step 4: Calculate the length of the converted String and store it in a variable – STRLEN
Step 5: Set L=STRLEN
Step 6: Store the value of STRLEN in the first pixel of the Cover Image using the function
STOR_LEN (STRLEN, Cover Image)
Step 7: For 1 to STRLEN repeat the steps 8 to 13.
Step 8: Convert the ASC II value of the individual character of the target String into its 7 bit binary equivalent
Step 9: Cut the MSB to convert the 7 bit binary into 6 bit binary
Step 10: Use the function M (L) to find out the pixel location of the cover image for
Embedding the target Massage
Step 11: Replace two LSB (7^{th} & 8^{th} position from MSB) of red green and blue component
of the target pixel by sequentially $1^{st} – 2^{nd}$, $3^{rd} – 4^{th}$ and $5^{th} – 6^{th}$ positional bit value
from MSB of 6 bit binary.
Step 12: L=L+1.
Step 13: Restore the bits of the Red, Green and Blue component of modified pixel in the cover image.
Step 14: Send the stego- image to the receiver.
Step 15: End

Algorithm for Function STOR_LEN (STRLEN, Cover Image)

/* Storing of the length of the message */

Step 1: Start
Step 2: Convert the value STRLEN into its 8 bit binary equivalent.
Step 3: Replace three LSBs (6^{th}, 7^{th} & 8^{th} position from MSB) of Red & Green component
and two LSBs(7^{th} & 8^{th} position from MSB) of Blue component of first pixel by
sequentially 1^{st} - $2^{nd} – 3^{rd}$, $4^{th} – 5^{th}$ - 6^{th} & 7^{th} - 8^{th} positional bit value from MSB of
the 8 bit binary.
Step 4: End

Algorithm for Function M (L)

/* Finding the affected pixel location */

Step 1: Start
Step 2: If L is equal to STRLEN,
Put P=L
else
Put P= (L+(L-1))*L
Step 3: Return P
Step 4: End

Algorithm for extracting the secret data

Step 1: Start
Step 2: To obtain the length of the target text message from the 1^{st} pixel of the image call
function RET_LEN(Cover image) and store the length in STRLEN.
Step 3: Set L=STRLEN
Step 4: For 1 to STRLEN repeat the steps 5 to 10.
Step 5: Call the function M (L) to find out the pixel location of the stego image where the target text Massage is embedded.

Step 6: Read two LSBs (7^{th} & 8^{th} position from MSB) of Red, Green & Blue components of the modified pixel of the stego image.
Step 7: Concatenate the retrieved bits according to the order Red, Green & Blue to form 6 bit
binary (where the collected values of R,G & B components are put as 1^{st} -2^{nd} , 3^{rd} –
4^{th} & 5^{th} – 6^{th} positional bit value from MSB 6 bit binary)
Step 8: If the two LSBs (7^{th} & 8^{th} position from MSB) of red component be 00 or 01, then
Concatenate 1 with 6 bit binary as the MSB bit.
else if the two LSBs (7^{th} & 8^{th} position from MSB) of red component be 10 or 11,
then concatenate 0 with 6 bit binary as the MSB bit.
From here we get the 7 bit binary from its 6 bit binary.
Step 9: Convert this 7 bit binary into its equivalent ASC II value of the individual character
of the target String.
Step 10: set L=L+1
Step 11: Concatenate all the characters to obtain target message
Step 12: End

Algorithm for Function RET_LEN (Cover image)

/* Recover the length on receiver side */

Step1: Start
Step 2: Read the three LSBs (6^{th} , 7^{th} & 8^{th} position from MSB) of Red, Green components & two LSBs (7^{th} & 8^{th} position from
MSB) of Blue component of the modified pixel of the stego image.

Step 3: Concatenate the retrieved bits according to the order Red, Green & Blue to form 8 bit binary.
Step 4: Convert the binary to its corresponding decimal.
Step 5: Return the length of the target text message.
Step 6: End

The algorithm executes in the following steps:

REPLACEMENT OF RGB COMPONENT

In case of data hiding we replace two LSBs of each of the RGB components with the binary value of ASCII of the character to be hidden.
Now we demonstrate our proposed method. Suppose we want to hide the character 'A' within a pixel of our image. Let the gray level value of R, G, B component of that pixel of the image be as follows:

1 1 1 1 1 1 1 1	1 1 0 1 1 0 1 0	1 0 0 1 1 0 1 0
R=255	**G=218**	**B=154**

Now suppose we want to store 'A' in this pixel. The ASCII value of 'A' is 65 in binary which is represented in 7 bit as '1000001'. Our aim is modify the first 2 bit LSB of R, G, B component each of a pixel. To do this i.e. to make 7 bit ASCII into 6 bit we discard the MSB '1' of '1000001'. Then we get 6 bit as '000001'. Then cut these bits in 2- 2- 2 order as follows:

00 00 01

and replace the LSB of the pixel with them. So the final pixel becomes-

1 1 1 1 1 1 **0 0**	1 1 0 1 1 0 **0 0**	1 0 0 1 1 0 **0 1**
R=252	**G=216**	**B=153**

PIXEL SELECTION

As we can see the change thus obtained in the pixel value is negligible. But if all the adjacent pixels are changed at the same time it may bring about a large change. For this reason instead of changing all the adjacent pixels, we change a certain number of selected pixels. The pixels are selected according to the following order-Suppose the starting point is 1, then after the 1^{st} pixel, the next pixels changed will be the following order: -

1	6	15	28	45
[(1+0)*1]	[(2+1)*2]	[(3+2)*3]	[(4+3)*4]	[(5+4)*5]

But if the length of the string be 21, we start it from 21. Then the affected pixels are selected according to following order:

21	861	946	. . .	3321
[(21+20)*21]	[(22+21)*22]	b	. . .	[(41+40)*41]

LENGTH OF STRING STORING

We place it at the 1st pixel of the cover image in general 3-3-2 format according to the following:
Let the value of R, G, B component of 1st pixel be

1 1 1 1 1 0 1 0	1 1 0 1 0 0 1 0	1 0 1 1 1 1 1 0
R=250	**G=210**	**B=190**

Suppose the length of the string be 21. The binary value of 21 is '00010101' (8 bits). Cut the bits 3-3-2 format

000 101 01

and replace the R, G, B component of the 1st pixel. So the final Gray level value of R, G, B component of 1st pixel be

1 1 1 1 1 **0 0 0**	1 1 0 1 0 **1 0 1**	1 0 1 1 1 1 **0 1**
R=248	**G=213**	**B=189**

Decoding Technique
At the time of decoding we pick the length from 1st pixel according to the 3-3-2 format. Then follow the above series to get the modified pixels where the target information is hidden.
Suppose at a particular pixel of the stego image the Gray level value of R, G, B component be

1 1 1 1 1 1 **0 0**	1 1 0 1 1 0 **0 0**	1 0 0 1 1 0 **0 1**
R=252	**G=216**	**B=153**

Collect the 2 LSB bits of each component and concatenate them according to the order Red, Green & Blue to get 6 bit binary value. Here it becomes '000001' (00 (R) 00 (G) 01 (B)). Then we examine 2 LSB bits of R component of that particular pixel of the stego image (here it is 00) . If it is 00 or 01 then we consider MSB as 1 else if it is 10 or 11 then we consider MSB as 0.
Here 2 LSB bits of R component is 00 so we concatenate 1 with the 6 bit binary '000001' and get 7 bit binary as '1000001'. Then convert this 7 bit binary into equivalent decimal to get the ASCII (here it is 65), from the ASCII we retrieve the embedded textual message (so 65 is converted to character 'A').

RESULT & DISCUSSION

An English message text is written by using the alphabetic characters of the English language ((which are 26 letters ('A' … 'Z')) as well as numeric digits (which are 0 to 9). Some other special characters are also used to give the reader a proper understanding of the message. Here we consider only the uppercase alphabetic characters; numeric digits and some most commonly used special character for the better understanding of target message. The characters consider in this study are given in the following Table I.

Table -I

Character	ASCII Code	Binary				Character	ASCII Code	Binary				Character	ASCII Code	Binary			
		MSB	R	G	B			MSB	R	G	B			MSB	R	G	B
A	65	1	00	00	01	O	79	1	00	11	11	2	50	0	11	00	10
B	66	1	00	00	10	P	80	1	01	00	00	3	51	0	11	00	11
C	67	1	00	00	11	Q	81	1	01	00	01	4	52	0	11	01	00
D	68	1	00	01	00	R	82	1	01	00	10	5	53	0	11	01	01
E	69	1	00	01	01	S	83	1	01	00	11	6	54	0	11	01	10
F	70	1	00	01	10	T	84	1	01	01	00	7	55	0	11	01	11
G	71	1	00	01	11	U	85	1	01	01	01	8	56	0	11	10	00
H	72	1	00	10	00	V	86	1	01	01	10	9	57	0	11	10	01
I	73	1	00	10	01	W	87	1	01	01	11	.	46	0	10	11	10
J	74	1	00	10	10	X	88	1	01	10	00	,	44	0	10	11	00
K	75	1	00	10	11	Y	89	1	01	10	01	?	63	0	11	11	11
L	76	1	00	11	00	Z	90	1	01	10	10	!	33	0	10	00	01
M	77	1	00	11	01	0	48	0	11	00	00	blank	32	0	10	00	00

N	78	1 00 11 10	1	49	0 11 00 01	-	45	0 10 11 01

Before going into detailed discussion, at the sender side we choose the information of the target message and a cover image file under which the target message is to be hidden.

Test Case 1:

Cover Image: lena.bmp

Stego Image:

Target message:
THIS IS A STEGO IMAGE

Retrieved message:
THIS IS A STEGO IMAGE

Test Case 2:

Cover Image: lena.jpg

Stego Image:

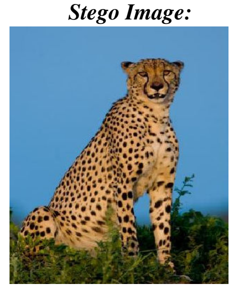
Target message:
COMMONWEALTH GAMES - 2010 – INDIA

Retrieved message:
COMMONWEALTH GAMES - 2010 - INDIA

Test Case 3:

Cover Image: cheetah.bmp

Stego Image:



Target message:
THIS IS A STEGO IMAGE

Retrieved message:
THIS IS A STEGO IMAGE

Test Case 4:

Cover Image: dance.jpg

Stego Image:

Target message:
COMMONWEALTH GAMES - 2010 – INDIA

Retrieved message:
COMMONWEALTH GAMES - 2010 - INDIA

For detailed discussion we consider the test case 1. In test case 1

Target message: **this is a stego image**
Cover image: **lena.bmp**

Before embedding process we need to convert the target message into upper case. Then we store the length of the message in the first pixel using 3-3-2 approach. Then we start our approach using our proposed method.

Now we start our work with "**THIS IS A STEGO IMAGE**". Then we start our work according to the Proposed Method as per the following Table II.

Table II

Character Of Target Message	ASCII Of Corresponding Character	MSB	Binary 6 bit R G B	Affected Pixel Number	Old gray level value in binary for Red	Changed gray level value in binary for Red	Old gray level value in binary for Green	Changed gray level value in binary for Green	Old gray level value in binary for Blue	Changed gray level value in binary for Blue
T	84	1	01 01 00	21	176	177	91	89	60	60
H	72	1	00 10 00	861	190	188	110	110	75	72
I	73	1	00 10 01	946	242	240	172	174	120	121
S	83	1	01 00 11	1035	190	189	106	104	69	71
Blank	32	0	10 00 00	1128	157	158	82	80	59	56
I	73	1	00 10 01	1225	159	156	78	78	57	57
S	83	1	01 00 11	1326	155	153	83	80	59	59
Blank	32	0	10 00 00	1431	182	182	98	96	70	68
A	65	1	00 00 01	1540	198	196	114	112	78	77
Blank	32	0	10 00 00	1653	161	162	84	84	58	56
S	83	1	01 00 11	1770	157	157	83	80	58	59
T	84	1	01 01 00	1891	214	213	135	133	102	100
E	69	1	00 01 01	2016	137	136	76	77	57	57
G	71	1	00 01 11	2145	144	144	73	73	53	55
O	79	1	00 11 11	2278	160	160	85	87	62	63
Blank	32	0	10 00 00	2415	186	186	100	100	65	64
I	73	1	00 10 01	2556	149	148	79	78	54	53
M	77	1	00 11 01	2701	160	160	82	83	60	61
A	65	1	00 00 01	2850	155	152	80	80	57	57
G	71	1	00 01 11	3003	248	248	185	185	132	135
E	69	1	00 01 01	3160	142	140	70	69	58	57

According to the same way other test cases hide data within the particular cover image. Editing the old gray level value for R, G & B component of a selected pixel with intended binary values causes a negligible hange in the original cover image file that remains almost imperceptible to HVS.

At the receiving end the data retrieving algorithm work for decoding the massage from the cover image as follows.

First the length of the hidden message is retrieved from the first pixel. Now According to the defined series we find out the pixel position where the data bits are embedded. Then we pick the 2 least significant bits from each of the R, G & B component of each pixel and concatenate them to get 6 bit binary. Then either 0 or 1 based on 2 least significant bit value of red component (0 if they are 11 or 10, 1 if they are 00 or 01) is added as the MSB to get the 7 bit binary. Convert the 7bit to ASCII from which the text can be retrieved.

The whole retrieval process can be depicted thoroughly as the test case 1 in the following Table III.

Table III

Affected pixel number	Gray level value in decimal for Red	Gray level value in binary for Red	Gray level value in decimal for Green	Gray level value in binary for Green	Gray level value in decimal for Blue	Gray level value in binary for Blue	MSB (decide based on 2 LSB of Red)	Concatena-ted 6 bit	Concatena-ted 7 bit	Correspo nding ASCII	Correspo n-ding Character
21	177	10110001	89	01011001	60	00111100	1	01 01 00	1010100	84	T
861	188	10111100	110	01101110	72	01001000	1	00 10 00	1001000	72	H
946	240	11110000	174	10101110	121	01111001	1	00 10 01	1001001	73	I
1035	189	10111101	104	01101000	71	01000111	1	01 00 11	1010011	83	S
1128	158	10011110	80	01010000	56	00111000	0	10 00 00	0100000	32	blank
1225	156	10011100	78	01001110	57	00111001	1	00 10 01	1001001	73	I
1326	153	10011001	80	01010000	59	00111011	1	01 00 11	1010011	83	S
1431	182	10110110	96	01100000	68	01000100	0	10 00 00	0100000	32	blank
1540	196	11000100	112	01110000	77	01001101	1	00 00 01	1000001	65	A
1653	162	10100010	84	01010100	56	00111000	0	10 00 00	0100000	32	blank
1770	157	10011101	80	01010000	59	00111011	1	01 00 11	1010011	83	S
1891	213	11010101	133	10000101	100	01100100	1	01 01 00	1010100	84	T
2016	136	10001000	77	01001101	57	00111001	1	00 01 01	1000101	69	E
2145	144	10010000	73	01001001	55	00110111	1	00 01 11	1000111	71	G
2278	160	10100000	87	01010111	63	00111111	1	00 11 11	1001111	79	O

2415	186	10111010	100	01100100	64	01000000	0	10 00 00	0100000	32	blank
2556	148	10010100	78	01001110	53	00110101	1	00 10 01	1001001	73	I
2701	160	10100000	83	01010011	61	00111101	1	00 11 01	1001101	77	M
2850	152	10011000	80	01010000	57	00111001	1	00 00 01	1000001	65	A
3003	248	11111000	185	10111001	135	10000111	1	00 01 11	1000111	71	G
3160	140	10001100	69	01000101	57	00111001	1	00 01 01	1000101	69	E

From table II we can see that the changes in R, G, & B are so minimal that it cannot affect in human eyes. If we follow the column "Affected pixel number" from table II & III we see that the size of the cover image is also very small; here to store target message of size 21 we require minimum 60 × 60 sized cover image. From the experimental result we found that for the target message of size 50 we require minimum 150 × 150 sized cover image. So when we send this type of stego image through internet it takes lesser bandwidth.

Here we choose 512 × 512 image for better vision of the stego image. But if we choose such types of large cover image another option is also open to us. In that case we can hide more than one target message in the image. But one thing is need to be considered – the size of the second message should be greater by 2 × size of the first message, the size of the third message should be 2 × size of the second message and so on.

Here we consider only the uppercase alphabetic character for interpretation of characters from the stego message at the receiver side. Otherwise according to our hidden process some of the lower case letters in 6 bit match with 6 bits of some mostly used special characters that we consider in this study.

CONCLUSIONS

Any different techniques exist and continue to be developed, while the ways of detecting hidden messages also advance quickly. Since detection can never give a guarantee of finding all hidden information, it can be used together with methods of defeating steganography, to minimize the chances of hidden communication taking place. Even then, perfect steganography, where the secret key will merely point out parts of a cover source which form the message, will pass undetected, because the cover source contains no information about the secret message at all. Here we [9] proposed a new method for embedding data within an image so that the image will look unchanged to HVS.

REFERENCES

[1] Johnson, N. F. and Jajodia, S. (1998). Exploring steganography: Seeing the unseen. Computer, 31(2):26–34.

[2] David Kahn, "The History of Steganography", Proc. of First Int. Workshop on Information Hiding, Cambridge,UK, May30-June1 1996, Lecture notes in Computer Science, Vol.1174, Ross Anderson (Ed.), pp.1-7.

[3] B.Pfitzmann, "Information Hiding Terminology", Proc. of First Int. Workshop on Information Hiding, Cambridge, UK, May30-June1, 1996, Lecture notes in Computer Science, Vol.1174, Ross Anderson(Ed.), pp.347-350.

[4] F.A.P.Petitcolas, et al.,"Information Hiding – A Survey", Proceedings of the IEEE, Vol.87, No.7, July 1999, pp.1062-1078.

[5] Johnson, N. F. and Jajodia, S. (1998). Exploring steganography: Seeing the unseen. Computer, 31(2):26–34.

[6] Westfeld, A. (2001). F5-a steganographic algorithm: High capacity despite better steganalysis. In Proc. 4th Int'l Workshop Information Hiding, pages 289–302.

[7] W. Brown and B.J. Shepherd, Graphics File Formats: Reference and Guide, Manning Publications, Greenwich, Conn, 1995.

[8] E. Koch, J. Rindfrey, and J. Zhao, "Copyright Protection for Multimedia Data," Proc. Int'l Conf. Digital Media and Electronic Publishing, Leeds, UK 1994.

