

A Graph Theoretic Approach to Sustainable Steganography

Vinay Kumar¹, Sunil Kumar Mutttoo²

¹Scientist 'E,' National Informatics Center, MoCIT, Government of India

²Department of Computer Science, University of Delhi

ABSTRACT: *An algorithm based on graph-theoretic approach is introduced in this paper. A bipartite graph is created from message and cover object. Message M is broken into units of x ($= 2, 4$, or 8) bits long. For each x , a matching with m number of such x bits from cover file is determined using the bipartite graph. Wherever a matching for a node in left side is found with a node in right side then this part of the message is treated as either naturally or cross embedded in that part of the cover. Nodes in left side correspond to bits in message and those in right side correspond to group of bits in cover. The matching relationship is then embedded in the extra bytes of cover, fully utilizing the available redundancy or alternatively the sequence of indices is compressed and sent through separate channel. The algorithm achieves almost 100% matching for message elements in cover elements. The embedding algorithm has been put through mathematical and statistical test to ensure that it not only retains visual similarity in stego with cover file but also leaves other statistics of cover undistorted after embedding. Therefore it achieves sustainability. In this paper, we have taken BMP file to implement the algorithm.*

KEYWORDS: *Extra Bytes, Graph Theoretic Approach, Steganography, Information Hiding, Sustainable Embedding, Natural Embedding, Partial Embedding, Cross Embedding, Explicit Embedding.*

1. Introduction

Steganography, also called “covered writing” is defined as the art and science of communicating in a way that hides the very existence of the communication. Steganography and Cryptography are excellent means to achieve privacy and secrecy of information to be shared between communicating partners. A mechanism to combine them provides multiple layers of security. Statistical and visual undetectability of a stego object when compared with cover object is an important consideration for any steganographic schemes. By undetectability, we understand the inability of an attacker to distinguish between stego and cover objects with success rate better than random guessing, given the knowledge of embedding algorithm and the source of cover media.

There are a number of steganographic approaches in use for hiding information in digital images. The spatial domain, frequency domain and spread spectrum technique are mostly used for information hiding (Anonymous, 1995; Sellars, 2006). The simple

and most obvious is substitution method that replaces Least Significant Bit (LSB) or randomly selected bit, image downgrading, cover region and parity bits etc (Katzenbeisser & Petitcolas, 2000). Another approach is statistical steganography that utilizes a 1-bit steganographic scheme (Katzenbeisser & Petitcolas, 2000). These approaches alter some bits that are part of color pallet of the image. Thus statistical analysis between the known cover and stego object may reveal the presence of information in the image. Another work based on information theoretic approach by Cox et al. (2005) embeds information using correlation coefficient between two distributions drawn from message and cover. The number of bits to be altered depends on the value of the correlation.

Steganographic research is primarily driven by the desire to have complete secrecy of information in an open-systems environment. This is not available with cryptographic systems. Steganography (Cole, 2003; Cox et al., 2007; Johnson, 1995; Johnson & Jajodia, 1998a) and cryptography (Stallings, 1999) are used for the purpose to protecting the information from a third party. In a cryptographic system, a hacker may be able to detect, intercept, modify and/or destroy messages without being able to violate certain security premises guaranteed by a cryptosystem (Lenstra, Wang & Weger, 2005; Merkle, 1990; SANS Institute, 2001; Schneier, 1996), whereas in a steganographic system a message is embedded inside other harmless messages in a way that it does not allow third party to even know the presence of the message (Johnson, Duric & Jajodia, 2001; Krinn, 2000). Information hiding is used in digital watermarking (Cox et al., 2007) wherein cover needs to be retained after retrieval of message unlike in steganography. Also in administrative control some administrative decision needs delayed disclosure of information. Until that time privacy of information is to be ensured. This is applicable to both data at store as well as data on move in the era of Internet where most of data is shared using telecommunication transmission i.e. computer network.

If a message is embedded without replacing or exchanging any color bits of cover data, an almost perfect steganography is achieved! In this paper, a steganographic algorithm using graph theoretic approach is introduced that exactly achieves that. The following four factors influence a steganographic security.

- Type of cover media
- Method of selection of places within the cover that might be modified
- The embedding operations
- The number of embedding changes

The idea behind a steganographic process is to find an embedding algorithm that finds a suitable cover, determines redundancy in it and finally preserves statistics of the cover. Statistics such as color frequency, average absolute difference (AD), mean squared

error (MSE), L^p -norm, Laplacian mean squared error (LMSE), peak signal to noise ratio (PSNR) and histogram similarity (HS) of the original cover image should be preserved in the stego to confuse the hacker. The graph theoretic approach to steganography is presented by Hetzl and Mutzel (2005) that basically preserves color frequency of the cover by exchanging a basic elements of cover from one block to another to infer message as a functional output of the exchanged block. The approach does not preserve the other statistics outlined above. A novel approach is presented in this paper that preserves the above statistics besides the color frequencies.

There are two ways of using graph theoretic concept in steganography.

- Find relationship (if required) between smallest data unit of message and a group of such smallest unit of cover object and represent the relationship using a graph. If required, hide the relationships in the zero bytes of cover.
- Use a graph as cover object and find redundancy in its feature like node or segment or its attributes and embed payload in it (Krinn, 2000).

A digital cover image can be treated as collection of data units. Each data unit is nothing but string of some x bits. Thus a cover is an array of such data units. Similarly a secret message to be embedded in the cover may be treated as an array of data units each of the same x bits length. Utilising this concept and the first approach listed above, an algorithm is presented in this paper to hide information in an image in sustainable way. We generate a bipartite graph from cover image and message.

The paper is organized is seven sections. Section 2 contains terminology and a theoretical description of the graph theoretic approach for finding relationships. The algorithm to represent the relationship in graph and to store, wherever required, in cover is described in Section 3. In Section 4, we have explained how to get basic information about cover image and use it to store any information if required. Extraction process is mentioned in Section 5. Steganographic security of our approach is described in Section 6. The paper is concluded in Section 7.

2. Theoretical approach

We treat message (M) as an array of elements, each element containing sequence of x bits, where x is taken from set $\{2, 4, 8\}$ to avoid padding of bits. The value 4 is found very optimal and in many cases ($> 99\%$) we have achieved 100% natural and cross embedding. Before proceeding further it is important to briefly introduce terms frequently used in this paper.

2.1 Terminology

2.1.1 Bipartite graph

A graph $G = (V, E)$ is called a bipartite graph if vertex set V can be partitioned into two non empty disjoint subsets V_1 and V_2 in such a way that every edge in set E joins a vertex in V_1 to a vertex in V_2 . No node in $V_1(V_2)$ is adjacent to any node in $V_1(V_2)$. However some nodes in V_1 are adjacent to some nodes in V_2 .

2.1.2 Embedding factor

Let size of message array be L . An embedding factor is defined as ratio

$$k = \frac{\text{size of cover array}}{\text{size of message array}}$$

It implies that potentially k data units are available in cover for one data unit of secret message.

2.1.3 Cross embedding

Let embedding factor k be 4 and data unit size x be 2. A r^{th} data unit of message is said to be cross embedded in the s^{th} k data units of cover, if $r \neq s$ and additional modulo 2^x of s^{th} k data units of cover is equal to r^{th} data unit of message.

2.1.4 Explicit embedding

When none of the embeddings like cross, natural or partial is achievable, then r^{th} data unit of message is to be explicitly written in the extra bytes of the cover. This situation is defined as explicit embedding.

2.1.5 Extra byte

While storing an image in a file format a certain format constraint is maintained. In 24 bit BMP a pixel needs 3 bytes for RGB. Every line starts from a quad boundary, thus a few bytes (0 to 3) are padded to ensure that number of bytes in a line remains multiple of 4. These padded bytes are called extra bytes or zero bytes.

2.1.6 Natural embedding

Let embedding factor k be 4 and data unit size x be 2. A r^{th} data unit “01” of message is said to be naturally present in the r^{th} k data units of cover “10 00 01 10” because additional modulo 4 of 10, 00, 01 and 10 is equal to 01.

2.1.7 Partial embedding

When r^{th} data unit of message is neither naturally embedded nor cross embedded but it is present as one of the k in the r^{th} k data units of cover then it is called partially embedded in the cover.

2.1.8 Sustainable steganography

By sustainable steganography we mean preserving statistics of cover in stego by avoiding possible distortion in the cover due to embedding of the message in it.

2.2 Methodology

Let $m_1, m_2, m_3, \dots, m_L$ be L sequences, each of x bits, of message M . Let us use M to denote size of message, thus

$$M = L * x \text{ bits.}$$

We use 24bit BMP file as cover object (C) to implement the algorithm. The 24bit BMP image has 2^{24} colors and the palette field does not contain any entry (Gonzalez & Woods, 1992). Each 3-byte triplet in the bitmap array represents relative intensities of blue, green, and red color of a pixel. The actual pixel data begins at the offset of 54 bytes from the start. The 54 bytes header info is left unchanged. If size of cover image is $W * H$, where W is width and H is height in pixels then there are

$$3 * (W * H) \text{ bytes}$$

for colors, some zero bytes besides 54 bytes used for header in the cover. Zero bytes are added at the end of every line ($3 * W$) bytes to ensure that next line begins at the 32bits-boundary (Gonzalez & Woods, 1992).

We also consider $3(W * H)$ bytes of cover image C as array of elements, each element containing x bits. The value of x is same as in M . Let $c_1, c_2, c_3, \dots, c_N$ be N sequences, each of x bits, of cover C . If we take C as size of bitmap array in 24bit BMP cover then

$$C = 3 * (W * H) = N * x.$$

This implies that each m_j has k

$$k = \frac{N}{L}$$

c_j 's available in C . For implementation purpose, $c_1, c_2, c_3, \dots, c_N$ can be organized as $(c_{1,1}, c_{1,2}, c_{1,3}, \dots, c_{1,k}), (c_{2,1}, c_{2,2}, c_{2,3}, \dots, c_{2,k}), \dots, (c_{L,1}, c_{L,2}, c_{L,3}, \dots, c_{L,k})$ such that

$$N = k * L$$

Since k bits of cover are used for one bit of message, k is called embedding factor of the algorithm. We find relationships between each m_j and $(c_{j,1}, c_{j,2}, c_{j,3}, \dots, c_{j,k})$ as follows.

- If $(c_{j,1} + c_{j,2} + c_{j,3} + \dots + c_{j,k}) \text{ modulo } 2^x = m_j$, then m_j is considered as naturally embedded in C .
- For the leftover m_i 's, if it is found that it is naturally embedded in a $(c_{j,1}, c_{j,2}, c_{j,3}, \dots, c_{j,k})$ for $i \neq j$ then index j of cover is used to represent m_i . This relationship is actually found using bipartite graph adjacency where m_i is adjacent to c_j . In this case m_i is considered as cross embedded in C .
- For the left over m_j 's, if it is equal to any of $c_{j,1}, c_{j,2}, c_{j,3}, \dots, c_{j,k}$, say $c_{j,x}$ then triplet (C, j, x) is stored in zero byte area of cover. In this case, m_j is said to be partially embedded in C .
- If still any m_j is left then it is treated as isolated node of graph and is stored in zero byte as triplet (M, j, m_j) , where M stand for message, j is index in array of message elements and m_j is the corresponding x bits. We refer the situation of embedding as explicit embedding.

The concept introduced for message hiding is shown in the schematic diagram in Figure 1 and the symbols used are described in Table 1.

The steganographic method finds *natural embedding* of the message in the selected cover. Natural embedding can be described as an embedding process in such a way that message bits can be thought of already present in the cover. For example, if j^{th} four groups $(c_{j,1}, c_{j,2}, c_{j,3}, c_{j,4})$, each of 4 bits, in the cover be

0110 0101 1001 1100

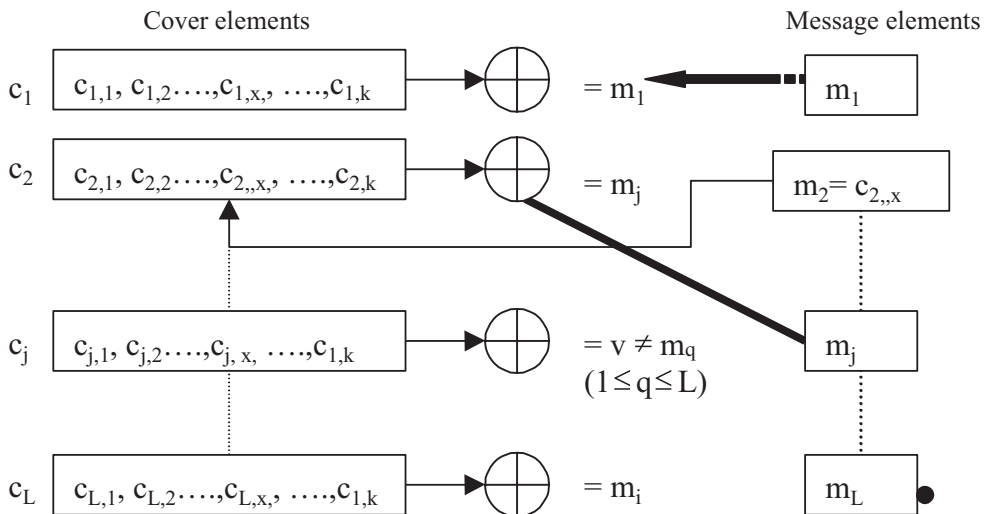






Figure 1 Schematic Diagram for Embedding

Table 1 Legends

Sr. No.	Symbol	Meaning
1.	\oplus	Addition Modulo 2^x
2.		Natural Embedding
3.		Bipartite graph edge for cross matching
4.		Partial embeddings
5.		Implies that the corresponding message element is an isolated node and has to be embedded as (M, L, m_i) .

then

$$(0110 + 0101 + 1001 + 1010) \text{ modulo } 2^4 = 1110,$$

implies that bit string

$$m_j = 1110$$

of message is naturally embedded where m_j is j^{th} 4 bits in M . If all m_j 's are not naturally embedded in the cover and say some

$$m_i = 0010 \ (i \neq r)$$

is equal to addition modulo 2^4 of some

$$(c_{r,1}, c_{r,2}, c_{r,3}, c_{r,4}) = (1000, 0100, 0011, 0011)$$

then index r of cover is used to represent m_i . Similarly, if

$$m_r = 1001$$

then it is partially embedded in

$$(c_{r,1}, c_{r,2}, c_{r,3}, c_{r,4}) = (0110, 0101, 1001, 1100)$$

at 3rd location in c_r and a triplet $(C, r, 3)$ is stored in zero bytes of cover to embed this part of message. Here C stands for cover, r for r^{th} element of message array and 3 for 3rd element in r^{th} block in C . Finally, if a m_j , say 0000, is still not found to be embedded in or related to any block of elements in C , then this part is stored as (M, j, m_j) . This too is stored in extra bytes.

Notice that no information part is directly stored but only the association and that too without disturbing any color bits of the cover. While reconstructing the message this information is used to reconstruct the message part from the cover itself. In the following section, we describe the algorithm to implement the concept by taking an example. We have used 24 bit BMP (Gonzalez & Woods, 1992; Kirkby, n.d.) image file as cover in this paper for the purpose.

3. Implementation

Given message is scanned byte by byte from left to right and its size is optimized for embedding by removing all white spaced, punctuation marks and formatting characters. Format is not so important as it can be restored (reformatted) at the receiving end. Each byte of the resultant text is unpacked into $8/x$ units. Each unit contains x bits. To illustrate the concept we have taken $x = 4$. Thus a given message of M bytes is stored in an array of size

$$L = 2M$$

We use two mask bits 00001111 and 11110000 to unpack one byte into two nibbles: most significant and least significant four bits. Let the text be “This is steganography.” After removing spaces and punctuation the resultant text is “Thisissteganography.” 19 characters of this text are divided into 38 nibbles and stored in an array TEXT [38] as shown below.

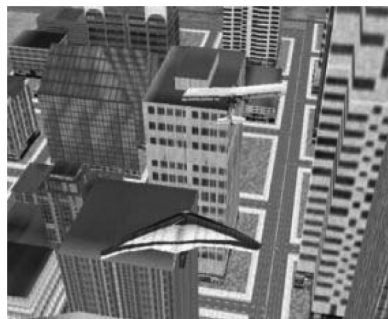
$$\text{TEXT}[38] = \{5, 4, 6, 8, 6, 9, 7, 3, 6, 9, 7, 3, 7, 3, 7, 4, 6, 5, 6, 7, 6, 1, \\ 6, \text{E}, 6, \text{F}, 6, 7, 7, 2, 6, 1, 7, 0, 6, 8, 7, 9\}$$

After transforming text into array of x bits data, it is turn to find graph relationships between image and data units. The basic information about a 24-bit BMP file is retrieved using structure that is defined in Table 3 and 4 of Section 4. For example see the image in Figure 2(a). The image is 24 bits BMP having height = 135 pixels and width = 167 pixels. The number of bytes required in every line is

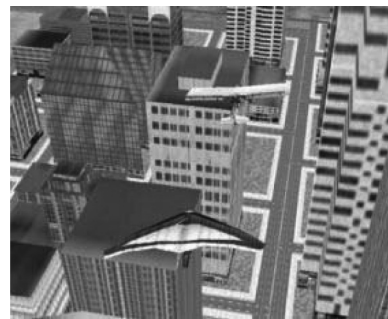
$$167 * 3 = 501$$

Next line starts at quad word boundary, so each line in the colour palette portion shall contain

$$[4 - (501 \% 4)] = 3$$



(a)



(b)

Figure 2 (a) Original Image and (b) Image after Embedding the Message

extra (or zero bytes).

The image has in total $135 \times 3 = 405$ extra bytes. The total bytes for colour pixels

$$= 167 \times 135 \times 3$$

$$= 67635 \text{ bytes} = 135270 \text{ nibbles.}$$

Therefore for every nibbles of text, we have

$$\frac{135270}{38} = 3559$$

nibbles in the image to hide. The addition modulo 2^4 of these 38 units of 3559 nibbles is stored in IMAGE [38] array and the values are as below.

$$\text{IMAGE}[38] = \{0, 6, A, 3, 9, C, 5, 0, B, 2, 8, F, 5, E, 6, 8, 7, 4, 0, 7, \\ 5, 0, 8, 9, 1, D, 0, 0, 8, 7, B, 7, B, 9, 0, C, B, 0\}$$

Now we find relationships between TEXT [38] and IMAGE [38] for the correspondence as per algorithm outlined in Section 2. Table 2 is partitioned in three parts (a), (b) and (c) to accommodate 38 indexes.

Table 2 Matching Index Table

(a)													
Text Index	1	2	3	4	5	6	7	8	9	10	11	12	13
Nibble Value	5	4	6	8	6	9	7	3	6	9	7	3	7
Image Index	7	18	2	11	2	5	17	4	2	5	17	4	17

(b)													
Text Index	14	15	16	17	18	19	20	21	22	23	24	25	26
Nibble Value	3	7	4	6	5	6	7	6	1	6	E	6	F
Image Index	4	17	18	2	7	2	20	2	25	2	4	2	12

(c)												
Text Index	27	28	29	30	31	32	33	34	35	36	37	38
Nibble Value	6	7	7	2	6	1	7	0	6	8	7	9
Image Index	2	17	17	10	2	25	17	1	2	11	17	5

First row of the Table 2 contains indexes of TEXT [] and the next row contains corresponding nibble value from TEXT. The third row contains the index from IMAGE for the corresponding nibble value. Notice the index 20 -- the natural embedding. The remaining 37 values are embedded through cross matching. The value to be embedded is

the indexes in third row i.e. (7, 18, 2, 11, 2, 5, 17, 4, 2, 5, 17, 4, 17, 4, 17, 18, 2, 7, 2, 20, 2, 25, 2, 4, 2, 12, 2, 17, 17, 10, 2, 25, 17, 1, 2, 11, 17, 5). The cover selection and embedding is described in the following section. The key of the steganography is

$$(<\text{Message length in bytes}>, x) = (19, 4).$$

We have experimented on over 200 images, and it is found that a data unit of text is either naturally embedded or cross embedded in an image if $x = 2$ or 4. The reason is the span of values that we have to find matching for. In case of $x = 2$, the possible values of data unit is $\{0, 1, 2, 3\}$ and in case of $x = 4$, it is 16 hexadecimal symbol. Thus the steps 3 and 4 of partial and explicit embedding are rarely required. Even if it is required in few cases ($< 1\%$) it is embedded in the extra bytes. The following section describes the method of retrieving information about a 24bit BMP and availability of extra bytes in it to accommodate partial and explicit embedding, if required.

4. Embedding process

BMP file format is designed to easily work with the Windows API using the same structures that Windows applications use to manipulate in-memory bitmaps. There are some variants of BMP file formats. With a little work, a single set of structures can be produced to describe all bitmap files. Table 3 illustrates the BMPFILEHEADER structure and Table 4 the BMPINFOHEADER. The structure defined in the “Box 1” is based on the BMPFILEHEADER. And the structure defined in the “Box 2” is based on the BMPINFOHEADER. Other information stored is pallet and image data. Pallet is an array of RGBQUAD structures, each of which is a color. The number of colors in the palette was specified in the biClrUsed field of the BMPINFOHEADER structure. The color Table is used only in 1-bit, 4-bit and 8-bit BMP files. Image data is a 1 dimensional array of unsigned characters, where each value is an index into the palette. In 16-bit, 24-bit and 32-bit BMP files each value in the image data section is a pixel stored in (b, g, r) format and therefore pallet is absent.

The oldest forms of the bitmap file format use 16- instead of 32-bit integers for the width and height fields of the BMPINFOHEADER structure. Fortunately, it is easy to tell when such a file is being read, since the *size* field is always 12 for those structures and greater than 12 for the newer structures. Additionally, files based on the “new” format will add an extra byte to the end of every RGBQUAD structure, in order to align it on a 4-byte boundary.

In order to ensure that the cover file is a BMP, header information from cover file is read using the following code. We then check *bitcount* field in bmpinfo structure to ensure that the BMP image file is of 24 bit. Information read in the bmpinfo structure is used to get

Table 3 BMPFILEHEADER Structure

Field	Bytes	Description
BfType	2	Bitmap identifier. Must be 'BM.'
BfSize	4	Can be set to 0 for uncompressed bitmaps, which is the kind we have.
BfReserved	2	Set to 0.
BfReserved	2	Set to 0.
BfOffbits	4	Specifies the location (in bytes) in the file of the image data. For our 24-bit bitmaps, this will be size of (BMPFILEHEADER) + size of (BMPINFOHEADER).

```
typedef struct
{
    char type[2];
    long size;
    char reserved[4];
    long offbits;
} bmpfileheader;
```

Box 1**Table 4** BMPINFOHEADER Structure

Field	Bytes	Description
BiSize	4	This is the size of the BMPINFOHEADER structure. Size of (BMPINFOHEADER).
BiWidth	4	The width of the bitmap, in pixels.
BiHeight	4	The height of the bitmap, in pixels.
BiPlanes	2	Set to 1.
BiBitCount	2	The bit depth of the bitmap. For 8-bit bitmaps, this is 8.
BiCompression	4	Our bitmaps are uncompressed, so this field is set to 0.
BiSizeImage	4	The size of the padded image, in bytes.
BiXPelsPerMeter	4	Horizontal resolution, pixels per meter, of device displaying bitmap. Not significant for us, thus set to 0.
BiYPelsPerMeter	4	Vertical resolution, in pixels per meter, of device displaying bitmap. Not significant for us, thus set to 0.
BiClrUsed	4	Indicates how many colors are in the palette.
BiClrImportant	4	Indicates how many colors are needed to display the bitmap. We set it to 0 as all colors are used.

```
typedef struct
{
    long size;
    long width;
    long int height;
    int planes;
    int bitcount;
    long compression;
    long sizeimage;
    long xpm;
    long ypm;
    long colused;
    long colimp;
} bmpinfoheader;
```

Box 2

the image size so that embedding factor can be calculated. This also provides information about the number of extra (zero) bytes availability in the image.

```

bmph = (bmpfileheader *) farcalloc (1,sizeof (bmpfileheader));
fread (bmph, sizeof (bmpfileheader), 1, fbmp);
bmpi = (bmpinfoheader *) farcalloc (1,sizeof (bmpinfoheader));
fread (bmpi, sizeof (bmpinfoheader), 1, fbmp);
Image_width = bmpi-> width;
Image_height = bmpi-> height;
extra_bytes_perline = 4 - ((Image_width*3) % 4); //% is C
language operator for mod
Total_extra_bytes_in_image = Image_hieght * extra_bytes_perline;

```

After computing the association (bipartite adjacency) between image data and text data we hide the relationship in the image. First, relationship related to the partial and explicit embedding, if there is any, is embedded in the extra byte. Number of extra bytes required is computed $\cong 5$ bytes for partial embeddings of the form (C, j, x) : 1 byte for C , 2 bytes for j and 2 bytes for x . Similarly all explicit embeddings of the form (M, j, m_j) is written in 4 bytes: 1 byte for M , 2 bytes for j and 1 byte for m_j . If number of partial and explicit embeddings is $> 1\%$ of the total data units of text, then we discard the selected cover and go for new one. We have not found even a single case of such embeddings till now. Natural and cross matching suffices the need.

Process of embedding starts from bottom. Compute number of lines required to hold the required number of bytes. If T be the total bytes required for partial and explicit embeddings, then Number of lines in image required for this is given by

$$\text{Number_of_lines_required} = \left\lceil \frac{T}{\text{extra_bytes_perline}} \right\rceil$$

Randomly select first location where partial and explicit embeddings are to be written in ascending order of j . Skip to randomly selected line from bottom and then skip to $(\text{Image_width} * 3)$ bytes from left in a line and overwrite extra bytes with value to be embedded. Then every time skip $(\text{Image_width} * 3)$ bytes from left in a line until all T bytes are written. Now to embed the relationships of natural and cross embeddings, we may explore one of the following two choices.

- (i) Send the key through one channel and store the list of indexes obtained in previous section in the extra bytes of image and send it through another channel.

- (ii) Send the image through one channel as it is and send combination of keys and list of indexes compressed in a mutually known ways through another channel.

The 38 indices are then written in the extra bytes using option (i). The stego is shown in Figure 2(b). Alternatively, the 38 indices are packed into 10 long integers of 4 bytes (1st in LSB and 4th in MSB). For the last two bytes (39th and 40th) we use NULL bytes to complete the process. Finally 10 long integers, thus computed together with key (19, 4) is sent through another channel using option (ii).

In this paper, we have restricted our discussion to 24 bit BMP files only. The same concept can be extended to other BMP file format. A message of reasonable size of say 500 characters, can very well be embedded in an image of size $\cong 800 \times 800$ 24bit BMP having 3 extra bytes per lines. If $x = 4$, then there are 1,000 indices, each requiring 2 bytes. Thus a total of 2,000 extra bytes will be needed to embed using option (i) or alternatively 500 long integers in a file can be send with key (500, 4) to retrieve the message at the recipient end.

5. Extraction process

While retrieving information from the stego, we use the given key (L, x). We compute the length m

$$= L * x \text{ bits} = \frac{m}{8} \text{ bytes or characters.}$$

Then we retrieve information about the image (stego) from its header using the structures defined in the previous section. The embedding factor k is determined from the size of message m and number of pixels in stego. Once basic information: length of message (m), embedding factor (k), size of stego in terms of number of pixels and number of extra bytes are computed, the addition modulo 2^x of the every k data units of stego is calculated. Let the computed values in sequence are:

$$V_1, V_2, V_3, \dots V_L$$

Now suppose the retrieved (or received) sequences of indices be

$$d_1, d_2, d_3, \dots d_L$$

where $1 \leq d_i \leq L$. Each d_i is an index number in the image (stego). Now values

$$V_1, V_2, V_3, \dots V_L$$

are rearranged using indices

$$d_1, d_2, d_3, \dots d_L$$

as

$$vd_1, vd_2, vd_3, \dots, vd_L$$

Now depending upon the value of x , $8/x$ values are grouped together from left to right to form string of characters without punctuation marks and white spaces. Thus retrieval is easy when key (L, x) is known.

6. Steganalysis

Most of the steganalysis techniques of images (Johnson & Jajodia, 1998b; Noto, 2001) are basically based on the detection of statistical difference introduced in the cover due to embeddings. Anonymous (2001) provides detailed study as to how to improve privacy using steganography. This graph-theoretic approach to steganography is based on the idea of sustainable embedding that neither exchanges nor overwrites any color bits. Mathematically, it is established in the following theorem that almost 100% matching for numeric value of nibbles of text is found in the in array of values from image. No statistics related to color of the image are changed at all. Additionally, since there are absolutely no visual differences between cover and stego, therefore there is no way that anybody can guess presence of stego in any communication channel. Even statistical error values like average absolute difference, mean squared error, L^p -norm, Laplacian mean squared error, peak signal to noise ratio and histogram similarity have been found in the imperceptible (excellent) embedding range because there is no difference in cover and stego.

Theorem: Probability $P(C)$ that a data unit of message is either naturally or cross matched in cover C is $\cong 1$, when data unit size is 2 or 4.

Proof: Let x be the size of smallest data unit in message then x is one from $\{2, 4, 8\}$. If it is 2, then four possible decimal values of bit string x are from the set $\{0, 1, 2, 3\}$. In case, $x = 4$, this count is 16 and in case of $x = 8$, it is 256. Obviously, the number of decimal values for bit string of length x is 2^x . Let Z be random variable indicating that a decimal value z out of 2^x is equal to a randomly selected value from array IMAGE $[L]$. Then $Z = 0, 1$. If z is not equal to the randomly selected elements from IMAGE $[L]$ then $Z = 0$ otherwise it is 1. Now the probability that z is equal to randomly selected one of the L items in IMAGE is given by

$$P(Z = 1) = \frac{1}{2^x}$$

and probability that it is not there is given by

$$P(Z = 0) = 1 - \frac{1}{2^x} = \frac{2^x - 1}{2^x}$$

Thus probability that a value z from TEXT is neither naturally matched with corresponding element in IMAGE nor cross matched with any element in IMAGE is $1 - P(C)$ and it is given by

$$1 - P(C) = \left(\frac{2^x - 1}{2^x} \right)^L$$

Therefore,

$$P(C) = 1 - \left(\frac{2^x - 1}{2^x} \right)^L$$

For an array TEXT of reasonable size $L = 100$, $P(C) \cong 0.99999$ if $x = 2$. Similarly $P(C) \cong 0.99842$ if $x = 4$, however $P(C) \cong 0.3239$ if $x = 8$ which goes up to $\cong 0.980$ for $L = 1,000$.

Let y_T be the number of different possible decimal equivalent z in TEXT corresponding to chosen value x and y_C be the corresponding number in IMAGE then following special cases arise. Note that in the example taken in the paper $y_T = 12$ and $y_C = 16$.

Case 1: When $y_C = 2^x$, then irrespective of the value of y_T $P(C) = 1$. Note that y_T can be 2^x at the maximum.

Case 2: When $y_T < 2^x$ and $y_C < 2^x$ but all different z in TEXT are also in IMAGE. Here again $P(C) = 1$.

Case 3: When $y_C < 2^x$. In this case possibly, $\exists z$ in TEXT such that z is not in IMAGE. The third step of the algorithm is used to find of partial embedding.

The probability of getting third case is < 0.001 . Even in this rare situation the algorithm uses partial embedding techniques. No explicit bit from message is written anywhere in the cover and no color bit of cover is either overwritten or exchanged.

A graph is constructed from the cover data and the secret message. A vertex in left of the bipartite graph corresponds to a smallest data unit of text and that in right corresponds to group (embedding factor) of data units from image (excluding extra byte). The L data units in left can match to a node in right in $L!$ ways. The value of L depends on message

size and value of x . The same image may carry many messages just by changing the key (L, x) . The situation when extra bytes need to be used has no effect on the statistics of the image color as these bytes do not participate in color patterns and are used just to maintain storage format of the file. In the absence of any knowledge about key and indices, any brute force approach is of order

$$(2^x)^L$$

for every guess of message size m . For a reasonable $L = 100$ and $x = 4$, it becomes of order 2^{400} even in the case of known stego.

Therefore, it is statistically infeasible for steganalyst to detect presence of a message in cover. The strength of this steganographic scheme lies in the concept of “hiding without disturbing any color bits.” In case when image is destroyed by converting from one format to another, the relationship is lost. Thus an active hacker who wants to destroy the stego may destroy it but cannot retrieve message from it (Anderson & Petitcolas, 1998; Bender et al., 1996; Weiss, 1993). In order to overcome even this menace of destroying the stego, the communicating sites may maintain a list of cover images and instead of transmitting stego, only key and compressed indices may be exchanged using option (ii) of key exchange.

7. Conclusion

We have introduced a graph theoretic approach to steganography that retains all bits that participate in the color palette of image. The method is based on exploring maximum natural and cross embedding and then finding relationship that conveys the presence of message in cover without either replacing or exchanging any bits of cover. This way the algorithm achieves sustainability. Sustainable steganography can be described as a method of hiding such that no color bits are altered. Today various digital data formats are used in steganography. Most popular among them are bmp, doc, gif, jpeg, mp3, txt and wav because of the relative ease by which redundant or noisy data can be removed from them and replaced with a hidden message. 24bit BMP image taken as cover in this paper is just for illustration. Since every cover file is simply stream of bits, the same algorithm can be applied to any image/audio format with a little modification in finding zero bytes and header information. The technique used in this paper for natural embedding can be further improved by using variable embedding factor k for adjusting its value whenever maximum natural embedding is achieved.

Steganographic research is primarily driven by the lack of strength in the cryptographic systems on its own and the desire to have complete secrecy in an open-systems environment. Redundancy is not always useless. A lot of research is required to evolve techniques to naturally embed message in digital cover media. It can be used

for the benefit of the society as well as for better administrative management by keeping any secret information secret and beyond the reach of spoiler by maintaining its utmost privacy. The rich resources of spatial data available under national spatial database project by many governments around the world may also be used for the purpose of steganography using graph-theoretic approach to steganography. “A successful steganography is one that neither disturbs nor replaces any useful bits of cover. A successful steganalysis is one that retrieves message from stego without any clue about it (Johnson et al., 2001).”

Acknowledgements

Inspiration comes from many sources. In fact it is always there. One has to look around to know the presence of something worth noticing. The paper is result of the present potential threat that the wide spread information exchange through network is facing from amateur hackers. The open world today wishes to exchange information using a public network infrastructure but in secured manner. Thankfully, the situation provides an opportunity to think about. We are also grateful to all those who have been constantly encouraging us to go for such scientific research work besides the regular work which we are doing in our respective departments. We are very grateful to the anonymous reviewers who helped in enriching the contents of the paper in improving the presentation.

References

- Anderson, R.J. and Petitcolas, F.A.P. (1998), ‘On the limits of steganography’, *IEEE Journal of Selected Areas in Communications*, Vol. 16, No. 4, pp. 474-481.
- Anonymous (1995), ‘Steganography (hidden writing)’, *WEPIN Store*, available at <http://www.wepin.com/pgp/stego.html> (accessed 26 June 2007).
- Anonymous (2001), ‘Steganography information, software and news to enhance your privacy’, *StegoArchive*, available at <http://www.StegoArchive.com> (accessed 26 June 2007).
- Bender, W., Gruhl, D., Morimoto, N. and Lu, A. (1996), ‘Techniques for data hiding’, *IBM Systems Journal*, Vol. 35, Nos. 3/4, pp. 313-336.
- Cole, E. (2003), *Hiding in Plain Sight: Steganography and the Art of Covert Communication*, Wiley, Indianapolis, IN.
- Cox, I.J., Kalker, T., Pakura, G. and Scheel, M. (2005), ‘Information transmission and steganography’, *Proceedings of the 4th International Workshop on Digital Watermarking (IWDW 2005)*, Siena, Italy, September 15-17, pp. 15-29.

- Cox, I.J., Miller, M.L., Bloom, J.A., Fridrich, J. and Kalker, T. (2007), *Digital Watermarking and Steganography* (2nd ed.), Elsevier, Burlington, MA.
- Gonzalez, R.C. and Woods, R.E. (1992), *Digital Image Processing*, Addison-Wesley, Reading, MA.
- Hetzel, S. and Mutzel, P. (2005), 'A graph-theoretic approach to steganography', *Proceedings of the 9th IFIP TC-6 TC-11 International Conference: Communications and Multimedia Security (CMS 2005)*, Salzburg, Austria, September 19-21, pp. 119-128.
- Johnson, N.F. (1995), 'Steganography', *Technical Report*, available at <http://www.jjtc.com/stegdoc/index2.html> (accessed 6 November 2007).
- Johnson, N.F., Duric, Z. and Jajodia, S. (2001), *Information Hiding: Steganography and Watermarking -- Attacks and Countermeasures*, Kluwer Academic Publishers, Norwell, MA.
- Johnson, N.F. and Jajodia, S. (1998a), 'Exploring steganography: seeing the unseen', *Computer*, Vol. 31, No. 2, pp. 26-34.
- Johnson, N.F. and Jajodia, S. (1998b), 'Steganalysis of images created using current steganography software', *Proceedings of the 2nd Information Hiding Workshop*, Portland, OR, April 14-17, pp. 273-289.
- Katzenbeisser, S. and Petitcolas, F.A.P. (2000), *Information Hiding Techniques for Steganography and Digital Watermarking*, Artech House, Boston, MA.
- Kirkby, D. (n.d.), 'Bmp format', *Arbitrary Transmission Line Calculator*, available at <http://atlc.sourceforge.net/bmp.html> (accessed 25 May 2008).
- Krinn, J. (2000), 'Introduction to steganography', *Global Information Assurance Certification*, available at <http://www.giac.org/paper/gsec/35/introduction-steganography/101757> (accessed 25 May 2008).
- Lenstra, A., Wang, X. and de Weger, B. (2005), 'Colliding X.509 certificates', *Cryptology ePrint Archive: Report 2005/067*, available at <http://eprint.iacr.org/2005/067> (accessed 23 December 2008).
- Merkle, R.C. (1990), 'A certified digital signature', in Brassard, G. (Ed.), *Lecture Notes in Computer Science: Vol. 435, Advances in Cryptology -- CRYPTO '89: Proceedings*, Springer-Verlag, Berlin, Germany, pp. 218-238.

- Noto, M. (2001), 'MP3Stego: hiding text in MP3 files', *SANS*, available at http://www.sans.org/reading_room/whitepapers/steganography/mp3stego-hiding-text-mp3-files_550 (accessed 16 February 2009).
- SANS Institute (2001), 'Encryption and exploits', *SANS Security Essentials*, Vol. 1.4, pp. 3-19.
- Schneier, B. (1996), *Applied Cryptography: Protocols, Algorithms, and Source Code in C* (2nd ed.), Wiley, New York, NY.
- Sellers, D. (2006), 'An introduction to steganography', available at <http://www.totse.com/en/privacy/encryption/163947.html> (accessed 16 February 2009).
- Stallings, W. (1999), *Cryptography and Network Security: Principles and Practice*, Prentice Hall, Upper Saddle River, NJ.
- Weiss, I. (1993), 'Review -- geometric invariants and object recognition', *International Journal of Computer Vision*, Vol. 10, No. 3, pp. 207-231.

About the authors

Vinay Kumar is a postgraduate in Mathematics and did his MCA from School of Computer and Systems Science, Jawaharlal Nehru University, New Delhi. He is working as a Scientist in National Informatics Center, MoCIT, Government of India. He has authored a book on Discrete Mathematics. His area of interest is graph algorithm, steganography, discrete mathematics, data security and privacy.

Sunil Kumar Muttoo is Reader in Department of Computer Science, University of Delhi, India. He completed his M. Tech. from IIT Kharagpur and Ph.D. from University of Delhi, India. His specialized area is coding theory, information hiding, encryption, data security and privacy. He has been Research Scholar of ISRO sponsored Project "Gross Migration Scheme."