ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 10, October 2016

Implementation of Security through hiding the Data within an Image using bytes of padding

Tanima Banerjee[1], Prof. Dr Pranam Paul[2]

MCA Final Year Student, Narula Institute of Technology, Agarpara, West Bengal, India[1]

HOD, Dept. of Computer Application, Narula Institute of Technology, Agarpara, West Bengal, India[2]

ABSTRACT: Nowadays "Steganography" becomes familiar in our day-to-day life. The term "Steganography" means hiding data within image, video, audio etc. In any communication, security is the most important issue in today's world. Data hiding is the art of hiding data for various purposes such as maintain private data, secure confidential data and so on. As increasingly more material becomes available electronically, the influence of steganography on our lives will continue to grow. With the growth of internet and network, the need for secure data transmission become more and more essential and important, as security is a major concern in internet world. Here, we introduce an algorithm of steganography which is based on hiding data by using bytes in padding. In our algorithm, we convert the secret message to binary sequence and hide the bits of embedded message into padding of cover image. In this, n byte(s) of padding of cover image are replaced by n byte(s) secret message [1, 7].

KEYWORDS: Steganography, encryption, padding, BMP Image file, steganalysis, data hiding, embedding data, information security.

I.INTRODUCTION

This paper presents a Steganography method based on the spatial domain for encoding extra information in an image by making small modifications to its padding. An image in a computer is an array of numbers that represent light intensities at various points (pixels). These pixels make up the image's raster data. Digital images are stored in either 24-bit or 8-bit per pixel files. A common image size is 640 × 480 pixels and 256 colors (or 8 bits per pixel). Such an image could contain about 300 Kb of data. Such large size images should be avoided since the attention when sending over a network or the Internet. Hence 8-bit color images, like GIF files, BMP files can be used to hide information. Here, each pixel is represented as a single byte, and the pixel's value is between 0 and 255. Grey-scale images are preferred because the shades are changed very gradually between palette entries. This increases the image's ability to hide information [9]. The most well known techniques to data hiding in images are least significant bit (LSB) substitution, and masking & filtering techniques [4]. But image manipulation can destroy the hidden information in this image. Padding bytes can be encoded with the message bits from which no change will appear in the cover image. Bitmap pixels are packed in rows. The size of each row is rounded up to a multiple of 4 byte by padding; each pixel is represented by three bytes. These bytes are for the Blue, Green, Red and the fourth byte is padding and always zero.

II.RELATED WORK

In this research base project work we reviewed many papers on steganography techniques. By reviewing these papers we observed that most of the steganography work is done in the year 2008 to 2015. In these years, LSB is the most widely used technique for steganography. Some researchers have also used the techniques like water marking, distortion technique, spatial technique, ISB, MSB in their work and provided a strong means of secure information transmission. The different security and data hiding techniques are used to implement steganography using LSB, ISB, MLSB. The Internet is an innovation technology that has become one of the most important events in modern world history [10]. The rapid growth of computer networks allowed larger files, such as digital image, text to be easily transmitted over the internet. Steganography conveys the information secretly by concealing the very existence of information in some other media files such as image, audio, video, or text files over non-secure communication channels. The information to be concealed is called the secret message or simply the secret; the content used to embed

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 10, October 2016

information is called the cover media, and the cover along with the secret is called the stego media [3]. There are many ways (methods) to hide information in images. Any text, image, or anything that can be embedded in a bit stream can be hidden in an image. Image steganography has come quite far in recent years with the development of fast, powerful graphical computers. Applying LSB technique to each byte of a 24-bit image, three bits can be encoded into each pixel, as each pixel is represented by three bytes. Applying LSB technique to each byte of an 8-bit image, only one bit can be encoded into each pixel, as each pixel is represented by one byte [4,9].

III.OBJECTIVE OF THE PROJECT

In a BMP file, the size of each row of pixel data is rounded up to a multiple of 4 byte by padding. In this algorithm, we use these padding bytes which are always zero. We converts the secret message to a binary sequence and hides the bits of embedded message into padding of cover image which occurs no change to the original image and store the message file size in 4 bytes of reserved bytes of BMP file header.

IV. ALGORITHMS

A. ALGORITHM FOR HIDING THE SECRETE DATA IN A COVER IMAGE USING THE BYTES OF PADDING

This is the simplest of the steganography methods based in the use of padding bytes. The embedding process consists of the sequential substitution of each bit of the padding of the cover image for the bit message. For its simplicity, this method can camouflage a great volume of information. The following steps illustrate how this method is used to hide the secret data in cover image which is a Bitmap Image file.

- **Algorithm**

STEP 1: Convert the message text (which will be hidden as secret data) into its binary form and print it into a text file (i.e. “b.txt”).
STEP 2: Segment the binary file into binary streams of n number of bits.
STEP 3: Take a Bitmap Image file (i.e. cover image) in which the secrete message will be hidden.
STEP 4: Read the image file (i.e. o_img). If the file is empty then print “the file cannot open “. Otherwise go to STEP 6.
STEP 5: Now, we have to copy the whole image file (i.e. o_img) into another Bitmap image file (i.e. h_img). So,

i) First, copy the whole 14 bytes bmp header file part of the file “o_img” into bmp image file “h_img”.

- This 14 bytes includes 2bytes for ASCII characters of bmp, 4bytes for size of bmp file, 4bytes for reserve bits (for future use) and 4bytes for Off bits(i.e. specifies the number of file to the starting of pixel data type).
- Convert the first 10 bytes of these 14bytes into binary streams and store it.
- Print the message file size which containing the secret message in the 4bytes of reserve bits.

ii) Then copy the 40 bytes of Bitmap header information of file “o_img” into file “h_img”.
iii) Copy the pixel data part of the image.

STEP 6: Calculate the padding portion each row pixel and store them into an integer variable (i.e pd).
STEP 7: Calculate the new width of the image after padding of each pixel row. Store the new width into another integer variable (i.e. new_w).
STEP 8: Print the header, width, height, padding and new width of image after padding.
STEP 9: Copy the pixel data as it is in the cover image. Then convert and segment the padding of each row into 8bit binary bit streams.
STEP 10: Now read the converted binary stream of secret message from text file (i.e. b.txt).
STEP 11: Replace the Bits of each segment of each padding in the row of the original image with binary message bits one by one b.txt. Store them in padding data part of another bmp image file (i.e. “h_img”). The bits of binary message will be taken one by one from MSB of binary streams.
Repeat STEP 12 until all message bits are processed.
STEP 12: After hiding all message bits image file “h_img” is containing the image with hidden data.
STEP 13: Close both files (i.e. “o_img” and “h_img”).
STEP 14: End of the process of hiding the secret data in image.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 10, October 2016

STEP 15: After hiding the secret message into image file again convert the binary format of the bmp image file into an image file into an image format and store it into file.

B. ALGORITHM FOR RETRIEVING THE HIDDEN MESSAGE FROM IMAGE

In this method we have to extract the secrete message (which is hidden in the image) from image.

- **Algorithm :**

STEP 1: Take the image file in which hidden data bits are included.
STEP 2: Check whether the image file is empty or not. If empty, then print "file cannot open". Otherwise go to STEP 3.
STEP 3: Read 14bytes bmp file header of the image file containing secrete message.
STEP 4: Segment the file header into binary bit stream of 8bit and print them into a text file (i.e. "temp.txt").
STEP 5: Close the text file (temp.txt).
STEP 6: Again open the text file (i.e. temp.txt) in read mode.
STEP 7: Calculate the bmp image file size and print the file size.
STEP 8: Read the 40 byte header information of the bmp image file (in which file the data is hidden).
STEP 9: Read the pixel part of the image file.
STEP 10: Calculate the padding of each row of pixels after replacing the message bits.
STEP 11: Calculate the new width of the image after hiding data.
STEP 12: Print the new width of image after hiding data.
STEP 13: Create a text file (i.e. c.txt).
STEP 14: Compare the new image file size (i.e. the file included hidden data) with size of original image file. If the file size is same then go to STEP 15.
STEP 15: Copy the pixel data and then convert and segment each byte of padding of the image into binary stream of 8bits and store them into an single dimension array (i.e. pda[ ]).
STEP 16: Pick bits of each segment and store them into a text file (c.txt). Repeat this step till the difference of pixel data of the image (i.e. consisting hidden message) with the pixel data of original image is caught.
STEP 17: After retrieving all hidden message bits from the image, now the text file (i.e.c.txt) is containing the secret message in form of binary bit stream of 8bits.
STEP 18: Convert the binary stream of text file (c.txt) character stream and store them into another text file (txt_fl). Now the new text file is containing the original secrete message. Message is extracted from image.

V. EXAMPLE

We have elaborated our project work through an illustrative example. We hide the secret message **"ABC"** in a cover image name **"duck.bmp"** using padding byte.

A. HIDING SECRET DATA IN COVER IMAGE USING BYTE OF PADDING

STEP 1: First each character of the text message (which will be hidden in cover image) is converted into its corresponding ASCII value.

A→ 65 B→ 66 C→ 67

STEP 2: Each ASCII value its converted into its binary form of 8 bits. And we get a binary stream for the text as below,

65→ 01000001 66→ 01000010 67→ 01000011

-------- Segment the binary bit streams.

01000001 01000010 01000011

STEP 3: Cover image **"duck.bmp"** is to be read.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 10, October 2016

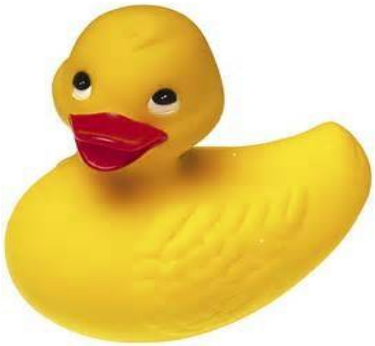


Figure V. 1(a): duck.bmp

Table V. 1(a): Equivalent ASCII the Figure V. 1(a)

156	141	59	132	…	…	0	0
111	125	99	76	…	…	0	0
88	131	95	85	…	…	0	0
84	96	126	135	…	…	0	0
59	80	102	114	…	…	0	0
…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…
155	115	105	94	…	…	…	…

STEP 4: Convert the Cover Image from decimal to binary.

Table V. 1(b): Equivalent ASCII of the Figure V.1(a)

10011100	10001101	00111011	10000100	…	…	00000000	00000000
01101111	01111101	01100011	01001100	…	…	00000000	00000000
01011000	10000011	01011111	01010101	…	…	00000000	00000000
01010100	01100000	01111110	10000111	…	…	00000000	00000000
00111011	01010000	01100110	01110010	…	…	00000000	00000000
…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…
10011011	01110011	01101001	01011110	…	…	…	…

*** Underline bytes are padding bytes.

STEP 5: Break the secret message bytes to be hidden into bits.

01000001 01000010 01000011

STEP 6: Take first row of pixels from binary table form of cover image

10011100	10001101	00111011	10000100	…	…	00000000	00000000

In this row last two bytes are used as padding.

STEP 7: So we take the first padding byte and we take the first message byte.

First message byte is 01000001 First padding byte is 00000000

Now we simply replace the padding byte with the message byte.

0	0	0	0	0	0	0	0
↓	↓	↓	↓	↓	↓	↓	↓
0	1	0	0	0	0	0	1

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

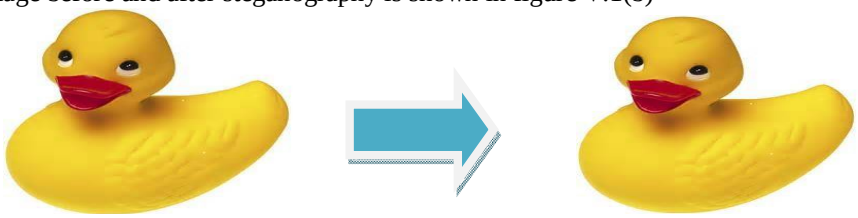
Vol. 4, Issue 10, October 2016

After this, second byte of padding from this row replace by the second message byte.
Then take the third padding byte from the second row and replace it with the third byte of message. We do this until all our message bytes are completed.

Table V.1(c): Pixel bits of cover image after hiding message.

10011100	10001101	00111011	10000100	…	…	01000001	01000010
01101111	01111101	01100011	01001100	…	…	01000011	00000000
01011000	10000011	01011111	01010101	…	…	00000000	00000000
01010100	01100000	01111110	10000111	…	…	00000000	00000000
00111011	01010000	01100110	01110010	…	…	00000000	00000000
…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…
10011011	01110011	01101001	01011110	…	…	…	…

*** Red underlines are the bytes after hiding message.
[N. B. the replaced bytes are highlighted only for shake of explanation]
Finally the cover image before and after steganography is shown in figure V.1(b)



Cover image before steganography **cover image after steganography**

Figure V. 1(b): Cover image before and after steganography

STEP 8: Number of message bit to be hidden is stored in the reserved section (4bytes) of image file header starting from 6th byte to 9th byte.

B. RETRIEVING THE HIDDEN MESSAGE FROM COVER IMAGE

STEP 1: Read the new cover image in which the message is hidden.

Figure V. 2(a) Stego image of duck.bmp

Table V. 2(a): Pixel values after steganography

156	141	59	132	…	…	65	66
111	125	99	76	…	…	67	0
88	131	95	85	…	…	0	0
84	96	126	135	…	…	0	0
59	80	102	114	…	…	0	0
…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…
155	115	105	94	…	…	…	…

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

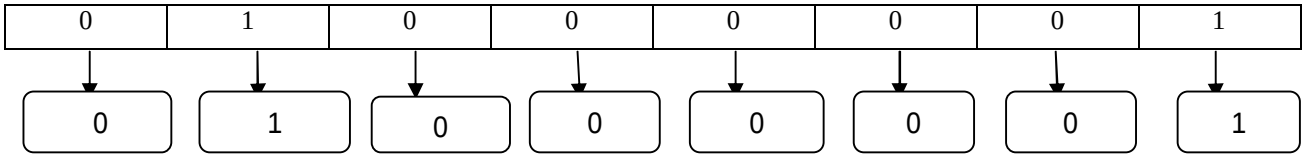
Vol. 4, Issue 10, October 2016

STEP 2: Convert the pixel values into 8 bits of binary stream.

Table V. 2(b): Binary stream of Cover image after hiding the message

10011100	10001101	00111011	10000100	…	…	01000001	01000010
01101111	01111101	01100011	01001100	…	…	01000011	00000000
01011000	10000011	01011111	01010101	…	…	00000000	00000000
01010100	01100000	01111110	10000111	…	…	00000000	00000000
00111011	01010000	01100110	01110010	…	…	00000000	00000000
…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…
10011011	01110011	01101001	01011110	…	…	…	…

STEP 3: After reading pixel value of each row, we extract only the padding portion and retrieve the message bits from there. Retrieving from the first padding byte of the first row.



-------Repeat this step for each pixel until all message bits are retrieved.

STEP 4: After repeating Step 3all the message bits are retrieved.

0 1 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 1 1

STEP 5: Segment the message bits into 8 bits of binary stream.

01000001 01000010 01000011

STEP 6: Convert each segment into their integer value respectively.

01000001 → 65 01000010 → 66 01000011 → 67

STEP 7: Convert the integers into their corresponding character value.

65→ A 66→ B 67→ C

--------The original message **"ABC"** is retrieved.

VI.RESULT AND ANALYSIS

Cover image name	Cover image File size	Text file name	Text file size	Stego image name	Stego image size	Hide message	Retrieve message
Name.bmp	60KB	text.txt	1KB	Namen.bmp	60KB	Succeed	Succeed
Smile.bmp	138KB	sml.txt	1KB	Smileb.bmp	138KB	Succeed	Succeed
Bird.bmp	153KB	bi.txt	1KB	Birdf.bmp	153KB	Succeed	Succeed
Fish.bmp	102KB	fi.txt	1KB	Fishw.bmp	102KB	Succeed	Succeed
Om.bmp	60KB	god.txt	1KB	Om1.bmp	60KB	Succeed	Succeed
Durga.bmp	1542KB	tr.txt	1KB	dur.bmp	1.5MB	Failed	Failed
Sea.bmp	81KB	s.txt	1KB	Seag.bmp	81KB	Succeed	Succeed

In our algorithm, we applied the Steganography technique for BMP image file. In order to investigate the performance of our algorithm, the algorithm is used in a number of experiments to hide text files of different sizes into various BMP image files. The minimum image pixel for width is at least 150 while the minimum image pixel for height is at least 112. Both cover and stego images are alike with the images with the images those are shown in our previous example with near-zero distortion noticeable by naked eyes. Therefore, this algorithm ia a strong yet robust algorithm to produce a stego image which will not be doubted by outsider that the image contains any secret message. If the file size exceeds

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 10, October 2016

1MB then the data hiding process fails. This failure occurs because we are using an array in our coding portion which supports up to array size 32667.

The image file format used in proposed algorithm is focused on bitmap (BMP) format. The BMP file format handles graphics files within the Microsoft Windows OS. Typically, BMP files are uncompressed, hence they are large. The advantage of using BMP files is the simplicity and wide acceptance of BMP files in Windows programs. Thus, this type of image is chosen to be used in our proposed algorithm. Since BMP image has a relatively larger size, the pixels in image are relatively larger as well. Thus, it provides more space for binary codes to be encoded within it. To increase as much as characters that can be hidden, zip technique can be used to reduce to total size of file and to enhance the security of the file. Above Table shows the comparison of different sizes in BMP image by using the proposed steganography algorithm.

VII. CONCLUSION AND FUTURE SCOPE OF PROJECT:

Our conclusion towards this project work is that we have tested the implementation of our proposed algorithm and this algorithm worked correctly for the above set of image files. From this we can assume that algorithm can correctly be implemented for various type and size of file. It will be secured. We have fulfilled our expectation only for BMP image file and it is working correctly. In further research, we are going to use more advance schemes like steganography with some hybrid cryptographic algorithm for enhancing the data security and also we try this algorithm with pixel values.

REFERENCES

[1] Vijay Kumar sharma ,Vishal Shrivastava ; "A Steganography Algorithm For Hiding Image In Image By Improved LSB Substitution By Minimize Detection " , Journal of Theoretical and Applied Information Technology 15th February 2012. Vol. 36 No.1.. ISSN: 1992-8645; E-ISSN: 1817-3195.
[2] Rosziati Ibrahim and Teoh Suk Kuan ; " Steganography Algorithm To Hide Secret Message Inside An Image ", Published: February 25, 2011.
[3] Hussein Al-Bahadili ; "A Secure Block Permutation Image Steganography Algorithm" ; International Journal on Cryptography and Information Security (IJCIS), Vol.3, No. 3, September 2013.
[4] A. E. Mustafa , A.M.F.ElGamal , M.E.ElAlmi , Ahmed.BD; "A Proposed Algorithm For Steganography In Digital Image Based on Least Significant Bit"; Research Journal Specific Education ; April. 2011.
[5] Andreas Westfeld; Steganographic Algorithm High Capacity Despite Better Steganalysis; I. S. Moskowitz (Ed.): IH 2001, LNCS 2137, pp. 289–302, 2001.
[6] Nagham Hamid, Abid Yahya, R. Badlishah Ahmad, Dheiaa Najim, Lubna Kanaan; "Steganography in image files: A survey", Australian Journal of Basic and Applied Sciences, 7(1): 35-55, 2013 ISSN 1991-8178.
[7] Dwayne Phillips; first edition of "Image Processing in C" (Copyright 1994, ISBN 0-13-104548-2) ; was published by R & D Publications.
[8] Jasleen Kour, Deepankar Verma; " Steganography Techniques –A Review Paper"; May 2014 ; ISSN: 2278-9359 (Volume-3, Issue-5).
[9] Aritra Dutta, Souvik Neogi, Prof. Dr Pranam Paul; "Implementation of security through hiding the data within an image with the help of least significant bit replacement"; 2016.
[10] Ramadhan Mstafa, Christian Bach ; "Information Hiding in Images Using
Steganography Techniques"; March 14-16, 2013.
[11] Wikipedia, the free encyclopedia.

BIOGRAPHY

Tanima Banerjee, pursuing MCA from Narula Institute of Technology, Agarpara, West Bengal, India. She completed her B.Sc. degree from Jogesh Chandra Chaudhuri College (Kolkata) under the Calcutta University. She completed her 10^{th} and 12^{th} examination from K.V. Fort William and K.V. Asansol respectively.

Dr Pranam Paul, Assistant Professor and Departmental Head, CA Department, Narula Institute of Technology (NIT), Agarpara had completed MCA in 2005. Then his carrier had been started as an academician from MCKV Institute of Technology, Liluah. Parallel, at the same time, he continued his research work. At October, 2006, National Institute of Technology (NIT), Durgapur had agreed to enroll his name as a registered Ph.D. scholar. Then he had joined Bengal College of Engineering and Technology, Durgapur. After that Dr. B. C. Roy Engineering College hired him in the MCA department at 2007. At the age of 30, he had got Ph.D. from National Institute of Technology, Durgapur. He had submitted his Ph.D. thesis only within 2 Years and 5

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 10, October 2016

Months. After completing the Ph.D., he had joined Narula Institute of Technology in Computer Application Department. He has 39 International Journal Publications among 54 accepted papers in different areas. He is also a reviewer of International Journal of Network Security (IJNS), Taiwan and International Journal of Computer Science Issue (IJCSI); **Republic of Mauritius.**

Achievements:

1. Selected his name as "Top 100 Engineers' 2011", by "International Biographical Centre", Cambridge, England.
2. Selected his name as "Outstanding 2000 Intellectuals of the 21st Century, 2012", by "International Biographical Centre", Cambridge, England.