International Journal of Advanced Science and Technology
Volume 6, May, 2009

Receiver Compatible Data Hiding in Color Image

Debnath Bhattacharyya[1], Arpita Roy[2], Pranab Roy[2], and Tai-hoon Kim[3]

[1]*Computer Science and Engineering Department,*
Heritage Institute of Technology,
Kolkata-700107, India
debnathb@gmail.com

[2]*Department of Information Technology,*
Purabi Das School of Information Technology,
Bengal Engineering and Science University,
Howrah-711103, India
arpita555@gmail.com, ronmarine14@yahoo.co.in

[3]*Hannam University, Daejeon – 306791, Korea*
taihoonn@empal.com

Abstract

Network Security for data transmission is the most vital issue in modern communication system. In this paper, we have discussed a new steganographic technique. The effectiveness of the proposed method is described through which idea of enhanced security of data can be achieved. To hide data in a binary image, no key is needed here rather this algorithm is based on the number of occurrence of 0s and 1s in data that has to hide and number of occurrence of 0s and 1s in the last bit of each pixel of binary image file. The proposed algorithm assures the security and the data hiding effect is quite invisible.

Keywords: *Data Hiding, steganography, encryption, cover image, pixel.*

1. Introduction

The amount of digital images has increased rapidly on the Internet. Image security becomes increasingly important for many applications, for example, confidential transmission, and video surveillance, military and medical applications. In steganography, the secret message is embedded into an image (or any media) called cover image, and then sent to the receiver who extracts the secret message from the cover message. After embedding the secret message, the cover image is called a stego-image. This image should not be distinguishable from the cover image, so that the attacker cannot discover any embedded message.

The security of the transformation of hidden data can be obtained by two ways: encryption and steganography. A combination of the two techniques can be used to increase the data security [1]. In encryption, the message is changed in such a way so that no data can be disclosed if it is received by an attacker [2]. Whereas in steganography [3], the secret message is embedded into an image often called cover image, and then sent to the receiver who extracts the secret message from the cover message [4]. When the secret message is embedded into cover image the it is called a stego-image. The visibility of this image should

not be distinguishable from the cover image, so that it almost becomes impossible for the attacker to discover any embedded message.

There are many techniques for encrypting data [5], which vary in their security, robustness, performance and so on. Also, there are many ways for embedding a message into another one. The most popular one is embedding a message into a colored image using LSB [6]. In this method the data is being hidden in the least significant bit of each pixel in the cover image. However, there are other known methods for hiding messages [7]. However, there are many techniques for a hiding a message in a binary image [8, 9].

This paper presents a new scheme for embedding bits. This algorithm is based on the number of occurrences of 0s & 1s in bit stream of data as well as the number of occurrences of 0s and 1s in LSB of image file that need to be modified to hide the data successfully.

2. Previous works

In Simple LSB modification bits from data that has to be hidden are put at the LSB of the cover image. Digitized images are made of pixels in which each pixel can use three bytes i.e. 24 bits. Here, three bytes are the representative of red, green and blue colors respectively.

In the LSB method the least significant bit of each byte is set to zero. Now according to the bits 0 or 1 in data LSB is being changed. If data bit is 0 then LSB is remained same & if data bit is 1 then LSB is changed to 1.For doing this modification, the image becomes a little bit lighter than the original one. Nowadays, more sophisticated approach has been taken for hiding data.

The most widely used technique to hide data [10] is the usage of the LSB [6]. Although there are several disadvantages to this approach, the relative easiness to implement it, makes it a popular method. To hide a secret message inside a image, a proper cover image is needed. Because this method uses bits of each pixel in the image, it is necessary to use a lossless compression format, otherwise the hidden information will get lost in the transformations of a lossy compression algorithm. When using a 24-bit color image, a bit of each of the red, green and blue color components can be used, so a total of 3 bits can be stored in each pixel.

While using a 24 bit image gives a relatively large amount of space to hide messages, it is also possible to use a 8 bit image as a cover source.

Because of the smaller space and different properties, 8 bit images require a more careful approach. Where 24 bit images use three bytes to represent a pixel, an 8-bit image uses only one. Changing the LSB of that byte will result in a visible change of color, as another color in the available palette will be displayed. Therefore, the cover image needs to be selected more carefully and preferably be in grayscale, as the human eye will not detect the difference between different gray values as easy as with different colors. Disadvantages of using LSB alteration are mainly in the fact that it requires a fairly large cover image to create a usable amount of hiding space.

The size of an image file, then, is directly related to the number of pixels and the granularity of the color definition. A typical 640x480 pix image using a palette of 256 colors would require a file about 307 KB in size (640 x 480 bytes), whereas a 1024 x 768 pix high-resolution 24-bit color image would result in a 2.36 MB file (1024 x 768 x 3 bytes).

The simplest approach to hiding data [11] within an image file is called least significant bit (LSB) insertion. In this method, we can take the binary representation of the hidden_data and overwrite the LSB of each byte within the cover_image. If we are using 24-bit color, the amount of change will be minimal and indiscernible to the human eye. As an example, suppose that we have three adjacent pixels (nine bytes) with the following RGB encoding:

10010101 00001101 11001001
10010110 00001111 11001010
10011111 00010000 11001011

Now suppose we want to "hide" the following 9 bits of data (the hidden data is usually compressed prior to being hidden): 101101101. If we overlay these 9 bits over the LSB of the 9 bytes above, we get the following (where bits in bold have been changed):

10010101 00001100 11001001
10010111 00001110 11001011
10011111 00010000 11001011

Note that we have successfully hidden 9 bits but at a cost of only changing 4, or roughly 50%, of the LSBs.

This description is meant only as a high-level overview. Similar methods can be applied to 8-bit color but the changes, as the reader might imagine, are more dramatic. Gray-scale images, too, are very useful for steganographic [12] purposes. One potential problem with any of these methods is that they can be found by an adversary who is looking. In addition, there are other methods besides LSB insertion with which to insert hidden information.

Without going into any detail, it is worth mentioning steganalysis [13] the art of detecting and breaking steganography [14, 15]. One form of this analysis is to examine the color palette of a graphical image. In most images, there will be a unique binary encoding of each individual color. If the image contains hidden data, however, many colors in the palette will have duplicate binary encodings since, for all practical purposes, we can't count the LSB. If the analysis of the color palette of a given file yields many duplicates, we might safely conclude that the file has hidden information.

3. Our work

We consider an algorithm that would affect the visuality of the image so little that it is almost impossible to notice any change in image by human being’s eye interpretation. Here we take an image file where each pixel is represented by red green & blue colour through three bytes i.e. 24 bits. Only the LSB of the colour blue i.e. the last bit of the pixel is used to hide the data bits. Also the number of occurrence of 0s and 1s in data bits that has to be hidden and number of occurrence of 0s and 1s in last bit of each pixel in binary image file that is needed to hide the data bits completely has been calculated and based on that approach for the decision of hiding the data would be taken. Here size of the Cover Image File should at least contain 8 times more pixels than the number of bytes in Data file.

3.1. Embedding Algorithm

1) Calculate number of bytes in data file that is supposed holding a text of the serial key of any software that is sending through internet to the receiver. Store the result in an Integer variable size_of_data.
2) Read character from data file and convert the ASCII value of the character into equivalent binary value into a 8 bit integer array suppose A in such a way so that MSB to LSB it will be stored like A[7] to A[0].

3) Calculate how many numbers of 0 s and 1s are present in each byte and store the cumulative sum into two integer variables suppose i_0 and i_1 holding total number of 0s & 1s respectively.
4) Repeat Step 1 to Step 4 until a terminating character is found. This Character can be Explicitly chosen by Sender. It may be . (dot) or any other character present in keyboard & only this character cannot be a part of data file.
5) From the Cover Image file ,Read the RGB colour of each pixel.
6) Read the character 1 by 1 from data file and convert the ASCII value of the character into equivalent binary value into a 8 bit array suppose A in such a way so that MSB to LSB it will be stored like A[7] to A[0].
7) Read the last bit of each pixel i.e. from RGB (8+8+8) bits – read the blue colour's 8 bits i.e. the last 8 bits of the pixel's colour.
8) Check this lat bit is 0 or 1 and present in each pixel and store the cumulative sum into two integer variables suppose c_0 and c_1 holding total number of 0s & 1s respectively.
9) Repeat this Step 5 to Step 8 for 8 x size_of_data times. This number of pixels actually needs to Read to hide all bits of data file.
10) If $(c_0 > c_1)$ and $(i_0 > i_1)$ or $(c_1 > c_0)$ and $(i_1 > i_0)$ the set integer variable flag to 0 Otherwise set flag to 1.
11) Now write the value of flag i.e. either 0 or 1 just at the left position of the last bit of the first pixel's information of the Stego Image file that is started to use to store the data.
12) Open a Cover Image file in read mode.
13) Open a new Stego Image file in write mode.
14) Read the header information from Cover Image File.
15) Write this header information to Stego Image File.
16) From the Cover Image file ,Read the RGB colour of each pixel.
17) Read the bit stream of data file one by one.
18) If value of flag is 0 then embed the data bit (either 0 or 1) to the last bit of colour blue of the pixel otherwise if flag is 1 then inverse the data bit embed it) to the last bit of colour blue of the pixel
19) Write the above pixel to Stego Image File.
20) Repeat the Step from 13 to 16 until all the bits of data file would be embedded in Pixels of Stego Image and after completion of embedding of data file Write the rest pixel to Stego Image File as it is in Cover Image File.

This is the required pixel information that is needed to store data. Here we count the LSB of the color blue of Cover Image & find out that the number of 1s is greater in amount than number of 0s. Again, it is also checked that number of 0s are greater in amount than number of 1s in data file. So, we need to invert data and flag is set to 1.

3.2. Extraction Algorithm

1) Open the Stego image File in read mode
2) Read the bit just before the last bit of first pixel in Stego Image File. Based on its Value set integer variable checkflag 0 or 1.
3) Read each pixel of the Stego Image file.

4) If checkflag is 0 then read the last bit of each pixel that is the LSB of colour blue and put it directly in an Array otherwise take the invert value of the last bit & Put it on Array
5) Read the each of 8 Pixel in this way & then content of the array converts into decimal Value that is actually ASCII value of hidden character.
6) If terminating character's ASCII found print nothing otherwise print the corresponding character of the calculated ASCII value.
7) Repeat Step 3 to Step 6 until decimal value of terminating character's ASCII is found.

This is the Pixel Information after Encoding .Here data is being inverted and embedded at the last bit of the blue color of each pixel of Cover Image so that the minimum modification is needed on required pixel information. As flag is set to 1 , This flag value would be stored in 1st pixel's bit which is just before the last bit of color blue of Cover Image. Extraction would be done based on this value.

4. Result

The algorithms are implemented and tested randomly, one such example is explained hereunder:

At Sender,

Data File: 01000000 00111000
Total Number of 0s at data bit stream is 12
Total Number of 1s at Last bit is 4
To store 16 bit 16 pixel of cover Image is needed.
Suppose the Pixel information of RGB colour of 16 pixels is as Figure 1.

11001000	01100001	10100001
11001011	11110000	10100001
01001111	01000001	10111101
01001111	11110000	10111001
01000000	01000000	10110000
11001111	01010000	10100001
11001111	11100001	10100000
11000000	11110000	10100001
11001111	10010000	00100000
11001111	11110000	10100001
11001111	11110000	10100001
11000011	11110000	00100000
00001111	11110000	00100000
11001111	11010000	10000001
11001111	10110110	10100001
01001111	01110000	10100001

Figure 1. Required Pixel Information before Embedding

Total Number of 0s at Last bit is 11
Total Number of 1s at Last bit is 5

According to Algorithm,

$i_0 = 12$
$i_1 = 4$
$c_0 = 5$
$c_1 = 11$

Here $i_0 > i_1$ but $c_1 > c_0$
So data bits would be inverted and it would be as follows

Data File : 01000000 00111000
Inverted Data File : 10111111 11000111

Flag's value will set to be 1.

Now the Pixel representation of Stego Image will be as Figure 2.

11001000 01100001 10100001
11001011 11110000 10100000
01001111 01000001 10111101
01001111 11110000 10111001
01000000 01000000 10110001
11001111 01010000 10100001
11001111 11100001 10100001
11000000 11110000 10100001
11001111 10010000 00100001
11001111 11110000 10100001
11001111 11110000 10100000
11000011 11110000 00100000
00001111 11110000 00100000
11001111 11010000 10000001
11001111 10110110 10100001
01001111 01110000 10100001

Figure 2. Required Pixel Information after Embedding

At Receiver,
First Pixel's bit before the last bit has read and checkflag is detected as 1,
Reading the last bit of 16 pixel data retrieved as follows:

10111111 11000111

Now According to checkflag's value above value is inverted and finally original data has been retrieved that is 01000000 00111000

Visual effects also considered and checked which are explained hereunder and shown in Figure 3 and Figure 4. No major changes in the image after embedding has been marked.

Author names and affiliations are to be centered beneath the title and printed in Times New Roman 12-point, non-boldface type. Multiple authors may be shown in a two or three-column

format, with their affiliations below their respective names. Affiliations are centered below each author name, italicized, not bold. Include e-mail addresses if possible. Follow the author information by two blank lines before main text.

Figure 3. Actual Image.

Figure 4. Stego Image.

5. Conclusion

This proposed Algorithm has following advantages:

i. Minimal change is permitted in Stego Image and normal human beings' eyes cannot catch any difference.
ii. This new algorithm does not need any secret key.

We can conclude that the suitability of steganography as a tool to conceal highly sensitive information has been discussed by using a new methodology This suggests that an image containing encrypted data can be transmitted to anybody any where across the world in a complete secured form. Downloading such image and using it for many a times will not permit any unauthorized person to share the hidden information. Thus, a new technique has been proposed to hide data in a binary image. The used algorithm is secure and the hidden information is quite invisible.

A modification to this algorithm can increase further security to hide data. Statistical analysis of 0s and 1s can be applied block by block on data bit stream to make the algorithm more complex & achieve a new methodology to hide data in a more secured way.

Acknowledgement

This work was supported by the Security Engineering Research Center, granted by the Korea Ministry of Knowledge Economy. And this work has successfully completed by the active support of Prof. Tai-hoon Kim, Hannam University, Republic of Korea.

References

[1] W. Bender, D. Gruhl, N. Morimoto, A. Lu, "Techniques for data hiding", IBM Systems Journal, Vol. 35, Issue 3-4, 1996, pp. 313-336.
[2] F. Petitcolas, R. Anderson and M. Kuhn, "Information Hiding-A Survey", Proceedings of the IEEE, Special Issue on Protection of Multimedia Content, Vol. 87, Issue 7, July 1999, pp. 1062-1078.
[3] N.F. Johnson and S. Jajodia, "Exploring steganography: seeing the unseen", Computer, Vol. 31, Isuue 2, February 1998, pp. 26-34.
[4] K. Stefan and A. Fabien, "Information hiding techniques for steganography and digital watermarking", Artech House Books, December 1999.

[5] Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, “Handbook of Applied Cryptography”, CRC Press, 1996.
[6] R. Chandramouli and Nasir Memon, "Analysis of LSB Based Image Steganography Techniques", 2001 International Conference on Image Processing, October 7-10, 2001, Thessaloniki, Greece, Vol. 3, pp. 1019-1022.
[7] (FIPS), F. I. P. S. “Data encryption standard (des)” , Federal Information Processing Standards Publication, October 25,1999.
[8] M. Wu and J. Lee, "A Novel Data Embedding Method for Two-Color Facsimile Images", International Symposium on Multimedia Information Processing, Taiwan, December 1998.
[9] M. Wu, E. Tang, B. Liu, "Data Hiding in Digital Binary Image", 2000 IEEE International Conference on Multimedia and Expo, Vol. 1, July 30-August 2, 2000, New York City, NY, USA, pp. 393-396.
[10] Hsiang-Kuang Pan, Yu-Yuan Chen, Yu-Chee Tseng, "A Secure Data Hiding Scheme for Two-Color Images", Fifth IEEE Symposium on Computers and Communications, 3-6 July, 2000, Antibes-Juan Les Pins, France, pp.750.
[11] M. U. Celik, G. Sharma, A. M. Tekalp, and E. Saber, “Reversible data hiding”, IEEE ICIP-2002, Rochester, NY, Vol. 2, September 22-25, 2002, pp. 157-160.
[12] A.D. McDonald and M.G. Kuhn, “StegFS: A Steganographic File System for Linux”, Third International Workshop on Information Hiding, LNCS, Vol. 1768, 1999, pp. 462-477.
[13] Ismail Avcibas, Nasir Memon and B. Sankur, "Steganalysis Using Image Quality Metrics", IEEE Transactions on Image Processing, Vol. 12, Issue 2, February 2003, pp. 221-229.
[14] Eric Cole ,"Hiding in Plain Sight: Steganography and the Art of Covert Communication", Wiley Publication, 2003.
[15] Petticolas, Fabien, "Information Hiding Techniques for Steganography and Digital Watermarking", Artech House Publishers, December 1999.

Authors

Debnath Bhattacharyya, M.Tech in Computer Science and Engineering from West Bengal University of Technology, Kolkata. He is working as a Faculty Member with the Computer Science and Engineering Department at Heritage Institute of Technology, Kolkata. He has 14 years of experience in Teaching and Administration. His research interests include Bio-Informatics, Image Processing and Pattern Recognition. He has 14 Years of experience in the line of Teaching and Projects. He is working towards his research, since, middle of 2006 under the supervisions of Prof. Samir Kumar Bandyopadhyay, University of Calcutta, Kolkata. He has published 43 Research Papers in International Journals and Conferences and 3 Text Books for Computer Science.

Arpita Roy, MCA from Sikkim Manipal University, India in 2007.She is currently working as a Technical Staff in the department of Computer Science and Engineering at Heritage Institute of Technology, Kolkata, India. She is pursuing M.Tech in Information Technology from Bengal Engineering & Science University, Shibpur, India. She has working experience more than 5 years .Her research interests include digital watermarking, data hiding, image processing, and authentication.

Pranab Roy has completed his graduation in Marine Engineering from MERI,Calcutta.He later worked as senior Marine Engineer with different Shipping Companies. Later he has completed his M.Tech in Information Technology from Bengal Engineering and Science University,Shibpur.Currently he is working as Lecturer and Laboratory Engineer with School of VLSI Technology, Bengal Engineering and Science University,Shibpur and is also pursueing Ph D in Physical Design Automation of VLSI Circuits.His area of interest is Digital Microfluidics,Cryptocircuits,Embedded Systems,Computer Architecture,Programming Languages and Data Structures.

Prof. Tai-hoon Kim, M.S., Ph. D (Electricity, Electronics and Computer Engineering), currently, Professor of Hannam University, Korea. His research interests include Multimedia security, security for IT Products, systems, development processes, operational environments, etc. He has 14 Years of experience in Teaching & Research. He has already got distinctive Academic Records in international levels. He has published more than 100 Research papers in International & National Journals and Conferences.