Enhancing Embedding Capacity by Compartmentalizing Pixels Using LSB Techniques in Steganography

Sanjay Bajpai [1], Dr. Kanak Saxena [2]
[1] Department of Computer applications, Lakshmi Narain College of Technology,
Bhopal (MP), INDIA. sbajpai31@gmail.com
[2] Department of computer applications, Samrat Ashok Technological Institute,
Vidisha (M.P.), INDIA. ks.pub.2011@gmail.com

Abstract

Information hiding in images has given a new dimension to provide data security during transmission. In this paper, we present an algorithm by which embedding capacity of the image is extended to a large extent. Each character of the secret message is segmented into three parts in the forms of bits to make it more secure and each pixel of the colored image is compartmentalized into three components, that is, red, green and blue to enhance the embedding capacity. Various permutations and combinations are done on the segmented parts of the message and components of the pixel to make it more robust and efficient. Our algorithm imposes no constraints regarding the type of pixel chosen for embedding or the particular region of the image. This enhanced algorithm is supported by our experimental results that show that a secret message of sufficient long length can be embedded in the colored cover image causing least distortions.

Keywords – Segmentation, Embedding capacity, Distortion, Robustness

1. Introduction

Data security had been a great challenge since a very long time. Cryptography is being used by many organizations in various applications but it suffers from a severe side effect, that is, data is visible to the intruders and it gives the clue that something important is hidden [1, 2]. This clue is sufficient to explore the things and one may succeed in it. Against this technique, steganography is getting much popularity which is the science that involves communicating secret data in an appropriate multimedia carrier, e.g., image, audio and video files. Steganography's ultimate objectives, which are undetectability, robustness (resistance to various image processing methods and compression) and capacity of the hidden data, are the main factors that separate it from related techniques such as watermarking and cryptography [3]. The term ''steganography'', meaning 'hidden writing'', originates from the Greek language. The digital medium in which the secret information is hidden is called the cover [4, 5]. If this cover is a digital image, it is called a cover image (or host image), and the image containing the secret information is called a stego-image [6]. The main purpose of steganography is to prevent malicious interceptors or attackers from discovering the existence of secret data hidden in the stego-image and keep our secret information safe.

Hiding information in other objects has a very long history. It started about 5th century BC when a slave's head was shaved for hiding the message and he was sent when his hair grew back [2, 3, 7]. Later on invisible ink and microdots in the text were used to hide messages. Then cryptography had been in use since a long time and still is in use which is a science for writing the secret message in such a way so that one cannot understand it. This method arouses the curiosity of malicious people who desire to recover or destroy. So we used the concept of steganography to hide the message in digital colored images. This paper presents a new secure and robust steganography method for data hiding in the least significant bits of the different pixel components. It is useful in various fields such as confidential communication and secret data storing, protection of data alteration, access control system for digital content distribution, media database systems etc. Other applications are audio-video synchronization, companies' safe circulation of secret data, TCP/IP packets (for instance a unique ID can be embedded into an image to analyze the network traffic of particular users) [4]. Peticolas [5] also discussed about its applications in Medical Imaging System where a privacy is considered necessary between patients' image data and their captions, for e.g., physician's name, patient's name, about the disease and other particulars. Thus, embedding the patients' information in the image could be a useful safety measure. It also supports laws of information privacy where no one can disclose or misuse the private data of an individual either intentionally or unintentionally. These data can cover a wide range such as criminal investigation reports, biological traits, bank details and in military to transfer important information hiding it from enemies etc [2, 3, 8].

The remainder of this paper is organized as follows. Section 2 briefly discusses about the related work that has been done with their analysis mentioning pros and cons. Section 3 describes the proposed method with its illustrations. In this section,

compartmentalization of pixels is illustrated together with the algorithm. Experimental results, discussions and comparisons with previous works are mentioned in section 4. Finally, the conclusion and enhancement scope are presented in section 5.

2. Related Work

Most popular and core concept of steganography is to hide the secret message in digital images by changing the least significant bits of the pixels. Most of the work is done using gray images. Some authors also used colored images to embed the data. Brief description of related work is illustrated for simplicity. Chin-Chen Chang and et al. [6] used run length (RLE) encoding technique and used gray images as the cover image. They proposed two data hiding methods incorporating both run-length encoding and modular arithmetic. The first method, BRL (hiding bitmap files by run length), is for embedding simple data with long streams of repeating bits; the second method, GRL (hiding general files by run-length), is for embedding complicated data with short streams of repeating bits. They used two consecutive pixels of the cover image for hiding the secret message. RLE works by reducing the space occupied by strings of repeating characters. Such a string is usually referred to as a run, which is recorded by two symbols that form an RLE packet: (i) the run count, denoting the number of characters in the run, and (ii) the run value, indicating the common value of the characters in the run. For example, AAAAABBBBAA is encoded as 5A4B2A in the RLE system: 5, 4, and 2 are the run counts; A and B are the run values; and 5A, 4B, and 2A are the RLE packets.

Xin Liao and et al. [9] also used gray-scale images as the cover image and gave emphasis on finding the edge pixels of the cover image to hide the secret message because changes in these locations are least visible. They partitioned the image into smooth area and edge area by calculating the differences of gray values of 4-pixel block. They used 3-bit common LSB substitution method to hide the data. For eg., if data is $000_{(2)}$ and pixel p = $1100111_{(2)}$ then p'=$1100000_{(2)}$ and then they increased the MSB of p' by 1 i.e. p' = $1101000_{(2)}$ to reduce the difference between p and p'. For messages like 010, 001, 100 etc., they did not discuss at all about how to change the MSB of p'. They used very simple LSB method for embedding but involved many calculations to find smooth and edge areas to make it more complex and thus reducing the efficiency. Rosziati Ibrahim and et al [10] proposed a SIS (Steganography Imaging System) in which they converted the secret message into binary codes and then embedded the two bits in each pixel. It simply means that four pixels will be needed to embed each character. Thus embedding capacity will not be too much. They only used the BMP images as the cover image and in some cases failures also occurred. K. Thilagam and S. Karthikeyan [11] proposed a seam carving method in which pixels of low energy or duplicating pixels are removed from the cover image and then after embedding the message image is resized to get the actual size. In this process, image is distorted and is noticeable by HVS. It is clearly visible by the figure shown in their results Fig. 1.

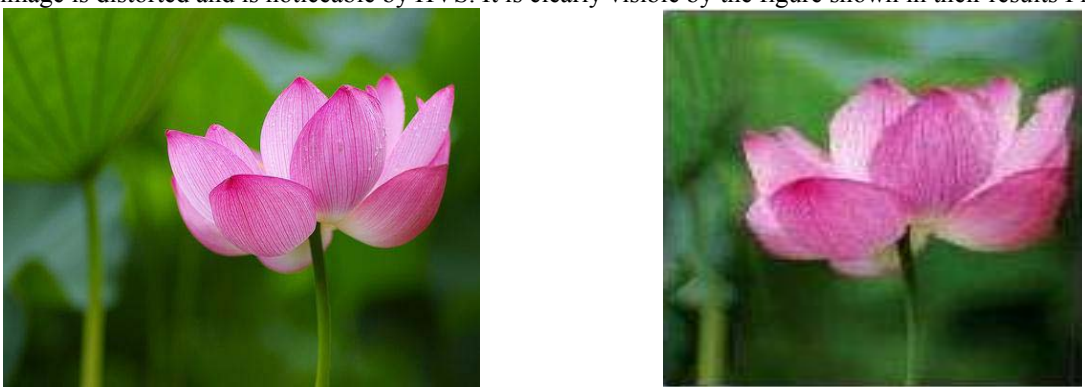


Fig. 1 Example of image distortion by seam carving

Mahmud Hasan and et al. [12] proposed a block processing mechanism in which they divided the image into a number of 4×4 non-overlapping blocks. Most Frequent Pixels (MFPs) and Second Most Frequent Pixels (SMFPs) of each block were identified and after deleting their occurrences, remaining pixels were over written by the encoded bits of the secret message. We will compare their results in section 4.

3. Proposed Method

3.1 Analysis

Most of the authors' work that we discussed in the previous section used gray images to hide the secret images. Since pixels of the gray images consist of only 8 bits so large embedding capacity cannot be expected. If one tries to embed the secret message of a little longer length then image gets distorted to a large extent [13] and it is clearly visible by the HVS (human visual system) and hence the main purpose of providing security to the data is lost. Xin Liao [9] as discussed in section 2 uses only the selected portion of gray images to embed the secret message. Some authors implemented steganography by using colored digital images. Though colored images can embed a secret message of a little larger length compared to gray images but to avoid distortion in

the stego-image, a sufficient gap among the pixels of the image should be given. If secret message is stored in the contiguous pixels then distortion in the image is clearly visible [13]. Hence length of the secret message to be embedded in colored images is also limited. Mahmud Hasan and et al. [12] also used monochrome images to embed data but they selected pixels other than the MFPs and SMFPs in a block which again resulted into low embedding capacity.

The approach that we used in our algorithm removes these existing shortcomings and it is free from any such constraints. A secret message of very long length can be embedded at any region smoothly without causing any distortion to the cover image. To show the robustness of the algorithm, we selected that region of the image where color is changing drastically. Moreover we chose the contiguous pixels of the image to embed the secret message. Each character of the secret message is embedded in all the three components of a pixel.

3.2 Description

At first all the pixels of the cover image are stored in an array P and then pixels of the selected region are picked one by one and they are segmented to achieve the red, green and blue components separately. Two keys KEY3 and KEY4 are calculated to select the region of the image and to decide the gap between the pixels that are to be identified for embedding and this again depends on the size and texture of the cover image. The segmentation process is illustrated below:

Let value of the retrieved pixel from the array: P = -10856622

Its binary conversion in the form of bits is (p): 11111111 01011010 01010111 01010010

Bits of the Red, Green and Blue components of a pixel are shown by respective colors for simplicity.

(a) Red component of the pixel is obtained by shifting it right by 16 bits and then ANDing the result by -1. After performing these operations on the above pixel, we get the resultant pixel as-
Resultant pixel (r): 00000000 00000000 00000000 01011010 [integer value 90].

(b) Green component of the pixel is obtained by shifting it right by 8 bits and then ANDing the result by -1. The resultant pixel obtained is-
Resultant pixel (g): 00000000 00000000 00000000 01010111 [integer value 87].

(c) Blue component of the pixel is obtained by ANDing it with (-1) and resultant pixel is-
Resultant pixel (b): 00000000 00000000 00000000 01010010 [integer value 82].

Red, Green and Blue components of this pixel:

R = 90 = 01011 010
G = 87 = 01010 111
B = 82 = 010100 10

Bits of the pixel components that are to be embedded depend on the KEY1 (k11, k12, k13). In this particular example, we have taken k11=3, k12=3 and k13=2. That is why last 3, 3 and 2 bits of the red, green and blue components of the pixel are shown by their respective colors that means these bits are to be over written by the bits of the character. If the character H is to be embedded then its ASCII value is obtained that is 72 and its binary notation is 010 010 00. These corresponding bits of H are overwritten in the LSBs of rgb components that again depend on the KEY2 (k21, k22, k23). In the example stated above, we have taken k21=3, k22=3 and k32=2. So the resultant components of the embedded pixel now becomes-

R = 90 = 01011 010
G = 85 = 01010 101
B = 80 = 010100 00

Red component did not change at all and green & blue are changed by 2. Maximum changes that can take place in the red and green components are by ±7 and in the blue component is by ±3. These minute changes in the components of a pixel are completely invisible and stego-image looks the same as cover image. Bits of the pixel to be embedded and bits of the character both can be permuted in any number of ways to make it more secure.

3.3 Algorithm (Embedding Process)

1. Create one dimensional **integer** array of size M × N and store all the pixels of the cover image in it where M × N is the size of the cover image.
2. Find the length of the secret message and let it be P. Create a **byte** array T of size P and store ASCII code of all the characters of the secret message in it**.**
3. Calculate the keys KEY3 and KEY4 by incorporating the size and texture of the cover image and length of the secret message. KEY3 fixes the starting position of the cover image and KEY4 fixes the gap between two pixels.
4. Read the code from the array T and segment 8 bits of the character into 3 parts using KEY2 (k21, k22, k23).
5. Select a pixel using keys KEY3 and KEY4, and bifurcate components of a pixel into three fragments that is Red, Green and Blue**.**

6. Overwrite the LSBs of Red, Green and Blue component of a pixel using the KEY1 (k11, k12, k13).
7. Repeat steps 4 to 6 until array T becomes empty.
8. Rejoin all the pixels of the array to form the stego-image.

3.4 Algorithm (Extraction Process)

1. Extract all the keys KEY1, KEY2, KEY3, KEY4 and P, the length of the message.
2. Read pixels from the stego-image and accumulate in **byte** array T.
3. Pick up the pixel from the array T by using KEY3 and KEY4.
4. Extract bits from the selected pixel using KEY1 and KEY2.
5. Combine these bits and store in array temp.
6. Repeat steps 3 to 6 until P pixels are selected from array T
7. Write contents of array temp as string, the secret message.

4. Experimental Results

We have tested our results by taking 20 different images of type jpg for hiding the data out of which 5 images are shown in the result. Each image is of dimension 450×338 pixels. Their names are img1o, img2o, img3o, img4o, img5o (o for original). To show the robustness of algorithm we have chosen that area of the image for hiding the message where color is changing drastically. That is why we have chosen the middle portion of the images img1o, img2o, img3o and upper portion of the image img4o and lower portion of the image img5o because these are the regions where color is changing drastically, as can be seen in the cover images Fig. 2(a), Fig. 3(a), Fig. 4(a), Fig. 5(a) and Fig .6(a).

Data is stored from 170th row in images img1o, img2o, img3o, from 80th row in img4o and from 200th row in img5o. Message is stored in continuous pixels and length of the message is varied every time. Original images are img1o, img2o, img3o, img4o, img5o and images with hidden message (i.e. stego-images) are img1s, img2s, img3s, img4s and img5s as shown in Fig. 2(b) to Fig. 6(b). After visualizing both the cover images and stego images very minutely, no one is able to notice any differences between them but these stego images are hiding a secret message of more than 1500 characters that is shown in Table 1.

Result images for comparison –

Fig. 2 (a) img1o (Cover image)

(b) img1s (Stego image)

Fig. 3 (a) img2o (Cover image)

(b) img2s (Stego image)

Fig. 4 (a) img3o (Cover image)

(b) img3s (Stego image)

Fig. 5 (a) img4o (Cover image)

(b) img4s (Stego image)

Fig. 6 (a) img5o (Cover image) (b) img5s (Stego image)

Maximum performance given by our algorithm is compared with the algorithms proposed by Rosziati Ibrahim [10] and Mahmud Hasan [12] which clearly shows that our results are better and these are shown in Table 2. Rosziati Ibrahim et al. hided only two bits out of 8 bits of a character per pixel in their SIS using LSB technique, so the embedding capacity is already less and moreover nearly half of their cover image could not be converted into stego-image and failing in both the cases of hiding and retrieving the message. They also used bmp images as the cover images which already take too much space. In our case, we are storing 8 bits of a character per pixel by compartmentalizing it into its components and in no case failure occurred. All the cover images are converted into stego image and message is extracted as well. Mahmud Hasan et al. in their block processing approach are left with only 32 bits out of 128 bits in which characters of 7 bits are to be embedded. On an average, 4.57 characters can be embedded in 32 bits and considering the complete block of 4×4, 3.50 pixels are required to embed one character.

Table 1

Results of the Proposed Algorithm

Cover Image		**Number of characters**	**Stego Image**		**Message Hidden**	**Message Retrieved**
Size	Type		Size	Type		
70.8 KB	jpg	1537	363 KB	png	√	√
76.1 KB	jpg	1685	386 KB	png	√	√
57.2 KB	jpg	1425	275 KB	png	√	√
58.9KB	jpg	1773	259 KB	png	√	√
67.4 KB	jpg	1953	344 KB	png	√	√

Table 2

Comparative Performance Measurements

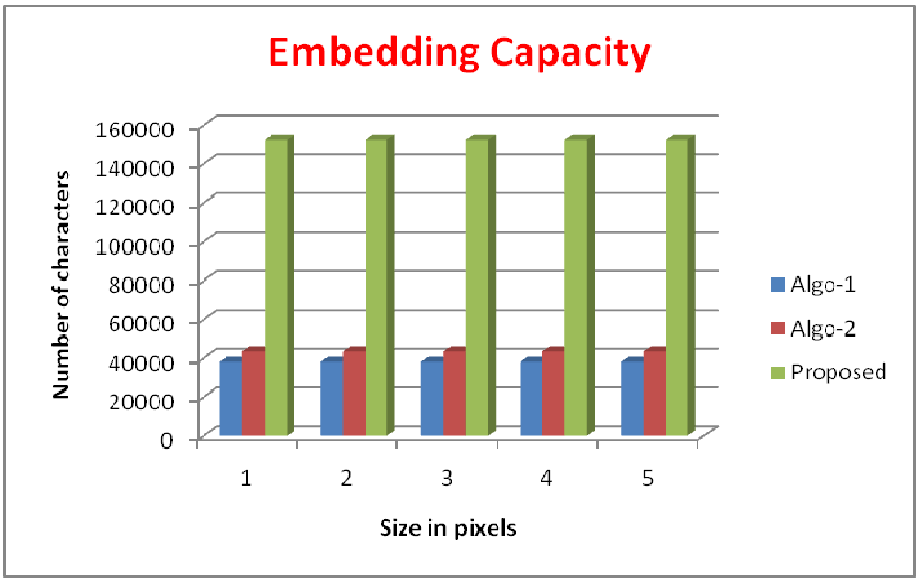
Cover Image			**SIS LSB Technique**		**Block Processing Approach**		**Proposed Method**	
Size in KB	Type	Dimension in pixels	Number of characters	Stego image	Number of characters	Stego image	Number of characters	Stego image
70.8	jpg	450×338	38025	349	43457	355	152100	363

76.1	jpg	450×338	38025	Failed	43457	396	152100	386
57.2	jpg	450×338	38025	284	43457	279	152100	275
58.9	jpg	450×338	38025	Failed	43457	252	152100	259
67.4	jpg	450×338	38025	347	43457	351	152100	344

5. Conclusion and Scope

In this paper, we approached some of the existing algorithms, such as, mentioned by Dr. Mohammed Abbas [8], Rosziati Ibrahim and et al. [10] and Mahmud Hasan and et al. [12] and devised our enhanced algorithm by which embedding capacity of the cover image increased to a great extent and at the same time making the least distortions. The number of characters of the secret message can be equal to the number of pixels in the cover image. If we take a cover image of dimension 450×338, as we have taken in our example, then it can hide at the most 1,52,100 characters without causing any distortions. Here we have taken cover image of type jpg only but we can take it of any type. Table 1 shows our result and Table 2 shows the comparison with other algorithms and it clearly show that embedding capacity has increased to a large extent but we have not compromised with the security and used 4 different keys and thus we can say that we have enhanced the algorithm.

Our stego image is always of type 'png' and we have to improve so that cover image can be converted into any type. One more point is noticed that size of stego image is much higher than the size of the cover image and it has to be taken care of. We are further trying to increase the embedding capacity by using any robust compressing mechanism.



Graph – Showing the Performance Measurement

6. References

[1] M. Sitaram Prasad, S. Naganjaneyulu, A Novel information hiding technique for security by using Image Steganography, Journal of Theoretical and Applied Information Technology (2005-2009), pp. 35-39, 2009.

[2] Sanjay Bajpai, Kanak Saxena, Techniques of Steganography for Securing Information: A Survey, International Journal on Emerging Technologies **3**(1), ISSN No. (Print): 0975-8364, pp. 48-54, 15 April, 2012.

[3] Abbas Cheddad, Joan Condell, Kevin Curran, Paul Mc Kevitt, Digital image steganography: Survey and analysis of current methods, Signal processing 90 (2010), pp. 727-752, 18 August, 2009.

[4] Jamil, T., Steganography: The art of hiding information is plain sight, IEEE Potentials, 1999.

[5] Min-Wen Chao, Chao-hung Lin, Cheng-Wei Yu, and Tong-Yee Lee, A High Capacity 3D Steganography Algorithm, IEEE Transactions on Visualization and Computer Graphics, Vol 20 No. 3, pp. 1-11, June, 2009.

[6] Chin-Chen Chang, Chih-Yang Lin, Yu-Zheng Wang, New image steganographic methods using run-length approach, International Journal of Information Sciences (176), Elsevier, pp. 3393-3408, 03 February, 2006.

[7] P. Moulin, R. Koetter, Data-hiding codes, Proceedings of the IEEE 93 (12), pp. 2083–2126, 2005.

[8] Dr. Mohammed Abbas Fadhil Al-Husainy, Message Segmentation to Enhance the Security of LSB Image Steganography, International Journal of Advanced Computer Science and Applications, Vol. 3, No. 3, pp. 57-62, 2012.

[9] Xin Liao, Qiao-yan Wen, Jie Zhang, A steganographic method for digital images with four-pixel differencing and modified LSB substitution, Journal of Vis. Commun. Image R 22 (2011), Elsevier, pp. 1-8, 22 August, 2010.

[10] Rosziati Ibrahim and Teoh Suk Kuan, Steganography algorithm to hide secret message inside an image, Journal of Computer Technology and Application 2 (2011), pp. 102-108, 25 February, 2011.

[11] K. Thilagam, S. Karthikeyan, Optimized Image Resizing using Piecewise Seam Carving, International Journal of Computer Applications (0975 – 8887) Volume 42– No. 14, pp. 24-30, 14 March, 2012.

[12] Mahmud Hasan, Kamruddin Md. Nur, Tanzeem Bin Noor, A Novel Compressed Domain Technique of Reversible Steganography, International Journal of Advanced Research in Computer Science and Software Engineering ISSN: 2277 128X, pp. 1-6, 03-March, 2012.

[13] Mehdi Kharrazi, Husrev T. Sencar, Nasir Memon, Performance study of common image steganography and steganalysis techniques, Journal of Electronic Imaging 15(4), 041104, pp. 1-16, 18 December, 2006.

REFERENCES

[1] Francois Raphael and Lison Pierre. .May 04, 2005, *Probabilistic Language Modeling with N-grams*, Artificial Intelligence Seminar,pp 1-43

[2] Josan Gurpreet Singh and Lehal Gurpreet Singh. *Size of N for Word Sense Disambiguation using N gram model for Punjabi Language*, Yadwindra College of Engg., Talwandi Sabo,pp 1-11

[3] Daniel Jurafsky and James H. Martin. Speech and Language Processing: *An introduction to natural language processing ,computational linguistics, and speech recognition,*pp ,189-230

[4] Marti U.V. & Bunke H. 2001, *On the influence of vocabulary size and language models in unconstrained handwritten textrecognition*, Proc. 6th Int. Conference on Document Analysisand Recognition, pp 260 – 265.

[5] Sinha Manish et. al. 2004. *Hindi Word Sense Disambiguation,* International Symposium on Machine Translation, NLP& Translation Support Systems, Delhi, India, pp 230-237.

[6]https://en.wikipedia.org/wiki/Natural_language_processing Accessed on 27/3/2014

[7] http://en.wikipedia.org/wiki/N-gram accessed on Accessed on 27/3/2014