IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 4 Issue 3, March 2017
ISSN (Online) 2348 – 7968 | Impact Factor (2016) – 5.264
www.ijiset.com

A Secured and Robust Image Steganographic Technique to Map a Large Text File without Stuffing Data Bits

K.S.Sadasiva rao[1], Dr A.Damodaram[2]

[1] Associate Professor, Dept of CSE, Sri Indu Institute of Engg. & Tech., Hyderabad, India

[2] Professor of CSE, School of Information Technology,
JNT University, Hyderabad &
The Vice Chancellor, SV University, Tirupathi, India

Abstract

Steganography is the process of hiding data bits on cover or carrier file. The carrier file may be text file, image file, audio file or video file etc. The purpose of the steganography is to hide the existence of the information from the attacker [2][3]. If that carrier file is an image file, then that technique is called Image steganography[1]. If color image is used as a carrier file to embed data bits, then that type of steganographic technique is called as color image steganography [7]. Most of the steganographic algorithms are using bit replacement algorithms. While embedding information in an image, a better balance is required in between the increasing embedding capacity and in reducing the stego-image distortion. In this proposed work, neither single bit of data was embedded in the picture nor the cover file was transmitted on the channel. In this proposed work, instead of embedding the data bits directly on carrier color image, original text data bits will be mapped on the carrier file with the help of indexed key vectors, which are mapped with corresponding red, green and blue planes. Instead of embedding the data bits directly on the image, a comparison study implemented with each and every bit of the pixel/plane against the actual text data bit need to embed. If they are equal the corresponding positions are additively grouped and stored as big integer (representing the positions) in the respective cell of the mapping table. At the destination end the respective position bits are collected back to frame the original text.

The original data bits get transmitted without embedding on the carrier or cover image, but there is a logic with which we are transferring the data bits on the carrier file. So even any unauthorized user finds this cover image, it will not give any information as it is plain cover image without any data was stuffed and image quality measuring parameters are also not modified. Hence this technique is a very efficient technique. At least a single bit is not modified in the cover image, so we could not be able to identify steganographic process with any powerful steganalysis techniques, hence it is robust. Hence this algorithm is named as A secured and robust image steganographic technique to map a large text file without stuffing data bits.

Keywords : *Bit replacement algorithms, Steganography, Steganalysis, mapping technique, indexed key vectors.*

1. Introduction

Steganography is the process of embedding the data bits in the particular positions of the pixels in the carrier file. Spatial domain and Transform domain techniques are used to implement steganographic process. Generally in spatial domain, RGB planes are used to stuff the data bits at least significant bit positions of the pixel [9]. Hence three bits can be stuffed in each pixel among the three planes [6]. But in the transform domain, the pixel values are converted into the transform coefficients, and then those transformed coefficients to be used to modify the data. Hence either spatial domain or transform domain algorithms will replace the cover image bits with the original message bits using least significant bit algorithm or some other variations on those algorithms [10]. A powerful commercial steganalysis technique may detect the image stegnographic process even though it is not identified by human necked eye. If the carrier file used is a color image file, then that type of technique is called as color image steganographic technique. The file on which data bits will be stuffed is called as carrier file or cover file. After embedding the data bits on the cover image, the cover image is called as stego image. Most of the image steganographic algorithms will be using the Least Significant Bit (LSB) method, because it will not give much affect on the quality of the image.

2. Proposed System at Sender Side

Hence almost all the steganographic algorithms either spatial domain or transform domain are using bit replacement algorithms. Steganalysis techniques are used to find whether the data bits are stuffed in the carrier image. These Steganalysis techniques are mostly working on the statistical characteristics of an image. Whenever there is partial or slight change in the carrier image, those changes can be detected by commercial steganlaysis tools, even though it is not identified by human necked eye. Most of the LSB based steganographic algorithms will be getting PSNR value is much more than 20 (i.e., if this value is greater than 20, human eye cannot detect the difference between the cover image and stego image). Here in this proposed algorithm, carrier image is transmitted as it is to the destination end. The original data bits of a text file is going to be compared with the each pixel / plane bits. First, each character of the text file is identified with its ASCII code and this code again converted into binary form. This binary information is compared with the each pixel / plane bit of the image. If any bit is equal, immediately it's position is going to be recorded in index set. Sequentially all the 8 bits of the character is compared with RGB planes bits and recorded the matching positions as a big continuous positive integer, at the corresponding pixel/plane matching vector. In such a way all the binary information of the text file,

mapped in the corresponding pixel/plane mapping vectors.

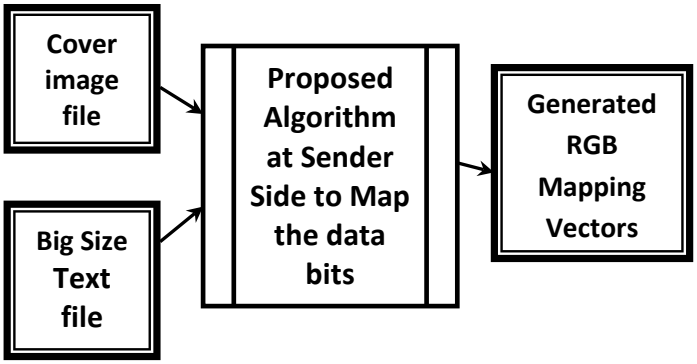

Fig 1: Proposed System on sender side

Algorithm at Sender Side :-

1. *Read a cover image which is a color image.*
2. *Display the cover image.*
3. *Extract the three planes (red, green and blue) and convert into matrices.*
4. *Read the original message bits from any input data file.*
5. *For every plane, do the following:*
   a. *Read the data bit and the pixel/plane bit serially from the MSB bit of the corresponding plane.*
   b. *Compare data bit with each pixel/plane bit.*
      i. *If these bits are equal, record the position of the pixel/plane bit at mapping vector and continue with the next bit of the data .*
      ii. *If these bits are not equal, then continue with the next bit of the pixel/plane.*
      iii. *Repeat the step 'i & ii' until all the 8 bits are matching with pixel/plane bits.`.*
      iv. *All the identified matching pixel position are recorded additively at the corresponding pixel/plane mapping vector.*
   c. *Construct the Mapping matrixes for all RGB pixel/planes.*
6. *Send the mapping vectors to the destination end*
7. *Stop the process.*

3. Proposed System at Receiver Side

At the receiver end, the proposed algorithm uses cover image file and generated RGB mapping vectors as input. The cover image file can be transmitted from source to destination end or it can be maintained by the repositories at both ends. If only the generated RGB mapping vectors are transmitted on the channel, that improves the security level, as it does not contains any information about cover image / embedded data. The Proposed algorithm reconstructs the original text file data bits by implementing the mapping vectors on the cover image file.

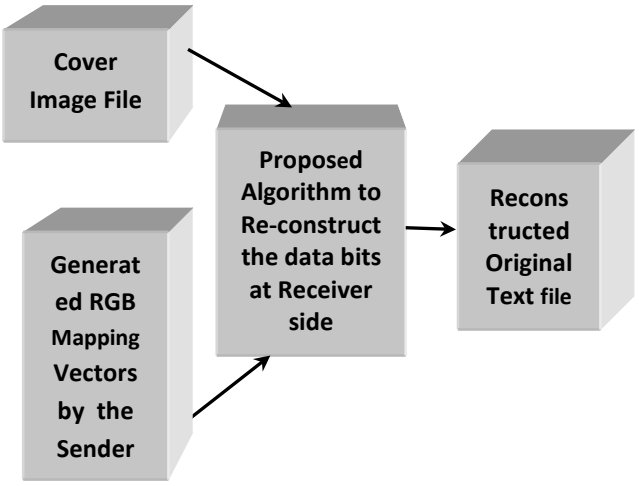

Fig 2: Proposed System on receiver side

Algorithm at Receiver Side :-

1. *Read a cover image which is a color image.*
2. *Display the cover image.*
3. *Extract the three Mapping vectors of pixel/planes (red, green and blue) .*
4. *For every plane, do the following process*
   a. *Read the key vector.*
   b. *Each digit of the corresponding cell, indicates the respective pixel/plane binary digit value as the original data bit.*
   c. *Collect all the bits, indexed by the mapping vectors (RGB), from the corresponding pixel/plane of the cover image.*
   d. *Repeat the above process for the pixel/planes to collect all the 8 bits, to frame a character of the text file.*
5. *Extract all the data bits of the text file on the above mentioned way.*
6. *Display the cover image and original message of the text file.*
7. *Stop the process.*

4. Illustration for Mapping the bits with the Color Image

In this process a separate vector is considered for each Red, Green and Blue pixel/planes. Each and every character of the actual text file was converted into

its equal-ant ASCII code subsequently into its binary format of 8 bits. All the 8 bits of a character is going to be compared with the RGB planes vectors alternatively. Initially R(1,1) bits are taken, then G(1,1) bits and finally B(1,1) are considered for comparison study. Then the next character is again compared with the RGB bits of the next pixel in the same way. Whenever a bit is matching the corresponding location of the bit is stored in the mapping vector of the corresponding pixel/plane location. If more bits are matching then an additive number is stored as a big integer, which shows each and every digit of that big number as a location identified with matching bit. While reconstructing the bits at receiver end, the bits are collected serially with the help of mapping vectors of RGB planes alternatively. Every 8 bits group used to reconstruct a character and all the mapping vectors are sharing their respective matching locations in reconstructing the total text file. A maximum of (256 * 256) size image file can embed (256 * 256 * 8 * 3) / 8 = 1,96,608 bytes/chars can be embedded i.e. at max 1,96,608 * 8 = 15,72,864 bits of information can embed. As all the text file bits are not going to match serially with the RGB vectors bit values, in worst case also at least it can embed 1/2 of the max capacity i.e. 7,86,432 bits. This proposed algorithm shows a maximum level of embedding capacity without any distortion in the image quality.

Example :-

INPUT TEXT	T	H	I	S		I	S		A		T	E	S	T
ASCII CODES	84	72	73	83	32	73	83	32	65	32	84	69	83	84

Tab 1. A Sample text input to map on image

Char	Ascii	Binary Code							
T	84	0	1	0	1	0	1	0	0
H	72	0	1	0	0	1	0	0	0
I	73	0	1	0	0	1	0	0	1
S	83	0	1	0	1	0	0	1	1
	32	0	0	1	0	0	0	0	0
I	73	0	1	0	0	1	0	0	1
S	83	0	1	0	1	0	0	1	1
	32	0	0	1	0	0	0	0	0
A	65	0	1	0	0	0	0	0	1
	32	0	0	1	0	0	0	0	0
T	84	0	1	0	1	0	1	0	0
E	69	0	1	0	0	0	1	0	1
S	83	0	1	0	1	0	0	1	1
T	84	0	1	0	1	0	1	0	0

Tab 2. Corresponding Ascii and Binary data for text

Pixel/plane	Ascii	Binary Code
(1,1) R	61	0 0 1 1 1 1 0 1
(1,1) G	49	0 0 1 1 0 0 0 1
(1,1) B	37	0 0 1 0 0 1 0 1
(1,2) R	61	0 0 1 1 1 1 0 1
(1,2) G	49	0 0 1 1 0 0 0 1
(1,2) B	37	0 0 1 0 0 1 0 1
(1,3) R	61	0 0 1 1 1 1 0 1
(1,3) G	49	0 0 1 1 0 0 0 1
(1,3) B	37	0 0 1 0 0 1 0 1
(1,4) R	62	0 0 1 1 1 1 1 0
(1,4) G	50	0 0 1 1 0 0 1 0
(1,4) B	38	0 0 1 0 0 1 1 0
(1,5) R	62	0 0 1 1 1 1 1 0
(1,5) G	50	0 0 1 1 0 0 1 0
(1,5) B	38	0 0 1 0 0 1 1 0
(1,6) R	62	0 0 1 1 1 1 1 0
(1,6) G	50	0 0 1 1 0 0 1 0
(1,6) B	38	0 0 1 0 0 1 1 0
(1,7) R	61	0 0 1 1 1 1 0 1
(1,7) G	49	0 0 1 1 0 0 0 1
(1,7) B	37	0 0 1 0 0 1 0 1
(1,8) R	61	0 0 1 1 1 1 0 1
(1,8) G	49	0 0 1 1 0 0 0 1
(1,8) B	37	0 0 1 0 0 1 0 1
(1,9) R	62	0 0 1 1 1 1 1 0
(1,9) G	50	0 0 1 1 0 0 1 0
(1,9) B	38	0 0 1 0 0 1 1 0
(1,10) R	62	0 0 1 1 1 1 1 0
(1,10) G	50	0 0 1 1 0 0 1 0
(1,10) B	38	0 0 1 0 0 1 1 0

Tab 3. Binary Representation of Image (TIGER)
(Using First 10 Pixels of RGB Planes)

Pixel Location	Original bits Match location
(1,1)	**T** { 8731
(1,2)	21 }
(1.3)	**S** { 8731
(1,4)	21 }
(1,5)	**S** { 831
(1,6)	{ 8321
(1,7)	321 }
(1,8)	**T** { 8731
(1,9)	31 }
(1,10)	**T** { 831

Tab 4. Mapping Vector for Red Plane

Pixel Location	Original bits Match location
(1,1)	6531 }
(1,2)	**I** { 86531
(1.3)	4321 }
(1,4)	**I** { 876531
(1,5)	7653
(1,6)	6521 }
(1,7)	{ 765321
(1,8)	6531 }
(1,9)	**S** { 87531
(1,10)	8753

Tab 5. Mapping Vector for Green Plane

Pixel Location	Original bits Match location
(1,1)	**H** { 765431
(1,2)	321 }
(1,3)	{ 754321
(1,4)	31 }
(1,5)	3 }
(1,6)	**A** { 85431
(1,7)	21 }
(1,8)	**E** { 875431
(1,9)	631 }
(1,10)	1 }

Tab 6. Mapping Vector for Blue Plane

5. Results

The following are the results of an efficient and robust image steganographic technique without stuffing data bits. Here no single bit of data embedded in the cover image, all the text data has been mapped with the help of mapping vectors.

Fig 3: Cover Image used at Sender side

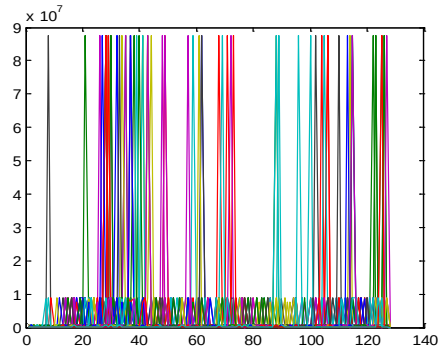

Fig 4 : Red key vector histogram

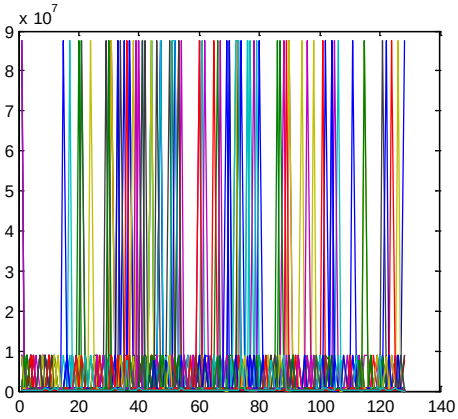

Fig 5 : Green key vector histogram

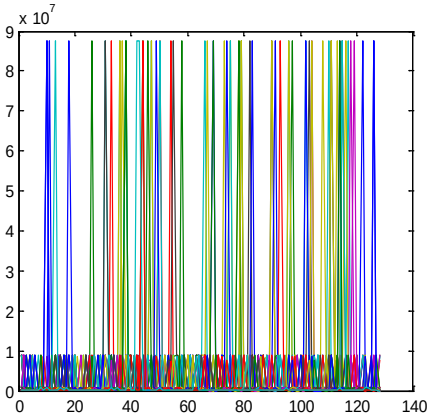

Fig 6 : Blue key vector histogram

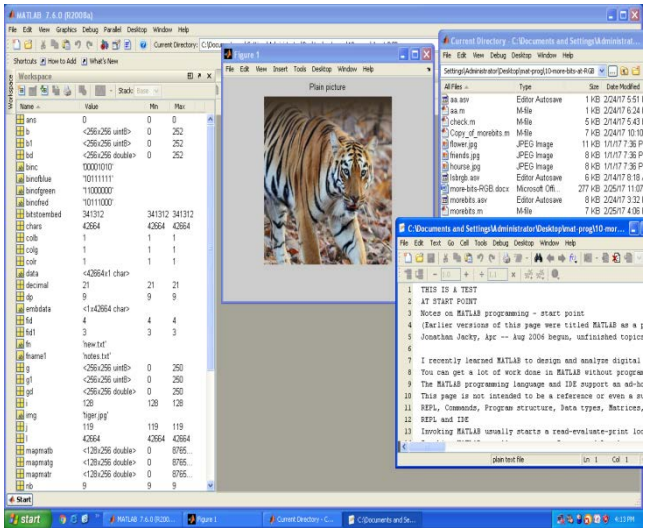

Fig 7 : Output Screen

7. Conclusion

In our proposed system, we have constructed and sent the Mapping vectors for all the three planes red, green and blue along with the plain cover image. We can also maintain the Cover image at the repository of both sender and receiver sides, so that we can avoid the image transmission and it also improves the security. The Mapping key vectors are constructed with proposed algorithm on sender side and transmitted through the channel to the receiver. At the Receiver end, the mapping key vectors and the Cover image file at the repository are used as input to the proposed algorithm at receiver-end. The proposed algorithm recollects all the data bits represented by the mapping vector and makes each 8 bit group. Each 8 bits group useful in Re-constructing the original text file. In this proposed algorithm, we are using a logic with which we can transfer a big text file data bits without embedding on the cover image.

REFERENCES

[1] Niels Provos and Peter Honeyman, Hide and Seek: An Introduction to Steganography, IEEE Security & Privacy, 2003.
[2] Saiful Islam, Mangat R Modi, Phalguni Gupta, Edge-based Image Steganography, Springer 2014.
[3] Anil Kumar, Rohini Sharma, A Secure Steganography Based on RSA Algorithm and Hash-LSB Technique, International Journal of Advanced Research in Computer Science and Software Engineering, July, 2013.
[4] Deepesh Rawat, Vijaya Bhandari, A Steganography Technique for Hiding Image in an Image using LSB Method for 24 Bit Color Image, International Journal of Computer Applications, Volume 64, 2013.
[5] Abbas Cheddad, Joan Condell, Kevin Curran, Paul Mc Kevitt, Digital Image Steganography: Survey and Analysis of Current Methods, Elsevier, 2010.
[6] Wien Hong And Tung-Shou Chen, A Novel Data Embedding Method Using Adaptive Pixel Pair Matching, IEEE Transactions On Information Forensics And Security, Volume 7, No. 1, February 2012.
[7] Niel F Johnson, Sushil Jajodia, Exploring Steganography: Seeing The Unseen, IEEE, 1998.
[8] Gurmeet kaur, Aarthi kochhar, Transform domain analysis of image steganography, International journal for science and Engineering Technologies with latest trends, 2013.
[9] Hemalatha S, U Dinesh Acharya, Renuka A, Priya R. Kamath, A secure color image steganography in transform domain, International journal on Cryptography and Information Security, Vol. 3, No. 1, March 2013.
[10] Suchitra B, Priya M, Raju J, Image steganography based on DCT algorithm for data hiding, International journal of Advanced Research in Compuer engineering and technology, vol. 2, Nov 2013.