IOSR Journal of Computer Engineering (IOSRJCE)
ISSN: 2278-0661 Volume 4, Issue 1 (Sep-Oct. 2012), PP 37-44
www.iosrjournals.org

Hide and Seek: Embedding Audio into RGB 24-bit Color Image Sporadically Using Linked List Concepts

Dr. Emad S. Othman,
Senior Member IEEE - Region 8, High Institute for Computers and Information Systems, AL-Shorouk Academy, Cairo – Egypt,

Abstract: *Although people have hidden secrets in plain sight now called steganography throughout the ages, the recent growth in computational power and technology has propelled it to the forefront of today's security techniques. Steganography is the art and science of hiding communication; a steganographic system thus embeds hidden content in unremarkable cover media so as not to arouse an eavesdropper's suspicion. In the past, people used hidden tattoos or invisible ink to convey steganographic content. Today, computer and network technologies provide easy to use communication channels for steganography. The embedding process creates a stego medium by replacing bits with data from the hidden message.*

In this paper, the existing techniques for image Steganography have some serious drawbacks and this new approach is presented to overcome those drawbacks. It will be done using the concepts of linked list data structure. Initially the audio file is encrypted using Rijndael algorithm. Then first, creating a "stego-key" by the address of message blocks. Second, it makes the detection of audio message harder. Possible countermeasure to prevent detection included hiding encrypted-audio message in perceptually less significant areas of image and scattering that encrypted-audio message throughout the image. The concept of "adaptive steganography", where relatively busy areas of carrier are selected that are less likely to convey the existence of hidden audio.

Keywords: *Steganography, Cryptography, Linked List, Rijndael, Secure Communication, Audio.*

I. INTRODUCTION

The advantage of steganography over cryptography is that messages do not attract attention to themselves. Plainly visible encrypted messages. No matter how unbreakable and will arouse suspicion, and may in themselves be incriminating in countries where encryption is illegal [1]. Therefore, whereas cryptography protects the contents of a message, steganography can be said to protect both messages and communicating parties and its type are:

A. Physical Steganography:

Hidden messages on paper written in secret inks, under other messages or on the blank parts of other messages. Messages written in Morse code on knitting yarn and then knitted into a piece of clothing worn by a courier. Messages written on the back of postage stamps.

B. Digital Steganography:

Modern steganography entered the world in 1985 with the advent of the personal computer applied to classical steganography problems. Development following that was slow, but has since taken off, going by the number of 'stego' programs available: Over 725 digital steganography applications have been identified by the Steganography Analysis and Research Center (SARC).

C. Blog Steganography:

Messages are fractionalized and the (encrypted) pieces are added as comments of orphaned web-logs (or pin boards on social network platforms). In this case the selection of blogs is the symmetric key that sender and recipient are using; the carrier of the hidden message is the whole blogosphere.

D. Printed Steganography:

Digital steganography output may be in the form of printed documents. A message, the plaintext, may be first encrypted by traditional means, producing a ciphertext. Then, an innocuous covertext is modified in some way to as to contain the ciphertext, resulting in the stegotext. For example, the letter size, spacing, typeface, or other characteristics of a covertext can be manipulated to carry the hidden message. Only a recipient who knows the technique used can recover the message and then decrypt it.

Fig. 1 illustrates this matter and the following formula provides a very generic description of the pieces of the following steganographic process [2-4]:

cover_medium + hidden_data + stego_key = stego_medium

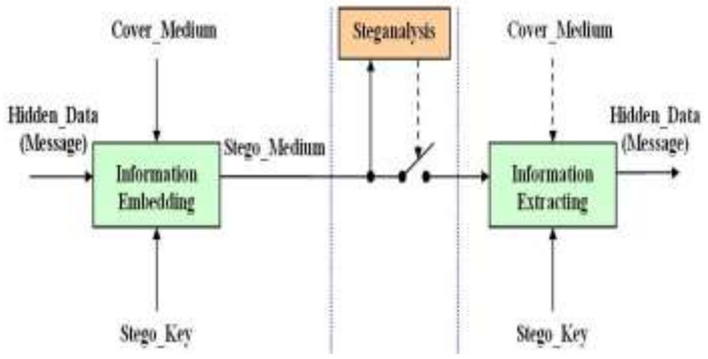


Fig.1 Dataflow for information hiding with an adversary.

Modern steganography attempts to be detectable only if secret information is known-namely, a secret key. This is similar to Kirchhoff's' Principle in cryptography, which holds that a cryptographic system's security should rely solely on the key material. For steganography to remain undetected, the unmodified cover medium must be kept secret, because if it is exposed, a comparison between the cover and stego media immediately reveals the changes [5].

In steganography there are two types of components: message and carrier [6-10]. Message is the secret data which should be hidden (and it is audio in the proposed method); and carrier is the context (image) that hides the message in it sporadically with a structure like linked list, and random locations of its data blocks. By this, two important goals are achieved:

- Make the detection of message harder to gain to stricter security.
- Create a security key for extracting message. Since the head of the message has a random location in the cover image, so the initial address of it can be used as a key [11-15].

There are some drawbacks in the current techniques like:

- Extremely liable to attacks like Image Manipulation techniques where the pixels will be scanned for a possible relation which will be used to trace out the actual characters.
- 24 bit images are to be used at great risk.
- Extreme Care needs to be taken in the selection of the cover image, so that changes to the data will not be visible in the stego-image.
- Commonly known images, such as famous paintings must be avoided.

For embedding the message in a 24-bit RGB image, the LSB (Least Significant Bit) technique was used. So, section II provides a talk about basic concepts about RGB color space and the LSB technique fundamentals. Section III describes the audio files as a payload to be concealed, while section IV designates what linked lists look like and Rijndael algorithm for audio Cryptography is presented in section V. In section VI explains the linked list structured audio message embedding algorithm and the number of pixels needed to store it. Main part of section VII is devoted to experiments. Finally, Conclusion and future work is sited at the end of paper.

II. BASIC CONCEPTS ABOUT RGB COLOR SPACE AND THE LSB TECHNIQUE FUNDAMENTALS

An image can be represented by a collection of color pixels. The individual pixels are represented by their optical characteristics like "brightness", "chroma" etc. Each of these characteristics can be digitally expressed in terms of 1's and 0's [16]. There are different color spaces that present different forms for storing images. A color space is a method by which it is possible to specify, create and visualize color [17]. The most common color space among all is RGB (Red, Green, and Blue). The RGB color model is additive in the sense that the three light beams are added together, and their light spectra add, wavelength for wavelength, to make the final color's spectrum. Each pixel in a 24-bit bitmap image in this space is described by 3 sets of 8 bits (3 bytes), that each set contains the intensity value of individual red, green and blue. Combination of these values forms the characteristics of the pixel. The RGB color model already had a solid theory behind it, based in human perception of colors. Fig. 2 illustrates this matter.

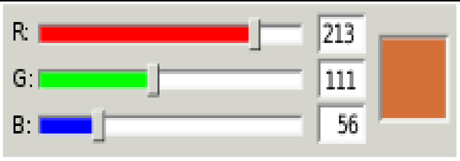


Fig.2 A pixel in RGB color space

One simple method is LSB, or Least Significant Bit Steganography. An image is nothing more than strings and strings of bytes, each byte representing a different color. The last few bits in a color byte, however, do not hold as much significance as the first few. This is to say that two bytes that only differ in the last few bits can represent two colors that are virtually indistinguishable to the human eye. For example, 00100110 and 00100111 can be two different shades of Red, but since it is only the last bit that differs between the two, it is impossible to see the color difference. LSB Steganography, then, alters these last bits by hiding a message within them [18].

To hide a secret message inside an 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. For example, the following grid can be considered as 3 pixels of a 24-bit color image, using 9 bytes of memory:

(00100111 11101001 11001000)
(00100111 11001000 11101001)
(11001000 00100111 11101001)

When the character **A**, which binary value equals 10000001, is inserted, the following grid results:

(0010011**1** 1110100**0** 1100100**0**)
(0010011**0** 1100100**0** 1110100**0**)
(1100100**0** 0010011**1** 11101001)

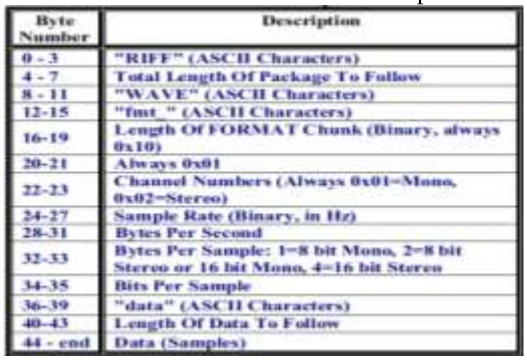
In this case, only three bits needed to be changed to insert the character successfully. On average, only half of the bits (50%) in an image will need to be modified to hide a secret message using the maximal cover size. The result changes that are made to the least significant bits are too small to be recognized by the human visual system (HVS), so the message is effectively hidden [19]. As one can see, the least significant bit of the third color is remained without any changes. It can be used for checking the correctness of 8 bits which are embedded in these 3 pixels. In other words, it could be used as “parity bit”. To hide message into an image in the LSBs of each byte of a 24-bit image, one can store 3 bits in each pixel. A 1024 x 768 image has the potential to hide a total of 2,359,296 bits (294,912 bytes) of information.

III. THE AUDIO (PAYLOAD) TO BE CONCEALED

The presented work studies the possibility of hiding audio messages within digital images. There are some numbers of types of audio files. The most common are Wave files (wav) and MPEG Layer-3 files (MP3). There are, however, many other audio file types. The type is usually determined by the file extension. Different audio file formats have been inspected to find the most suitable format to be used in the concealing process.

Dealing with audio files is the main topic and its format should be emphasized. WAV file format is a subset of Microsoft's RIFF specification for the storage of multimedia files. A RIFF file starts out with a file header followed by a sequence of data "chunks". A WAV file is often just a RIFF file with a single "WAVE" chunk which consists of two sub-chunks - a "fmt" chunk specifying the data format and a "data" chunk containing the actual sample data. The WAV file header contains the following 44 bytes in total only as listed in Table 1 and after that the data chunks starts in bytes.

Table 1: WAV File Format description



Byte Number	Description
0 - 3	"RIFF" (ASCII Characters)
4 - 7	Total Length Of Package To Follow
8 - 11	"WAVE" (ASCII Characters)
12-15	"fmt_" (ASCII Characters)
16-19	Length Of FORMAT Chunk (Binary, always 0x10)
20-21	Always 0x01
22-23	Channel Numbers (Always 0x01=Mono, 0x02=Stereo)
24-27	Sample Rate (Binary, in Hz)
28-31	Bytes Per Second
32-33	Bytes Per Sample: 1=8 bit Mono, 2=8 bit Stereo or 16 bit Mono, 4=16 bit Stereo
34-35	Bits Per Sample
36-39	"data" (ASCII Characters)
40-43	Length Of Data To Follow
44 - end	Data (Samples)

In all international listening tests, MP3 impressively proved its superior performance, maintaining the original sound quality at a data reduction of 1:12 (around 64 Kbit/s per audio channel) via the inner iteration loop (rate loop) and the outer iteration loop (noise control/distortion loop). If applications may tolerate a limited bandwidth of around 10 kHz, a reasonable sound quality for stereo signals can be achieved even at a reduction of 1:24. Fig.3 illustrates this matter.

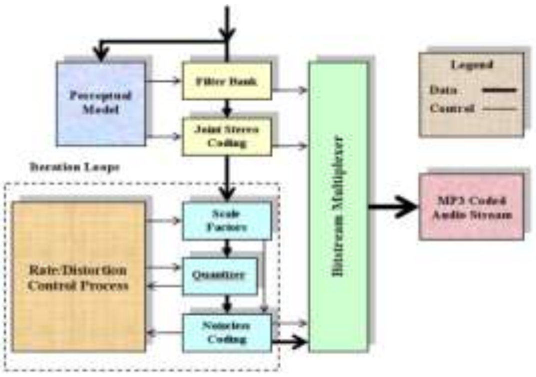


Fig.3 MP3 Encoder Flowchart.

IV. WHAT LINKED LISTS LOOK LIKE

An array allocates memory for all its elements lumped together as one block of memory. In contrast, a linked list allocates space for each element separately in its own block of memory called a "linked list element" or "node". So, linked list is a data structure where in each element contains both a data value and a pointer to next element in the list. The list gets is overall structure by using pointers to connect all its nodes together like the links in a chain. Each node contains two fields: a "data" field to store whatever element type the list holds for its client, and a "next" field which is a pointer used to link one node to the next node [20].

Thinking of linked list as a train which is illustrated in Fig.4. The programmer always stores the first node of the list. This would be the engine of the train. The pointer is the connector between cars of the train. Every time the train adds a car, it uses the connectors to add a new car. This is like a programmer using the keyword new to create a pointer to a new struct or class.

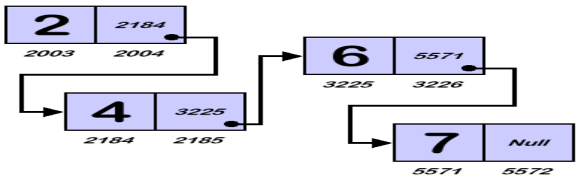


Fig.4 Linked Lists Structured example.

The front of the list is a pointer to the first node. Here is what a list containing the numbers 1, 2, and 3 might be illustrated in Fig. 5. That drawing shows the list built in memory by the function BuildOneTwoThree(). The beginning of the linked list is stored in a "head" pointer which points to the first node. The first node contains a pointer to the second node. The second node contains a pointer to the third node, and so on. The last node in the list has its next field set to NULL to mark the end of the list. Here is the function which uses pointer operations to build the list {1, 2, 3}.

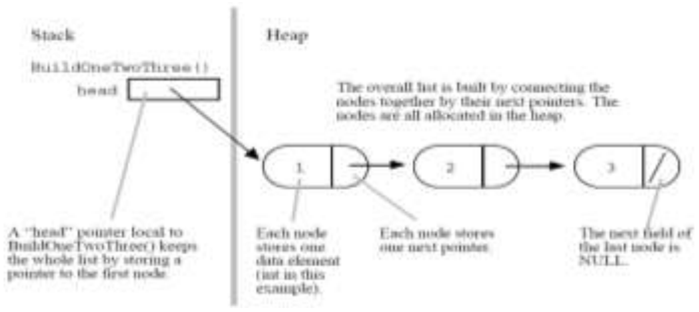

Stack
BuildOneTwoThree()
head
Heap
The overall list is built by connecting the nodes together by their next pointers. The nodes are all allocated in the heap.
1
2
3
A "head" pointer local to BuildOneTwoThree() keeps the whole list by storing a pointer to the first node.
Each node stores one data element (int in this example).
Each node stores one next pointer.
The next field of the last node is NULL.

Fig.5 The drawing of list {1, 2, 3} linked list structured

This function demonstrates how calls to malloc() and pointer assignments (=) work to build a pointer structure in the heap.

```
/* Build the list {1, 2, 3} in the heap and store its head pointer in a local stack variable. Returns the head pointer to the caller. */
struct node* BuildOneTwoThree()
{
  struct node* head = NULL;
  struct node* second = NULL;
  struct node* third = NULL;
  head = malloc(sizeof(struct node)); // allocate 3 nodes in the heap
  second = malloc(sizeof(struct node));
  third = malloc(sizeof(struct node));
  head->data = 1; // setup first node
  head->next = second; // note: pointer assignment rule
  second->data = 2; // setup second node
  second->next = third;
  third->data = 3; // setup third link
  third->next = NULL;
  /* At this point, the linked list referenced by "head" matches the list in the drawing. */
  return head;
  }
```

V. RIJNDAEL ALGORITHM FOR AUDIO CRYPTOGRAPHY

This standard specifies the Rijndael algorithm [21], a symmetric block cipher that can encrypt data blocks of 128 bits, using cipher keys with lengths of 128, 192, and 256 bits. In the description of this section, we assume a key length of 128 bits, which is likely to be the one most commonly implemented. The audio input, the output and the cipher key for Rijndael are each bit sequences containing 128 bits with the constraint that the audio input and output sequences have the same length. The Advanced Encryption Standard (AES) has these advantages: (Very Secure - Reasonable Cost - Main Characteristics: Flexibility, Simplicity). Fig. 6 depicts the structure of a full AES encryption round.

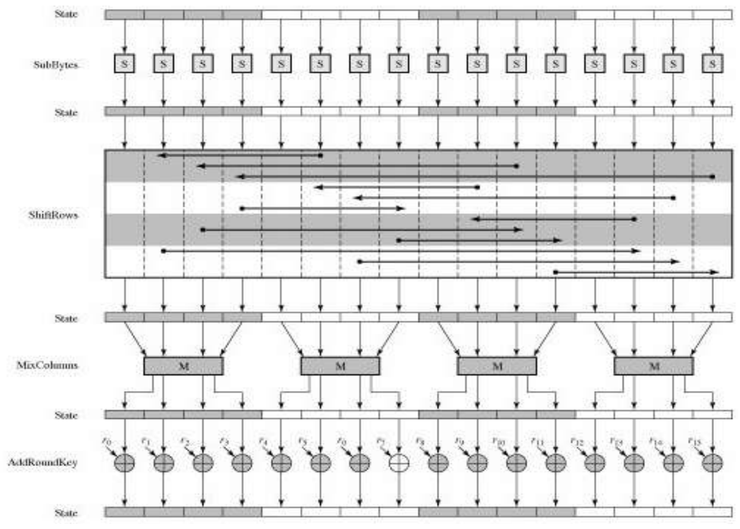


Fig.6 AES Encryption Round

VI. STEGANOGRAPHIC SYSTEM IN USE

In this section the encrypted-audio message will be embedded in cover image with a structure like what linked lists place in the memory as above. Ronak's methodology [22] has been exploited to present this steganographic system in use to deal with audio files. As one knows, linked list is a data structure like an array, but the most important difference between them is in their placement in RAM. Arrays sit in sequential order of memory places, but linked lists get sporadic addresses, and each item (which is called "node") stores proposed data and also the address of the next item in the memory [23]. Using this concept, separate bytes of encrypted-audio message will be embed sporadically in cover image, so that, address of the next byte far apart in the image will be placed just after embedding each byte. By this, two advantages will be achieved: First, non-sequence of message structure makes the detection harder, and it increases the security level. Second, the address of first byte of message could be used as stego-key. As one knows, while working with linked lists, always the address of first node is stored in a pointer for accessing the linked list data. Losing this address means losing the data stored in linked list. So, taking this concept to work in steganography for creating a key for message [24, 25].

Another important point must be haven to consider is about the random location of message bytes. Each image provides a 2 dimension (*X,Y*) space for putting the encrypted-audio message in it. Now, an algorithm to generate the address of non-repeated locations is used. There are some algorithms for it. For example, one can suppose the image as a simple 1 dimension array, and then do the block scheming. Each block can be a set of pixels for storing a byte of message and also some extra pixels to store the next byte address. The number of pixels in a block depends on the size of image. For an image with *x*y* pixels, following equation can be used to determine the number of pixels needed for storing the address:

$$p = \left\lceil \frac{k}{3} \right\rceil : x * y \leq 2^k \qquad \text{…………} \text{ (I)}$$

That *p* is the number of pixels for address. There are (*x*y*) items that should be addressed. So, a need of *k* bits so that $x * y \leq 2^k$. It can be concluded that the number of pixels will be equal to *p* in equation (I). For example, suppose that there is a 20*30 pixels image. For addressing:

$$20 * 30 \leq 2^{10}$$

$$p = \left\lceil \frac{10}{3} \right\rceil = 4\,(pixels)$$

So each block in this image should contain 3 pixels for a byte of message, in addition to 4 pixels for address of the next byte. After block scheming, the random numbers generation algorithms will be used to choose the blocks for storing data in them. Just suppose that there is a function which is named "RandomBlocks()" and it does the block scheming and random block selection. For more information about generating random numbers refer to [26]. Here, the algorithm for embedding the encrypted-audio message in a cover image with linked list structure. It is done by a function which called it "Embed()".

```
Class block
  {
  block(); // Constructor
  void SetData(byte); //Sets the byte for audio data part of current block
  void SetLink(block); //Sets the block for link part of current block
  byte GetData(); // returns the audio data part of current block
  block GetLink(); // returns the Link part of current block
  block GetAddress(); // returns the address of current block
  }
function Embed (encrypted-audio message)
  {
  new=RandomBlock();
  new.SetData(First byte of encrypted-audio message);
  key=GetAddress();
  previous=new;
  for each byte in encrypted-audio message // From second byte
      {
          new=RandomBlock();
          new.SetData(byte);
          previous.SetLink(new.GetAddress);
          previous=new;
      }
  previous.SetLink(NULL);
  }
```

For reading the message a recursive algorithm is presented. It is done by Extract() function. While calling this function for the first time, key should be sent as a parameter of this function.

```
byte Extract (block)
  {
          if (block.GetLink==NULL) return block.GetData;
          else return read(block.GetLink);
  }
```

To get into the characteristics of this algorithm, the following lines can be pointed out:

- A pseudorandom number generator (PRNG) seeded with a value known to both sender and receiver, is used to randomly select the first pixel in the cover image [27].
- Presented algorithm can be used for any kind of message embedding such as text, images and even the files; because all of them can be reached in bytes form.
- Considering equation (I), maximum bytes of the encrypted-audio message that can be embedded in an image with $x*y$ pixels (n_b) is calculated as follow:

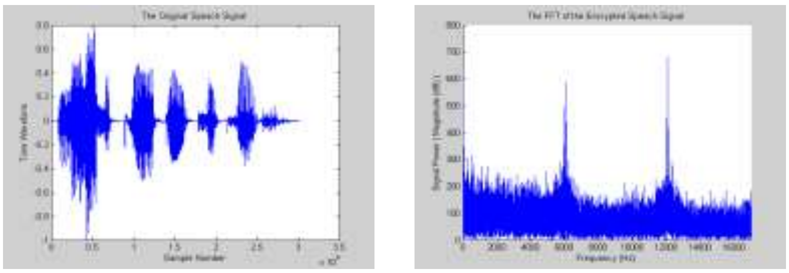
$$n_b = \frac{\sum(pixels)}{p+3} = \frac{x*y}{p+3} \qquad \text{……… (II)}$$

- It is easily possible to add new blocks of data in the chain of message. Also, removing them could be done easily [28].
- More than one encrypted-audio message can be embedded in a cover image. It means we can have more than one chain of message with different keys. This feature could be very useful when the receiver of audio message is more than one, and each of them should receive and retrieve their own audio message.
- This algorithm can be used as a layer of programming in the process of securing data.

VII. IMPLEMENTATION AND RESULTS

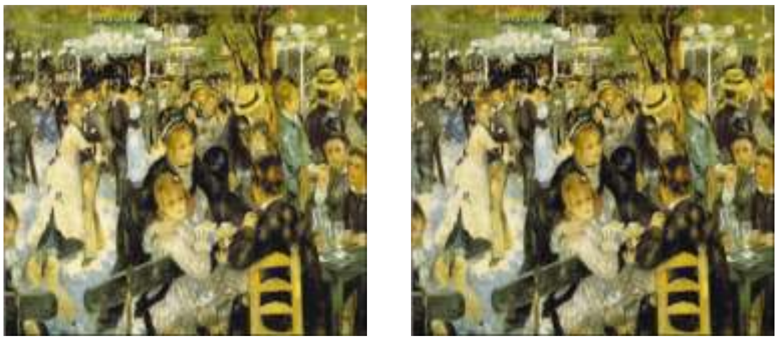
The above proposed work has been implemented in MATLAB 7a . A set of test cover images are investigated before and after hiding the encrypted-audio files using previously mentioned method. Here, replacement of the existing pixels in the image with the new ones in such a way that no difference is visible between the steganographed and the original image. In developing test scenarios it was found that the extracted audio messages when compared to the original audio files were identical with them.

Original audio with size 96 Kbytes and its encrypted (version using the AES) in frequency domains are shown in Fig 7 (a) and (b) respectively. It seems a noisy sound. Hiding the encrypted audio resulted from Fig.7 (b) into 24-bit Renoir 512 x 512 cover image Fig. 8 (a) has the potential to hide a total of 786,432 bits (96 Kbytes) of audio information yielding the stego-image as shown in Fig. 8 (b).



(a) (b)

Fig.7 An example of encryption audio signal using AES.
(a) A sample of speech signal in time domain;
(b) The FFT of the encrypted speech signal in frequency domain (amplitude spectrum).



(a) (b)

Fig.8 Hiding encrypted audio resulted from Fig.7 (b) into 24-bit Renoir cover image
(a) Renoir 512 x 512 Cover-image
(b) Renoir 512 x 512 pixels Stego-image

VIII. CONCLUSIONS AND FUTURE WORKS

In this project, a new robust system for the combination of Cryptography and Steganography have been introduced which could be proven to be a highly secured method for data communication. This system is to provide a good, efficient method for hiding the encrypted-audio in RGB 24-bit color images from hackers and sent it to the destination in a safe manner. AES algorithm has been used to increase the security of the system. This proposed system will not change the size of the image file even after encoding and it also suitable to deal with any type of audio file format. Thus the audio data hiding techniques can be used for a number of purposes other than covert communication or deniable data storage, information tracing and finger printing, tamper detection.

Here the audio message is first encrypted then embedded in image file with help of linked list structure to enhance confidentiality of audio information and provides a means of communicating privately. Based on the results that are recorded from the experiments, the performance of the proposed method is perfect and suitable to be used in a wide range of application. More quick algorithms may be developed and some heuristic approaches may be developed for this purpose.

IX. Acknowledgements

The author wishes to thank Prof. Dr. Shilong Ma in Beijing University of Aeronautics and Astronautics (BUAA), China. This work was supported in part by Prof. Dr. Mohammed Sakre, AL-Shorouk Academy, Egypt.

REFERENCES

[1] Saurabh Singh, Gaurav Agarwal "Use of image to secure text message with the help of LSB replacement", international journal of applied engineering research Volume 1, No.2, 2010.

[2] J. Fridrich and M. Goljan, "Practical Steganalysis-State of the Art," Proc. SPIE Photonics Imaging 2002, Security and Watermarking of Multimedia Contents, vol. 4675, SPIE Press, 2002, pp. 1–13.

[3] H. Farid, "Detecting Hidden Messages Using Higher Order Statistical Models," Proc. Int'l Conf. Image Processing, IEEE Press, 2002.

[4] S. Lyu and H. Farid, "Detecting Hidden Messages Using Higher-Order Statistics and Support Vector Machines," Proc. 5th Int'l Workshop on Information Hiding, Springer Verlag, 2002.

[5] S. Kurshid Jinna1, Dr. L. Ganesan," Lossless Image Watermarking using Lifting Wavelet Transform", International Journal of Recent Trends in Engineering, Volume 2, Number 1, November (2009), pp. (191-195).

[6] T. Zhang and X. Ping, "A Fast and Effective Steganalytic Technique Against JSteg-like Algorithms," Proc. 8th ACM Symp. Applied Computing, ACM Press, 2003.

[7] H.L.V. Trees, Detection, Estimation, and Modulation Theory, Part I: Detection, Estimation, and Linear Modulation Theory, Wiley Interscience, 2001.

[8] U. Grenander and A. Srivastave, "Probability Models for Clutter in Natural Images," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 23, no. 4, 2001.

[9] J. Fridrich, M. Goljan, and D. Hogea, "Attacking the OutGuess," Proc. ACM Workshop Multimedia and Security 2002, ACM Press, 2002.

[10] Neil F. Johnson, Exploring Steganography : Seeing the Unseen, IEEE Computer Feb. 1998, pp. 26-34.

[11] Nick Parlante, Linked List Basics, Copyright © 1998-2001.

[12] Pierre L'Ecuyer: Comparison of Point Sets and Sequences for Quasi-Monte Carlo and for Random Number Generation. SETA, 2008: 1-17.

[13] A. Westfeld, "F5- A Steganographic Algorithm: High Capacity Despite Better Steganalysis," Proc. 4th Int'l Workshop Information Hiding, Springer-Verlag, 2001, pp. 289–302.

[14] J. Fridrich, M. Goljan, and D. Hogea, "Steganalysis of JPEG Images: Breaking the F5 Algorithm," Proc. 5th Int'l Workshop Information Hiding, Springer-Verlag, 2002.

[15] N. Provos and P. Honeyman, "Detecting Steganographic Content on the Internet," Proc. 2002 Network and Distributed System Security Symp., Internet Soc., 2002.

[16] Soumyendu Das, Subhendu Das, Bijoy Bandyopadhyay, Sugata Sanyal, Steganography and Steganalysis: Different Approaches, Steganography Primer - Ruid, Computer Academic underground, 2004.

[17] Guillermito.Steganography: A Few Tools to Discover Hidden Data, 2006.

[18] Hide & Seek: An Introduction to Steganography: Niels Provos and Peter Honeyman, IEEE Security & Privacy Magazine, May 2003.

[19] C. Cachin, An Information-Theoretic Model for Steganography, Cryptology ePrint Archive, Report 2000/028, 2002, www.zurich.ibm.com/˜cca/papers/stego.pdf.

[20] S. Kurshid Jinna1, Dr. L. Ganesan," Lossless Image Watermarking using Lifting Wavelet Transform", International Journal of Recent Trends in Engineering, Volume 2, Number 1, November (2009), pp. (191-195).

[21] D.R. Stinson, Cryptography: Theory and Practice, Boca Raton, CRC Press, 1995. ISBN: 0849385210.

[22] Ronak Karimi, et. al., Embedding Stego-Text in Cover Images, World Applied Programming, Vol (1), No (4), October 2011. 264-268, ISSN: 2222-2510.

[23] Emad S. Othman, Digital Image Steganography Based on EHMC "Data Sneaking Between Pixels" Proceedings of the 1st International Conference on Computer Science from Algorithms to Applications (CSAA), JW Marriot Mirage City, Cairo, EGYPT, December 8-10, 2009.

[24] Emad S. Othman, Compression and Encryption Algorithms for Image Satellite Communication, International Journal of Scientific & Engineering Research, IJSER- France, Volume 3, Issue 9, September 2012.

[25] A. Cheddad, J. Condell, et. al., "Enhancing Steganography in Digital Images", IEEE, (2008), pp. (326-332).

[26] Elis Horowitz, Sartag Sahni, Dinish Mehta: Fundamentals of Data Structures in C++, 2nd ed. Silicon Press, 2006.

[27] www.waprogramming.com , 2011 WAP journal.

[28] http://www.americanscience.org