ISSN XXXX XXXX © 2017 IJESC

Research Article Volume 7 Issue No.4

Image Steganography Based on AES Algorithm with Huffman Coding for Compressionon Grey Images

Jacob Herison Kennedy[1], MD Tabrez Ali Khan[2], MD Junaid Ahmed[3], MD Rasool[4]
B.Tech Students[1,2,3], Associate Professor[4]
Department of Computer Science Engineering
Lords Institute of Engineering and Technology, Himayat Sagar, Telangana, India

Abstract:
A secret message hidden inside an image, the image's Encryption and Decryption using AES (Advance Encryption Standard) algorithm, LSB algorithm and compression and decompression of that image using Huffman Coding are proposed in this paper. The design uses the iterative approach with block size of 128 bit and key size of 256 bit. The numbers of round for key size of 256 bits is 14, as the secret key increases the security as well as complexity of the cryptography algorithms. This paper presents an algorithm, in which the image is an input to AES Encryption to get the encrypted image, then it is compressed with Huffman Coding and the encrypted image is uncompressed and given as the input to AES Decryption to get the original image.

Keywords: AES algorithm, LSB algorithm, image encryption, steganography, cryptography, image decryption, Huffman Coding.

I.INTRODUCTION

Steganography is the art and science of writing hidden messages in such a way that no one, apart from the sender and intended recipient, suspects the existence of the message, a form of security through obscurity. The word steganography is of Greek origin and means "concealed writing" from the Greek words steganos meaning "covered or protected", and graphei meaning "writing".

The use of the internet and wireless communications has been rapidly growing and occupying a wide area in everyday life. Millions of users generate and interchange large amount of electronic data on a daily basis in diverse domains. However, the issue of privacy and security is on the top of the crucial concerns which determine the diffusion of such applications into the daily life. Hence, cryptography turns to become the key for the reliability and effectiveness of the embedded Technologies. Nowadays cryptography has a main role in embedded systems designs. In many applications, the data requires a secured connection which is usually achieved by cryptography. Cryptography is divided in two types first is symmetric key cryptography (sender and receiver shares the same key) and the second one is asymmetric key cryptography (sender and receiver shares different key).

Symmetric systems contain Data Encryption Standard (DES), 3DES, and Advanced Encryption Standard (AES) use an identical key for the sender and receiver. DES (Data Encryption Standard) was considered as the standard of symmetric key encryption, which has a key length of 56 bit. This key length is considered as very small and can be easily broken. The National Institute of Standards and Technology (NIST), proposed Rijndael algorithm as the Advanced Encryption Standard (AES) in October 2000 providing strong security and high flexibility. AES has a fixed block size of 128 bits and a key size of 128, 192 or 256 bits. Huffman Coding was proposed by DR. David A. Huffman in 1952. — A method for the construction of minimum redundancy code. Huffman code is a technique for compressing data. Huffman's greedy algorithm looks at the occurrence of each character and it as a binary string in an optimal way.

Huffman coding is a form of statistical coding which attempts to reduce the amount of bits required to represent a string of symbols. The algorithm accomplishes its goals by allowing symbols to vary in length. Shorter codes are assigned to the most frequently used symbols, and longer codes to the symbols which appear less frequently in the string (that's where the statistical part comes in). Code word lengths are no longer fixed like ASCII .Code word lengths vary and will be shorter for the more frequently used characters

II. PROCEDURE

We perform the combination of both Steganography andCryptography thereby ensuring high degree of security for the data. For the steganography we employ LSB substitution technique and then carrying out encryption using AES technique which involves 128 bit block size of data and 128 bit block size of the key.

The process of hiding text into the image is done using the spatial domain approach named as LSB substitution technique, wherein LSB of the cover image is being replaced by the data. This method ensures larger data holding capacity with negligible compromise on the image quality.

This is because in an image there is high degree of redundant information as adjacent pixels are highly correlated and HVS cannot distinguish among correlated pixels.

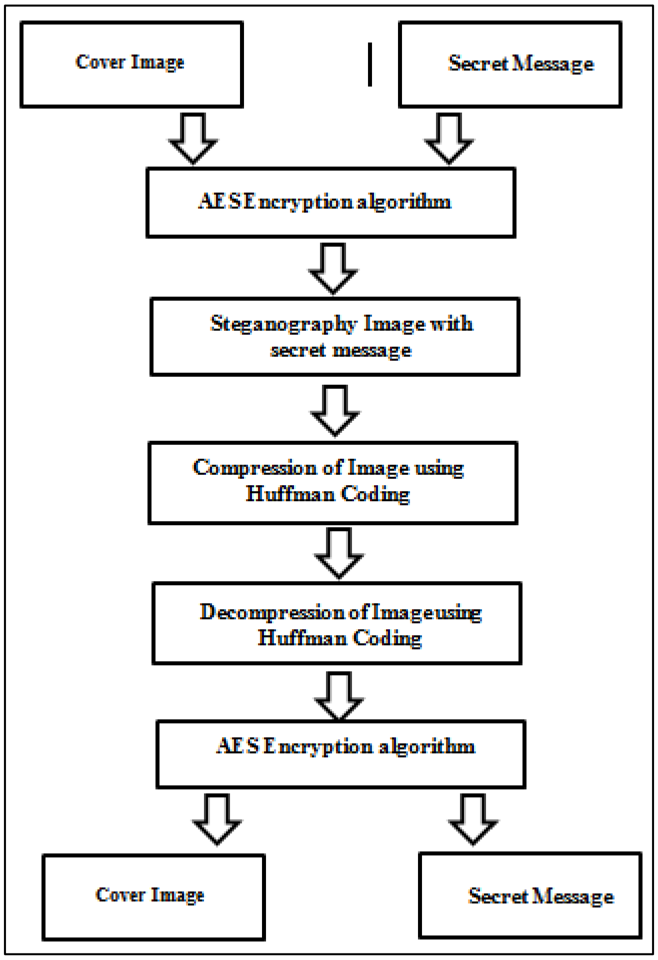

Cover Image
Secret Message
AES Encryption algorithm
Steganography Image with secret message
Compression of Image using Huffman Coding
Decompression of Image using Huffman Coding
AES Encryption algorithm
Cover Image
Secret Message

Figure.1. Image steganography based on AES algorithm with Huffman coding for compression

A.Steganography Implementation Using LSB Substitution

The complete algorithm of data hiding in an image is given in as follows:
Let C be the original 8-bit gray scale cover-image of
Mc× Nc pixels represented as
C = {Xij | 0≤ i <Mc, 0≤ j <Nc

Xij Ɛ {0, 1… 255} ………………….. (1)

M be the n-bit secret message represented as

M = {mi | 0≤ i <N, mi Ɛ {0, 1}} …….….. (2)

For embedding the n-bit secret message M into the k rightmost LSBs of the cover-image C, the secret message
M is rearranged to form a conceptually k-bit virtual image M' which is represented as,

M' = {m'ij |0≤ i <n', m'i Ɛ {0, 1… 2k− 1}}. (3)

Where, n'< Mc×Nc.
The mapping between the n-bit secret message M = {mi} and the embedded message M'= {m'i}
Can be defined as follows:

$m'i = \sum_{i=0}^{k-1} m\, i * k + j * 2k - 1 - j$ ………………………….……(4)

A subset of n' pixels {xl1, xl2,………., xln} is chosen from the cover-image C in a predefined sequence. The embedding process is completed by replacing the k LSBs of xli by m'i. Mathematically, the pixel value xli of the chosen pixel for storing the k-bit message m'i is modified to form the stego-pixel x'li as follows:

x'li = xli- xli mod 2k + m'I.……………………………………. (5)

Also, Algorithm for LSB Based extracting process is given as:
In the extraction process, given the stego-image S, the embedded messages can be directly extracted. Using the same sequence as in the embedding process, the set of pixels {xl1, xl2… xln} storing the secret message bits are selected from the stego-image. The k LSBs of the selected pixels are extracted and lined up to reconstruct the secret message bits.

B. Implementation of Encryption using AES method

1) Designing Steps

State array:
The input to the encryption algorithm is a single 128 bit block; now this block is required to be copied into a state array. State Array is a square matrix of bytes. This state array is modified at each stage of encryption.

Key Expansion:
This stage is the most important stage for both encryption as well as decryption. The AES key expansion algorithm takes as input a 4-word key and produces a linear array of 44 words. Each round uses 4 of these words as shown in Fig. 2. Each word contains 32 bytes which means each sub key is 128 bits long.

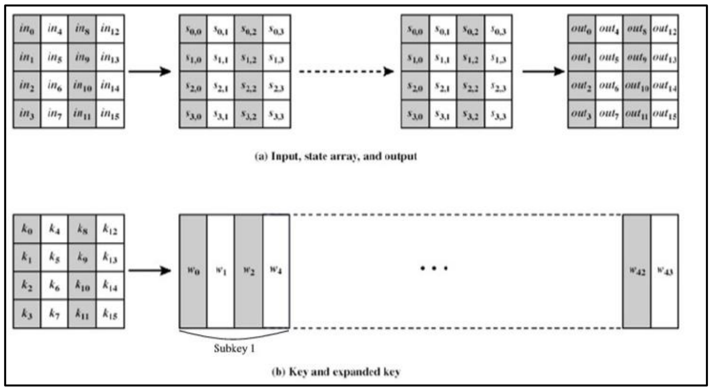


Figure.2. Key Expansion

Add Round Key Expansion:
The 128 bits of State array are bitwise XORed with the 128bits of the round key(4 words of the expanded key).The operation is viewed as a column wise operation between the 4 bytes of the State array column and one word of the round key (Fig. 3).

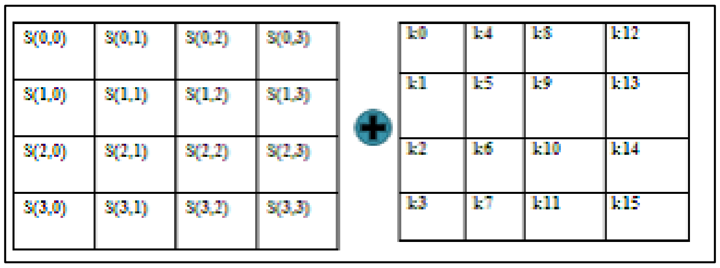
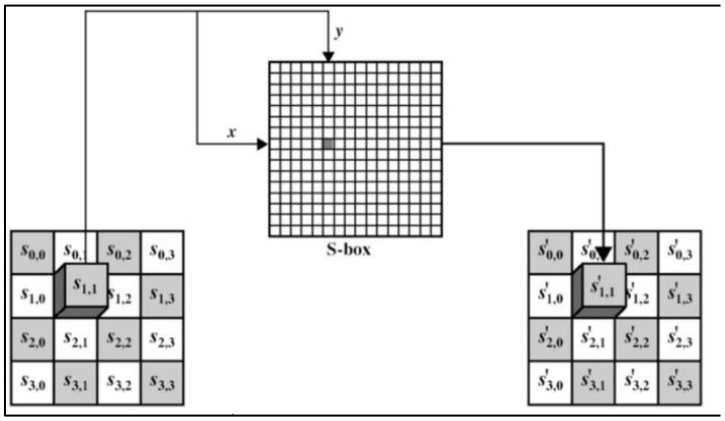

S(0,0) S(0,1) S(0,2) S(0,3)
S(1,0) S(1,1) S(1,2) S(1,3)
S(2,0) S(2,1) S(2,2) S(2,3)
S(3,0) S(3,1) S(3,2) S(3,3)
k0 k4 k8 k12
k1 k5 k9 k13
k2 k6 k10 k14
k3 k7 k11 k15

Figure.3. Add Round Key Expansion

S-Box Substitution:
AES defines a 16 x 16 matrix of byte values, called an S-box which contains a permutation of all possible 256 8- bit values. Each byte of State array is mapped into a new byte in the following way: The leftmost 4 bits of the byte are used as a row designated value and the leftmost 4 bits are used as a column designated value. These row and column designated values serve as indexes into the S-box to select a unique 8-bit output value.


y
x
S-box
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$
Figure.4. S-Box Substitution

Row Shifting:
For the second row, a 1-byte circular left shift is performed. For the third row, a 2- byte circular left shift is performed. For the third row, a 3- byte circular left shift is performed. For the first row, no shifting is performed.

Column Mixing:
It operates on each column individually. Each byte of a column is mapped into a new value that is a function of all four bytes in the column. The transformation can be defined by following matrix multiplication on State array.

C. ALORITHM AND WORKING OF HUFFMAN CODING

We have created a code in Delphi for Huffman method. Following steps describes working and algorithm for Huffman coding.

1. Read a BMP image using image box control in Delphi language. The TImage control can be used to display a graphical image - Icon (ICO), Bitmap (BMP), Metafile (WMF), GIF, JPEG, etc. This control will read an image and convert them in a text file.

2. Call a function that will Sort or prioritize characters based on frequency count of each characters in file.

3. Call a function that will create an initial heap. Then reheap that tree according to occurrence of each node in the tree, lower the occurrence earlier it is attached in heap. Create a new node where the left child is the lowest in the sorted list and the right is the second lowest in the sorted list.

4.Build Huffman code tree based on prioritized list. Chopoff those two elements in the sorted list as they are now part of one node and add the probabilities. The result is the probability for the new node. 5. Perform insertion sort on the list with the new node.

6. Repeat STEPS 3, 4, 5 UNTIL you only have 1 node left.

7. Perform a traversal of tree to generate code table. This will determine code for each element of tree in the following way. The code for each symbol may be obtained by tracing a path to the symbol from the root of the tree. A 1 is assigned for a branch in one direction and a 0 is assigned for a branch in the other direction. For example a symbol which is reached by branching right twice, then left once may be represented by the pattern '110'. The figure below depicts codes for nodes of a sample tree.

```
            *
           / \
        (0)  (1)
        /      \
     (10)    (11)
      /         \
   (110)      (111)
```

8. Once a Huffman tree is built, Canonical Huffman codes, which require less information to rebuild, may be generated by the following steps:

Step-1: Remember the lengths of the codes resulting from a Huffman tree generated per above.
Step-2: Sort the symbols to be encoded by the lengths of their codes (use symbol value to break ties).
Step-3: Initialize the current code to all zeros and assign code values to symbols from longest to shortest code as follows:

A. If the current code length is greater than the length of the code for the current symbol, right shift off the extra bits.
B. Assign the code to the current symbol.
C. Increment the code value.
D. Get the symbol with the next longest code.
E. Repeat from A until all symbols are assigned codes.

9. Encoding Data- Once a Huffman code has been generated, data may be encoded simply by replacing each symbol with its code.

10. The original image is reconstructed i.e. decompression is done by using Huffman Decoding.

11. Generate a tree equivalent to the encoding tree. If you know the Huffman code for some encoded data, decoding may be accomplished by reading the encoded data one bit at a time. Once the bits read match a code for symbol, write out the symbol and start collecting bits again.

12. Read input character wise and left to the tree until last element is reached in the tree.

13. Output the character encodes in the leaf and returns to the root, and continues the step 12 until all the codes of corresponding symbols is known.

III. EXPERIMENTAL RESULTS

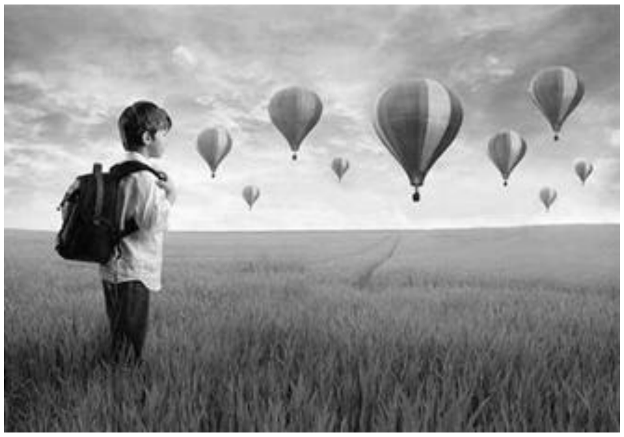


Figure.5. Original gray scale image

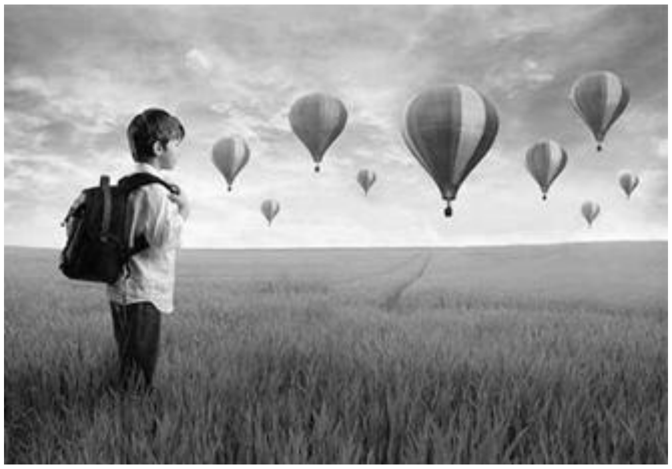


Figure.6. Reconstructed gray scale image

IV. CONCLUSIONS

Huffman coding suffers from the fact that the uncompresser need have some knowledge of the probabilities of the symbols in the compressed files this can need more bit to encode the file if this information is unavailable compressing the file requires two passes first pass: find the frequency of each symbol and construct the Huffman tree second pass: compress the file. This image compression method is well suited for gray scale (black and white) bit map images .This method can be improved using adaptive Huffman coding technique that is an extension to Huffman coding. With this approach we can encode and decode the data without any loss of the information.

V. REFRENCES

[1]. Ahsan, K. & Kundur, D., “Practical Data hiding in TCP/IP”, Proceedings of the Workshop on Multimedia Security at ACM Multimedia, 2002 [1]

[2]. Albiol, L. Torres, and E. J. Delp. “Optimum colour spaces for skin detection.” In proceedings of the 2001international conference on image processing, volume 1,vol. 1, pp. 122-124 , 2001.

[3]. Anderson, R.J. & Petitcolas, F.A.P., “On the limits of steganography”, IEEE Journal of selected Areas in Communications, May 1998

[4]. Artz, D., “Digital Steganography: Hiding Data within Data”, IEEE Internet Computing Journal, June 2001

[5]. Avcibas, I. Memon, N. and Sankur, B.: Image Steganalysis with Binary Similarity Measures. Proceedings of the international conference on Image Processing, 3: 645-648. 24-28 June 2002.

[6]. Bender, W., Gruhl, D., Morimoto, N. & Lu, A., “Techniques for data hiding”, IBM Systems Journal, Vol 35, 1996

[7]. Castleman, K.R., Digital Image Processing. Second ed. 1996, Englewood Cliffs, New Jersey: Prentice-Hall.

[8]. Chandramouli, R., Kharrazi, M. & Memon, N., “Image steganography and steganalysis: Concepts and Practice”, Proceedings of the 2nd International Workshop on Digital Watermarking, October 2003

[9]. Chiu-Yi Chen; Yu-Ting Pai; Shanq-Jang Ruan, Low Power Huffman Coding for HighPerformance Data Transmission, International Conference on Hybrid Information Technology, 2006, 1(9-11), 2006 pp.71 – 77.

[10]. Lakhani, G, Modified JPEG Huffman coding, IEEE Transactions Image Processing, 12(2),2003 pp. 159 – 169. [11] Rudy Rucker, "Mind Tools", Houghton Mifflin Company,1987

[12]. J. Ziv and A. Lempel, ``A Universal Algorithm for Sequential Data Compression," IEEE Transactions on Information Theory, Vol. 23, pp. 337--342, 1977.

[13]. T. A. Welch, ``A Technique for High-Performance Data Compression," Computer, pp. 8--18, 1984

[14]. R.G. Gallager — Variation on a theme by Huffman — IEEE. Trans. on Information Theory,IT-24(6), 1978, pp. 668-674.

[15]. DAVID A. HUFFMAN, Sept. 1991, profileBackground story: Scientific American, pp. 54-58.

[16]. A.B.Watson, ―Image Compression using the DCT‖, Mathematica Journal, 1995,pp.81-88.