2021 2nd International Conference on Secure Cyber Computing and Communications (ICSCCC) | 978-1-6654-4415-6/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICSCCC51823.2021.9478162

LSB Steganography Using Pixel Locator Sequence with AES

Krishnakant Tiwari
Department of Information Technology
ABV - IIITM, Gwalior
kkant5401@gmail.com

Sahil J. Gangurde
Department of Information Technology
ABV - IIITM, Gwalior
sahilgangurde08@gmail.com

***Abstract*—Image steganography is a technique of hiding confidential data in the images. We do this by incorporating the LSB(Least Significant Bit) of the image pixels. LSB steganography has been there for a while, and much progress has been made in it. In this paper, we try to increase the security of the LSB steganography process by incorporating a random data distribution method which we call pixel locator sequence (PLS). This method scatters the data to be infused into the image by randomly picking up the pixels and changing their LSB value accordingly. This random distribution makes it difficult for unknowns to look for the data. This PLS file is also encrypted using AES and is key for the data encryption/decryption process between the two parties. This technique is not very space-efficient and involves sending meta-data (PLS), but that trade-off was necessary for the additional security. We evaluated the proposed approach using two criteria: change in image dynamics and robustness against steganalysis attacks. To assess change in image dynamics, we measured the MSE and PSNR values. To find the robustness of the proposed method, we used the tool *StegExpose* which uses the stego image produced from the proposed algorithm and analyzes them using the major steganalysis attacks such as Primary Sets, Chi-Square, Sample Pairs, and RS Analysis. Finally, we show that this method has good security metrics for best known LSB steganography detection tools and techniques.**

***Index Terms*—Steganography, AES, Image Processing, Information Hiding, Cryptography**

I. Introduction

Steganography is a way of concealing a file, message, image, audio, or video into another file of the same or different category in a manner such that the hidden data cannot be easily recognized by anybody other than the sender and receiver. While cryptography deals with the security of the message, steganography deals with the method of hiding the data in a way that doesn't change the original file data at an enormous cost but yet implements the confidential data into it. Imperceptibility is an essential feature in steganography, also called transparency or anti-detection performance. The imperceptibility of steganography can be improved by enhancing the method of steganography or improving the matching relationship between secret information and carrier. Steganography complements the deficiency of concealment of the encryption. Though there is no genuine relationship between steganography and cryptography, this paper tries to bridge the gap between the two by encrypting the secret data and providing an alternative to a well-known steganographic method.

LSB-based steganography has been there for a while in which the secret data is encrypted in the LSB of a pixel in sequential order [2]. The problem of sequential ordering of this confidential data is that once the user is handed over a stego image, he/she may be able to find the personal information if not encrypted with a well-known encryption algorithm. This paper tries to overcome the traditional way of implementing LSB based image steganography technique with a unique sequence that locates the pixel where the information is hidden in sequential order. The advantage of using this technique over others is that the attacker will now no longer be able to access the data as the location of the data sequentially will be scattered around in random pixel location along with AES encryption.

II. LSB substitution technique

LSB substitution is a powerful technique where confidential text data bits are substituted with the LSB of the original image. This technique is popular because the human eye can't easily find the difference between the actual image and the stego image if the information change is done at least significantly. This technique can also be extended to 2,4 or up to 8 bits, but this may cause distortion and noise in the image resulting in a lossy image.

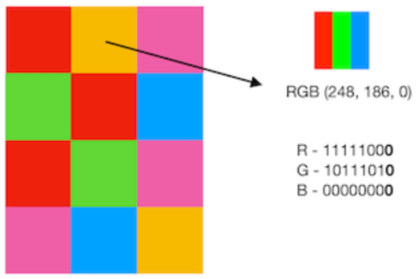

Fig. 1. RGB (3x8-bit pixel) mode

To understand better, let's consider an image as a 2D matrix of pixels. Each pixel contains some value depending upon the type and depth. Let's consider the RGB color mode. These values range from 0-255 (8-bit values). We will convert the hidden text into its corresponding binary format. Now traverse along the pixels and replace the LSB of that pixel according to the confidential text's binary representation. To decode, store the LSB of changed pixels and make a binary representation array out of it. Split it into the group of 8 bits and convert it into ASCII character.

III. AES

For encryption purposes, we have used AES encryption. AES is one of the most used symmetric algorithms in the world [4]. It's a 128-bit block cipher with three main vital sizes 128,192,256 bits. 128-bit key sizes are not safe, and that's why for optimum security, we have used 256-bit keys. AES works on bytes rather than bits. As AES has a block size of 16 bytes, these bytes are usually stored inside a 4*4 matrix.

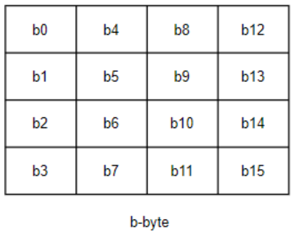

Fig. 2. Byte stored in Matrix

Each round in AES consist of 4 layers

1) SubBytes : SubBytes Layer consists of 16 S-box that are identical. Each S-box takes 1 byte of data and transform byte by taking its inverse in the finite field and then convert it using affine transform.
2) ShiftRows : Each byte is stored in a 4*4 matrix and transformed systematically. In the first row, no shifting is done, and from there onwards ith row is left-shifted by (i-1) column.

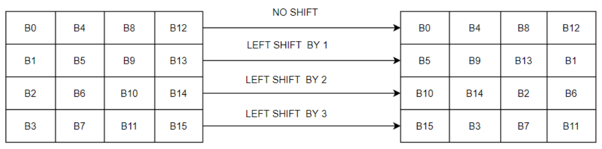

Fig. 3. Shift row operation in 4*4 matrix

3) MixColumns : In this layer, first, the columns are stacked in a column matrix and then multiplied with a unique 4*4 matrix. This matrix multiplication for each column results in a new 4*4 matrix sent to the next round. All the operations are performed in the finite field. This helps us to distribute the changes in the whole bit pattern.
4) AddRoundKey: The subkey which is generated for this round from key-scheduling is added to the output from the previous layer using the XOR operation.

A. *Encyption and Decryption using AES*

In standard implementation of AES, 14 rounds of the above operation are performed for 256-bit key size. As for each round, it requires a unique 128-bit ke. AES has a key schedule

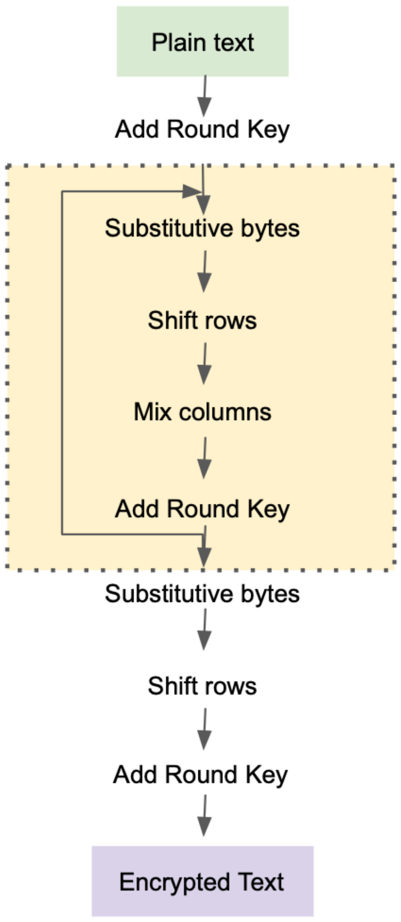

Fig. 4. Encryption in AES

to expand a short key into various separate round keys. In the last round, the mix column layer is not present.

In the decryption process, each step is reversed by taking the inverse of all the transformations in the reverse order.

Block cipher has two key property which makes them immune to an attacker i.e. confusion and diffusion. More formally confusion here means that the final output and input should have no connection between them and Diffusion means small changes in the input must have a huge effect on the final output. In AES confusion is added by the SubBytes layer and the Diffusion is added by ShiftRows and MixColumns layer.

IV. Related Work

LSB-based image steganography is one of the most basic data hiding techniques [1]. Some papers of LSB also deal with checking the authenticity and integrity of data [8]. Most of the research work done in LSB-based steganography primarily focuses on encrypting the secret text using a vengeance cipher like AES before hiding it into the LSB cover image in a linear fashion [2]. Hiding the data in LSB in a linear fashion makes the encrypted message bit more predictable and easy to retrieve. This paper tries to overcome these issues making the method more robust and effective against potential attacks.

V. Proposed Work

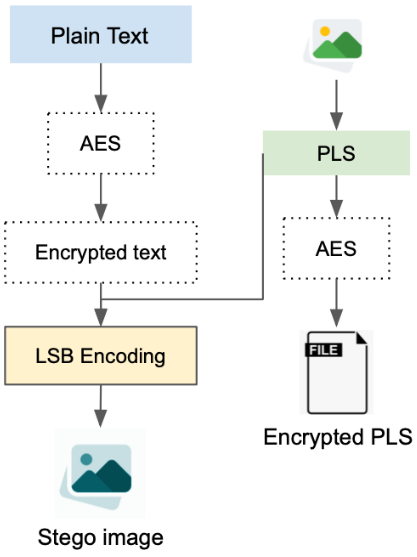

Fig. 5. Encryption Process

To enhance the security of LSB-based steganography, what we propose is to use PLS during the encoding and decoding process of LSB so that text is hidden in a randomly distributed sequence of a pixel inside the image. Pixel Locator Sequence (PLS) is a randomly generated pixel sequence generated uniquely for a particular image.

Till now, LSB-based steganography is done by iterating over the pixel in systematic order. PLS will now allow us to encode the data in random order. For additional security, we will encrypt the text before encoding it into the pixel. To make the decoding possible, we will have to send this sequence along with the stego image. So, we must encrypt the PLS using AES before sending it over to the receiver. See Fig.(5)

The receiver will receive a stego image along with an encrypted PLS file. At first, the PLS is decrypted using AES, and then our algorithm iterate over the PLS and from each pixel decodes the text hidden in the LSB of that pixel. The text we get from this decoding is an encrypted text that further needs to be decrypted to get the original text. See Fig.(6)

VI. Pixel Locator Sequence

Pixel Locator Sequence (PLS) is a randomly generated pixel sequence generated uniquely for a particular image. The sender could manually create PLS. PLS will act as a key during the decoding process. Without the PLS decoding process is impossible. With the help of PLS, we will be able to add randomness to our encoding process. Randomness increases the efficacy of the system and gives additional security to data. The procedure for generating PLS is

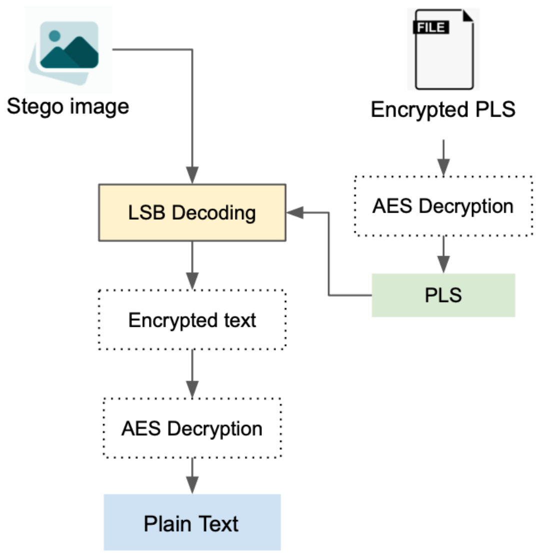

Fig. 6. Decryption Process

discussed below:

Let **N** be the total pixel count in the image and let $\mathbf{N_{enc}}$ be the length of encoded message. The number of pixels $\mathbf{N_p}$ required to encode the given message can be simple calculated using the given formula:

$$N_p = 3 \times N_{enc} \quad (1)$$

Each of the 3 pixels grouped will contain the information of one character present in the encrypted text. The random distribution of pixels in PLS can be done using the **Modern Fisher-Yates Shuffle**. The following algorithm can generate the required number of pixels to store the data in them.

Algorithm 1: Generating PLS sequence

```
Result: Distinct $N_p$ numbers generated in range 0 to
        N-1 stored in PLS array
n = N, m = $N_p$, i = 0;
while $i \leq n\text{-}1$ do
    arr[i] = i;
    i++;
end
i = 0;
while $i \leq m\text{-}1$ do
    swap(arr[n-i], arr[rand()%(n-i+1)]);
    i++;
end
PLS [], i=n-i;
while $i \geq n\text{-}m$ do
    PLS[n-1-i] = arr[i];
end
```

This PLS sequence will then be used to store the data into the image at PLS[i] pixel number, discussed in the LSB encoding algorithm. After the encoding process, this sequence file will be transferred to an AES encrypter and transmitted along with the stego image for the decoding process. For the manual generation of PLS, users can put any random but distinct N_p values from range 0 to N-1.

VII. LSB Encoding and Decoding

The PLS created will be used to distribute the data through pixels across the image. Before iterating over the image pixels, we need the information of the pixel in the i, j format because we consider an image to be made up of $[N] \times [M]$ matrix where N and M are whole numbers. This is easily achievable if we know N and M. Here N and M represent the pixel count in the vertical direction and horizontal directions, respectively. Let X be the pixel's location from the PLS where the data has to be stored. The row and column value of X can be easily calculated using

$$row = X/M, column = X\%M$$

Here % stands for the modular operation, which gives the remainder when X is divided by M.

A. Encoding

Encrypted data is converted into binary code using ASCII. A triad of pixels is created, giving a total of nine RGB values. The RGB values of the pixel are made odd or even according to the parity of the data bits.

Let **textlist** be a list of binary string of the encrypted data and **image** be the image matrix. Also, let the number of characters in the text be $\mathbf{T_n}$. For simplicity, let's assume that for every element in the **image** matrix, the RGB values are easily accessible. For complete implementation refer *(Algorithm 2)*. To implement this system, we used Python along with the Pillow library for image manipulations.

B. Decoding

To decode, read every 3 pixels from PLS. These 3 pixels will produce nine different RGB values. Store these RGB values of these 3 pixels in an array. We will add '1' to the binary string if the value is odd and add '0' if the value is even. Convert this binary string into its corresponding character and append it to the final encrypted string. To decode the actual message, use AES to decrypt the encrypted string.

Let the **stegoimage** be the stego image matrix and **data** be an empty string where the encrypted text will be stored. For complete implementation refer *(Algorithm 3)*.Implementation details can be found on GitHub [9].

Algorithm 2: LSB encoding

```
Result: Text encoded into the image
image[][], PLS[], i = 0, textlist[];
while $i \leq T_n-1$ do
    while $k \leq$ textlist[i].size() do
        text[] = textlist[i];
        pixel [] = Append RGB values of next three
          pixels from PLS;
        while $j \leq 7$ do
            if text[j]=='0' and pixel[j]%2!=0 then
                pixel[j]−−, j++;
            else
                if text[j]=='1' and !(pixel[j]&1)
                if pixel[j]==0 then
                    pixel[j]++, j++;
                else
                    pixel[j]−−, j++;
                end
            end
        end
        k++;
    end
    i++;
end
```

Algorithm 3: LSB decoding

```
Result: Extracting the encrypted text from image
image[][], PLS[], i = 0, data = ”;
while true do
    pixel [] = Append RGB values of next three pixels
      from PLS;
    b_string = ”;
    while $j \leq 7$ do
        if pixel[j]%2==0 then
            b_string += '0';
        else
            b_string += '1';
        end
        j++;
    end
    data += char(ASCII(b_string));
end
```

VIII. Results

The results were calculated based on two factors. Image dynamics addressed the consequences related to the change in image quality and other image-related properties before and after the steganographic procedure performed on them. Steganalysis detection provides results regarding whether the proposed work is secure enough that the standard online stego detection tools do not detect stego traces in the image and mark them non-stego with a good confidence level.

A. Image Dynamics

We compared five images and found the Peak Signal to Noise Ratio(PSNR) and Mean Square Error(MSE) values of them (Table 1). In our experimentation, we tried to encode the message **'This secret message has to be embedded into the image'**. The corresponding encrypted text *74890293687ebfa135e17563dc222c17696db16f6378f1a4e41e*

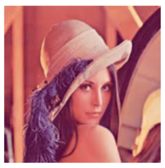
Image1

Image2

Image3

Image4

Image5

Fig. 7. Sample Images

TABLE I. MSE and PSNR values of stego images

Images	Resolution	MSE	PSNR
Image1	225×225	1.68467	45.86564
Image2	493×356	2.27218	44.56635
Image3	512×512	0.95186	48.34506
Image4	512×384	1.19884	47.34317
Image5	512×480	2.24326	44.62198

2ae71929b41199b441f70a75b212914c0f9a64b3f3f18aec3fda5 d379238 was encoded into the image. Below in *(Table 1)* we found the MSE and PSNR values of the given images. The histogram table (Fig. 8) compares the stego and non-stego variations of the same image. The left side of the histogram table contains information about non-stego images and the right side contains information about stego images. The PSNR values are considered to be decent if they are above the threshold limit of 40. All the images showed a greater than 40 value for PSNR. After the AES encryption, the size of the encrypted text is significant. This results in changing a lot of pixels in the image inserted. These values are high, but this trade-off was necessary for security and randomness.

B. Steganalysis Detection

We tested the images mentioned earlier against the various well-known steganalysis tools and techniques using a tool called *StegExpose* [7] built especially for LSB steganography detection. It has built-in algorithms for Primary Sets, Chi-Square, Sample pairs [5] and RS Analysis [6]. The Chi-Square method gives a value between $\{0,1\}$ where a near to zero value indicates the image contains no information. In contrast, a close to one value indicates that the image contains some data. Sample Pairs has a limitation where it tends to fail when the message bits are scattered around and exploits the detection system. Table (2) clarifies the methods discussed above on the stego images produced with the proposed method.

All the images gave a false value for the steganographic detection, and hence the proposed method qualifies for the robustness and efficiency against stego detection attacks.

TABLE II. Steganalysis results

Images	Primary Sets	Chi Square	Sample Pairs	RS Analysis
Image1	0.03653121	null	null	0.01045675
Image2	0.01289309	0.02026724	0.01344952	0.02654193
Image3	0.02220333	0.29121700	0.00224801	0.01922108
Image4	$8.259\text{x}10^{-4}$	0.00509353	0.00269046	0.00480165
Image5	0.01755492	0.02162384	0.01945972	0.02754041

IX. Possible attacks and Counter Measures

If the dimension of the image is small enough such that the brute force algorithm can find all the possible pixel combinations of varied sizes, then the encrypted message can be exposed. But even this brute force attack is of no use since the text itself is also encrypted, and thus you will require a key to decrypt it, which will supposedly be safe. Other than that, the results already show strong opposition to well-known detection systems.

X. Conclusion

The method which we proposed above overcomes the security and limitation of the existing LSB steganographic technique. In this method, the encoding and decoding in the image are done using a randomly generated pixel sequence, i.e., PLS, which helps us distribute data bits all over the image pixel pseudo-random order, making it impossible to decode without knowing the actual pixel sequence. We have also discussed PLS generation techniques using the Modern Fisher-Yates shuffling algorithm. For additional security, we have encrypted the secret text before encoding. Encoding after encryption will undoubtedly increase the data bits of hidden text in some cases, but experimental results show acceptable PSNR and MSE. As we have to send meta-data, i.e., PLS, along with the stego image, this method is not very space-efficient, but we have to make that trade-off for additional security.

We also discussed the various steganography detection techniques when applied on stego images produced by our method, and it gives a false value for stego detection. Also, the scores for the various techniques are significantly less; hence the confidence for such a report is high enough to be verified.

References

[1] F. A. Rafrastara, R. Prahasiwi, D. R. Ignatius Moses Setiadi, E. H. Rachmawanto and C. A. Sari, "Image Steganography using Inverted LSB based on 2nd, 3rd and 4th LSB pattern," 2019 International Conference on Information and Communications Technology (ICOIACT), Yogyakarta, Indonesia, 2019, pp. 179-184, doi: 10.1109/ICOIACT46704.2019.8938503.

[2] A. Pabbi, R. Malhotra and K. Manikandan, "Implementation of Least Significant Bit Image Steganography with Advanced Encryption Standard," 2021 International Conference on Emerging Smart Computing and Informatics (ESCI), Pune, India, 2021, pp. 363-366, doi: 10.1109/ESCI50559.2021.9396884.

[3] O. Elharrouss, N. Almaadeed and S. Al-Maadeed, "An image steganography approach based on k-least significant bits (k-LSB)," 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT), Doha, Qatar, 2020, pp. 131-135, doi: 10.1109/ICIoT48696.2020.9089566.

[4] C. Sanchez-Avila and R. Sanchez-Reillol, ”The Rijndael block cipher (AES proposal) : a comparison with DES,” Proceedings IEEE 35th Annual 2001 International Carnahan Conference on Security Technology (Cat. No.01CH37186), London, UK, 2001, pp. 229-234, doi: 10.1109/CCST.2001.962837.

[5] S. Dumitrescu, Xiaolin Wu and Zhe Wang, ”Detection of LSB steganography via sample pair analysis,” in IEEE Transactions on Signal Processing, vol. 51, no. 7, pp. 1995-2007, July 2003, doi: 10.1109/TSP.2003.812753.

[6] Marçal, André & Pereira, Patricia. (2005). A Steganographic Method for Digital Images Robust to RS Steganalysis. 3656. 1192-1199. 10.1007/11559573_144.

[7] Boehm, Benedikt. (2014). StegExpose - A Tool for Detecting LSB Steganography.

[8] N. Bansal, V. K. Deolia, A. Bansal and P. Pathak, ”Digital Image Watermarking Using Least Significant Bit Technique in Different Bit Positions,” 2014 International Conference on Computational Intelligence and Communication Networks, Bhopal, India, 2014, pp. 813-818, doi: 10.1109/CICN.2014.174.

[9] Github Link: https://github.com/lostmartian/LSB-Steganography-using-Pixel-Locator-Sequence-With-AES.