Volume 9, Issue 5
October 2020

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X

An Image Encryption Algorithm Based on Inter-Pixel Movement of RGB Values and Value Transformation

[1]Amnesh Goel and [1]Dr. Rakesh Kumar Bhujade

[1]Department of Computer Science and Engineering, Mandsaur University, Mandsaur, MP 458001, India

Email: [1]amneshgoel7@gmail.com, [2]rakesh.bhujade@gmail.com

ABSTRACT

In this paper, a new image encryption algorithm is presented which is a combination of RGB inter-pixel movement and the value transformation. Value transformation technique is used in image steganography where the original content is mixed with a temporary message to hide the information. This is a well effective solution, but this increases the size of overall communication because in this case we must pass the message and the temporary message together. In image steganography when a pixel loses its pixel value then it also has a positive impact on the histogram. In this paper, histogram analysis is presented between the plain image, encrypted image and the image which was obtained using value transformation. This is a lossless image encryption technique and core work of this algorithm is based on the RGB inter-pixel displacement. Histogram analysis confirms that the pixels have lost their original value and this loss is recovered using the key.

Keywords: *Image Encryption, Image Steganography, Histogram, RGB, Inter-pixel.*

1. INTRODUCTION

Image encryption is important in this era when we have started using images extensively. Images are widely used in several business verticals to keep the information secure from any possible image theft. Researchers have developed many image encryption algorithms which adopt different techniques to encrypt the images and get the desired output. Image encryption algorithm may or may not return the plain image back after the decryption process and this basically classify the image encryption algorithm as lossy algorithm where we lose the certain amount of data when we get the plain image back or lossless algorithm where we get the plain image back intact out of the entire image encryption process.

An image encryption [1] algorithm may work on the entire image together or can break the images into blocks and execute each block. Image encryption algorithms can be block based, bit based, or entire image based.

In this paper, we will discuss a new image encryption algorithm which will focus on the movement of RGB values within the entire image along with the value transformation procedure. Value transformation procedure enables to hide the original pixel values and replace the original value with a precalculated value. This calculation is very important to get the plain image back out of the encryption process, otherwise we may not get the plain image.

Image encryption [2] is a simple process where an image is taken and that is passed to the encryption process. Encryption process has several steps to encrypt the image along with a predefined key. Key helps to execute the steps. The same key should be present with the received end to decrypt the image to get the plain image. A simple encryption process and the decryption process is shown below.

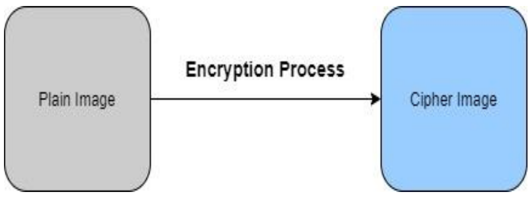


Fig 1.1: Image encryption process

Fig 1.2: Image decryption process

This paper is organized into mainly 3 sections followed by Introduction. Section 2 discusses the research work in the field of image encryption and other related work. Section 3 discusses the proposed work. Section 4 shows the experimental results. Code for this algorithm has been implemented in GNU Octave. Later we concluded this research work.

2. LITERATURE REVIEW

Here is the review of a few image encryption algorithms that were proposed lately.

Ratnakirti Roy, Shabnam Samima, Suvamoy Changder [3] explains one image encryption algorithm in which the authors used image steganography [4] technique to encrypt the image. The whole process had two steps embedding and extraction. In this paper, authors took one stego image which they used for image steganography purposes. Authors extracted the R, G and B plane out of the stego image, and they mixed these R, G and B planes with R, G and B planes of plain image or the host image. In this way, they performed the image steganography. Authors later presented the histogram analysis [5] of different planes where the Red plane did not change, but there were changes in the Blue and the Green planes.

Amnesh Goel, Nidhi Chandra [6] presented one image encryption algorithm in which they used the block permutation scheme along with the inter-pixel movement of R, G and B values. Block permutation was based on the 1:2:3 ratio where the original image was divided into the n number of blocks of n x n size and each of these blocks were moved horizontally and vertically using 1:2:3 format. After this step, the intermediate image was going under the inter-pixel encryption algorithm to encrypt the image further. Later authors ran this algorithm on a variety of images with different sizes of blocks. Three versions of this algorithm were presented where blocks were taken of size 10 x 10, 60 x 60, and 100 x 100.

In this image encryption algorithm research paper Zhi, L., Fen, S.A., and Xian, Y.Y. [7] proposed an algorithm which basically works on the steganography. They proposed to embed the bits of images in the least significant bit (LSB) position of the pixel values. In this paper authors proposed a method which they named gradient energy-flipping rate detection (GEFR). GEFR is a relation between the length of an embedded message and the gradient energy.

Amnesh Goel and Nidhi Chandra [8] presented this paper with a use case of usage of images in the medical industry and safety of images. This paper discussed the two important aspects of medication images i.e. secure storages of images and integration with PACS (Picture archiving and communication system) [9] and zooming option by which medical practitioners can zoom at a point in the medical imaging system.

Rui Zhang and Di Xiao [10] proposed an image encryption algorithm in 2020 based on the compressed sampling and chaos. This encryption method uses sampling, compression, and encryption framework. Authors presented the plain image, compressed sampling based encrypted image, final encrypted image, and the plain image which they received out of the decryption process. Authors also presented the histogram analysis, Correlation analysis, key sensitivity analysis in their paper to support their results.

Arwa Benlashram, Maryam Al-Ghamdi, Rawan AlTalhi and Pr. Kaouther Laabidi [11] proposed a novel image encryption technique in which authors performed the image encryption in three steps. The first step processed the image using pixel shuffling technique. The second step processed the image using a XOR operation between the output of the first step and a key. This key was generated using the mathematical equation. And the last step used 3D chaotic map technique to get the cipher image. Later in this paper authors presented the experimental results where they showed the image that they received using the decryption process and this image looks like the plain image. Authors also presented the histogram analysis in the paper where histograms varied from plain image and cipher image. However, if they have not changed the pixel values then ideally histograms should not change.

In this paper [12] Eugenijus Margalikas and Simona Ramanauskaitė proposed an image encryption technique which is based on image steganography. In this paper authors did not use any stego image for the encryption, rather they created a sub-cube of RGB color planes. Instead of using a reference image, authors proposed to segregate all colors (x, y, z) into RGB cube. If any color (x, y, z) is not present in the image then the value will be zero. And later each sub-cube is recursively processed. This algorithm does not change the pixel value, which is commonly used technique in image steganography, instead authors changed the location of color in the RGB cube using image color palette transformation.

In this paper [13] Karthikeyan B, Asha S, Poojasree B proposed a data hiding technique inside the color image using LSB embedding [14] [15]. Authors developed this encryption technique to conceal the text messages in the images. To execute the procedure, binary conversion was proposed to perform both image and the textual data and in the next step, LSB substitution was performed. However, the overall description of the proposed method lacks detailing of approach. MSE and PSNR [16] data was shown to validate the approach but still detailing of the proposed approach is somewhat missing in the entire paper.

In 2019, Amnesh Goel, Dr. Rakesh Bhujade [17] presented a review study of image encryption techniques. This paper covered the latest image encryption techniques such as Chaotic Encryption Scheme powered by 2D cat and S-box [18], Chaotic logistic map with a sequence in reverse order [19] where not all the pixels are diffused using the same key. First pixel is diffused using the key and the next pixel is diffused with the key and the first pixel. Another interesting image encryption technique which was included is based on watermark image [20] where a domain watermark image is embedded into the plain image to get the partial encrypted image. This watermark embedding technique resembles a little bit with the proposed image encryption technique because watermark embedding results to change in the original image pixel values.

3. PROPOSED ALGORITHM

Proposed algorithm has a three-step process of image encryption where the first step and the third step perform the same set of operations but on different inputs, and the second step belongs to the value transformation. Value transformation is done based on a predefined value and this is part of the encryption key. Following section describes all the 3 steps in detail.

Step 1: This first step is executed in two parts. The entire image is first divided into RGB planes from a given color image and then encryption is performed using the RGB inter-pixel encryption algorithm. In this step, the color image is taken and divided into its original Red, Green and Blue planes as the very basic step after reading the image. Following set of commands [21] can be executed in Matlab [22] to get the Red, Blue, and Green plane of a color image.

```
% Read in original RGB image.
plainRGCImage = imread(abc.jpg);

% Extract color channels.
redChannel = plainRGCImage(:,:,1); % Red channel
greenChannel = plainRGCImage(:,:,2); % Green channel
blueChannel = plainRGCImage(:,:,3); % Blue channel
```

These individual planes (i.e. redChannel, greenChannel and blueChannel) are then processed pixel by pixel for the entire image and these pixels are permuted in the horizontal and the vertical direction based on the key. Key decides the horizontal and/or vertical movement of a pixel value. And, to enhance the security, we took the different keys for Red, Green and Blue planes so that it becomes hard to guess the key. Also, at this step, it is not a linear execution between horizontal and vertical movement. In the next section, we will discuss how the horizontal and vertical movement is derived using the key. The same process repeats for all three planes until we receive an encrypted image.

Key details for this step: In this step, we used a key of 14 bits for each plane i.e. 14 bits for Red, 14 bits for Blue and 14 bits for the green plane. These 14 bits can be read as a segment of 1:3:4:6. First bit signifies the starting movement. It could be the horizontal movement at the beginning or the vertical movement. Initial 0 can represent the horizontal movement and a 1 can represent the vertical movement.

Next 3 bits represent the pattern in which horizontal and vertical movement will be performed. 3 bits can result in 8 movement combinations ($2^3 = 8$). Each of these 8 combinations has a specific format of horizontal and vertical mix.

Next 4 bits represent the number of times this execution will be performed. These 4 bits results in 16 combinations ($2^4 = 16$). For example, 1001 will define that iterations will be performed 9 times.

Remaining 6 bit represent the target pixel location. For example, if we are at pixel location [1, 1] and we are performing a horizontal movement. Then the pixel value of location [1, 1] will move to location [1, x]. This x will be determined by the remaining 6 bits. So, if the last 6 bits are represented as 110001 then the value of x will be 49 i.e. [1, 49]. This same pattern is then used for the other two planes as well. So, in total, at this step we are using a total key size of 42 bits.

Step 2: In the second step, we performed the value transformation process. In this step, we continued to work on the output of previous step and continued with all three planes i.e. Red, Blue, and Green. For each plane, we took a number which was either added or subtracted or multiplied with the pixel value of said planes. This value transformation mechanism differs for each plane. For example, in the Red plane, we may add the value whereas in the Green plane we may subtract a fix value. This addition, subtraction or multiplication becomes the part of the key. This step is added in this encryption scheme carefully to further confuse the encrypted image.

Key details for this step: In this step, we used a key of 8 bits for each plane i.e. 8 bits for Red, 8 bits for Blue and 8 bits for the green plane. These 8 bits can be read as a segment of 2:6. The first two bits derive the mathematical function which will be performed on the pixels i.e. addition, subtraction, or multiplication. 2^2 results into the following 4 combinations.

Table 1: Mathematical operation combinations for 2^2

Combination	**Mathematical operation**
00	Nothing (not applicable)
01	Addition
10	Subtraction
11	Multiplication

Next 6 bits represent the value which will be used in the mathematical operation. This same pattern is then used for the other two planes as well. So, in total, at this step we are using a total key size of 24 bits.

Following pseudocode represents the value transformation step for pixel location [1, 1].

1. Start
2. Read pixel value for location [1, 1].
3. Read the combination from key. This will determine the mathematical function.
4. Read the operand from the next 6 bits.
5. Perform the calculation.
6. Save the new value.
7. End

Following equation is a symbolic representation of an 8-bit key with respect to the pixel location [x, y].

img [x, y] = img [x, y] [0 1] [100001] 1

Above equation 1 can be translated as the following (considering the value of img [x, y] is 190).

img [x, y] = 190 + 33
img [x, y] = 223

Step 3: And, in the third or last step, we again encrypted the image using RGB inter-pixel encryption algorithm (this is repetition of step 1). This whole algorithm makes a sandwich of 3 processes where the first and last process remains the same and the middle steps acts as a confusion which further makes it difficult to decrypt the image without knowing the value transformation step keys. This step also uses a key of 42 bits. For description related to key size, please refer to step 1. At last, we combine all three planes to get the complete cipher image. Following code in Matlab can be used to get the color image back from individual red, green and blue planes.

```
% Recombine separate color channels into an RGB image.
rgbImage = cat(3, redChannel, greenChannel, blueChannel);
```

So, this image encryption uses a key of 108 bits. 42 bits are used in the first step, 24 in the second step and 42 in the last step.

The Decryption process is just the reverse of the encryption process, but this needs the key to get the plain image back. Any deviation in key or in the execution steps will not return the plain image in original format and will further encrypt the image which will make it impossible to get the plain image.

Following is the overall flow of steps in the proposed algorithm w.r.t Encryption and Decryption process.

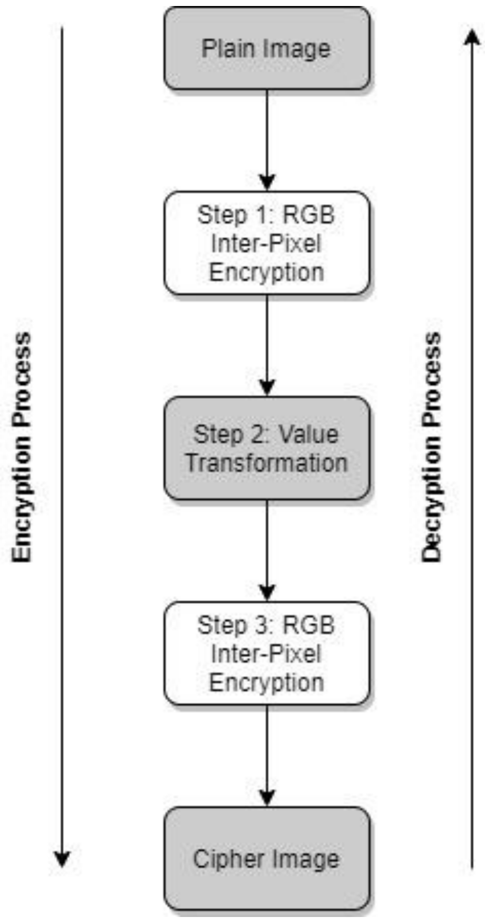

Plain Image
Step 1: RGB Inter-Pixel Encryption
Step 2: Value Transformation
Step 3: RGB Inter-Pixel Encryption
Cipher Image
Encryption Process
Decryption Process

Fig 3.1: Proposed algorithm functional steps

4. EXPERIMENTAL RESULTS

This section shows the experimental results which were obtained after implementing the proposed algorithm in the GNU Octave [23] software. For this experiment, we took 2 different images of different sizes. Our first image is of a flower and the size of this image is 500 x 750. The second image is a “leena” image which is widely used in the image encryption algorithms and the size of this image is 512 x 512.

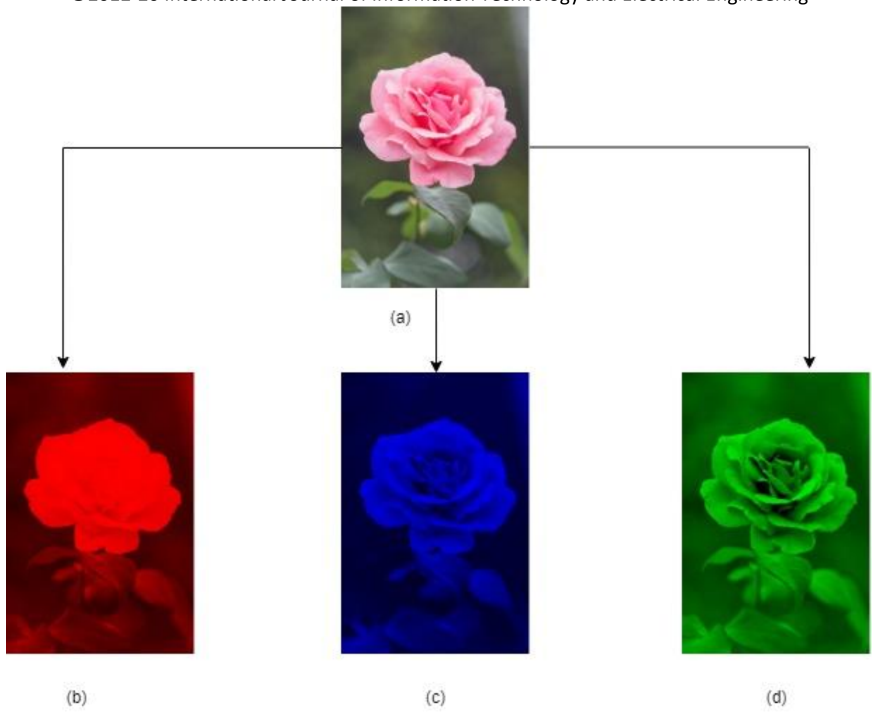


Fig 4.1 (a) Plain Image of flower (size 500 x 750), (b) Red plane from plain image, (c) Blue plain from plain image and (d) Green plain from plain image.

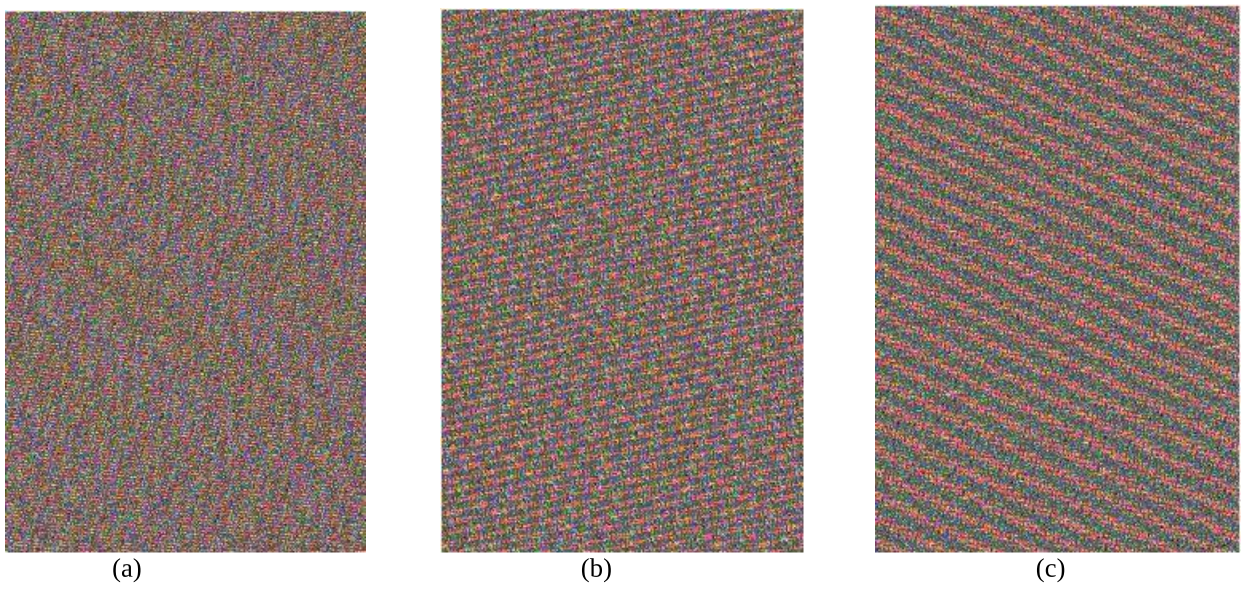


Fig 4.2 (a) shows the encrypted flower image after execution of step 1, (b) shows the encrypted image after execution of step 2 and (c) shows the encrypted image after execution of step 3.

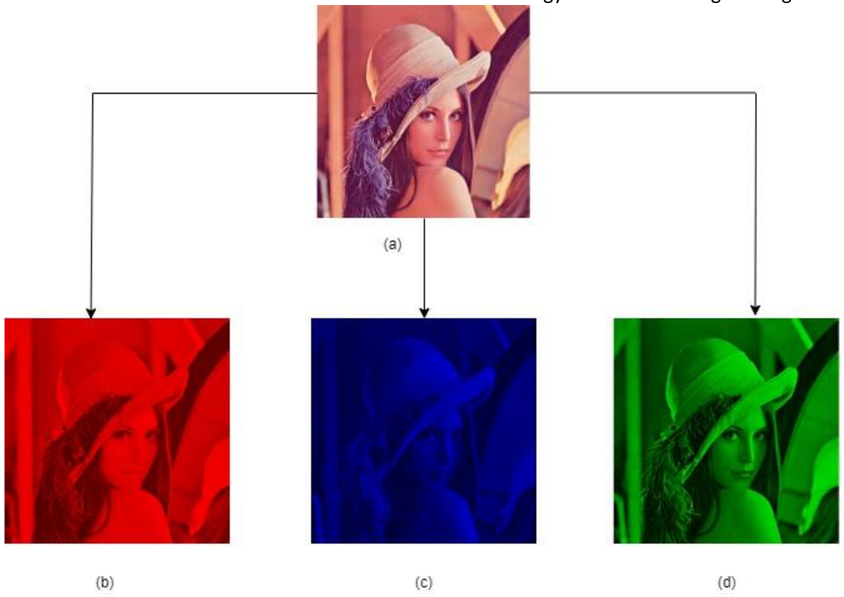


Fig 4.1 (a) Plain Image of "leena" (size 512 x 512), (b) Red plane from plain image, (c) Blue plain from plain image and (d) Green plain from plain image.

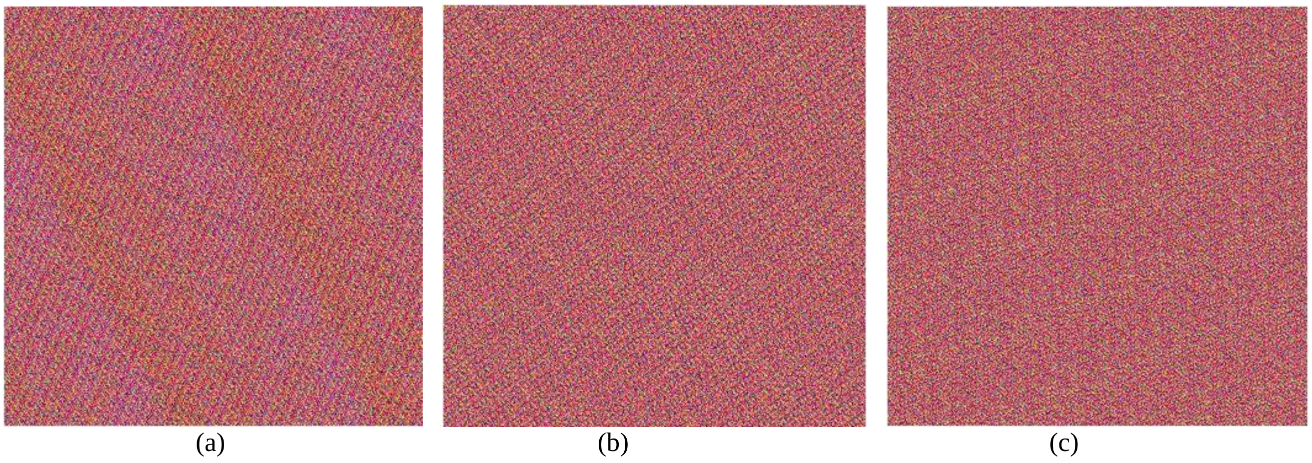
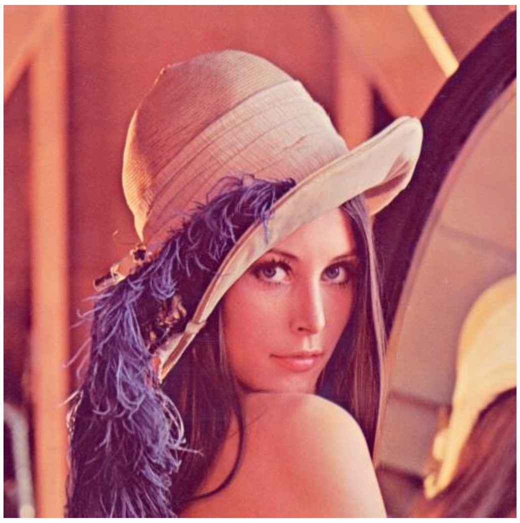
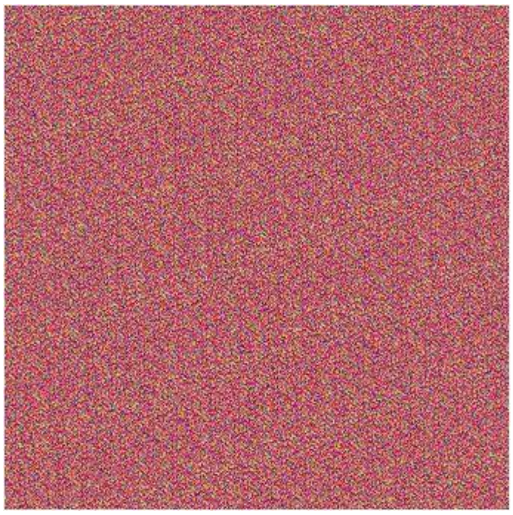


Fig 4.4 (a) shows the encrypted "leena" image after execution of step 1, (b) shows the encrypted image after execution of step 2 and (c) shows the encrypted image after execution of step 3.

Figure 4.1 (a) shows the plain image of a flower which we took for encryption. Figure 4.1 (b) shows the Red plane of original color flower image, figure 4.1 (c) shows the blue plane of original color flower image and figure 4.1 (d) shows the green plane of original color flower image. We proposed this algorithm to function in 3 steps and figure 4.2 shows the results after each step. Figure 4.2 (a) shows the encrypted image of the flower that we got after execution of step 1, figure 4.2 (b) shows the encryption state after we performed value transformation step and figure 4.2 (c) shows the final encrypted image from this image encryption algorithm. Figure 4.3 and 4.4 shows the similar behavior of image encryption for a different image. In the next few sections, we will discuss the outcome of this image encryption algorithm with respect to different analysis techniques.

4.1 Encryption Validation (Pixel Loss Study): To validate the proposed algorithm, we did study the pixels of plain image and the pixels of output obtained by the decryption process. We did this comparison at the RGB level and we got the 100% values as is after the decryption process. This confirms that this proposed image encryption algorithm has no pixel loss [24] during execution. We then ran this algorithm on a couple of more images to verify the output and we had the same results. To further validate these results, we took images of different sizes and the results remain the same.


(a)


(b)

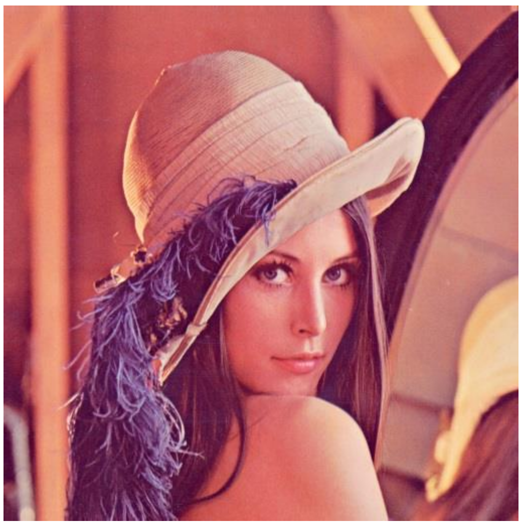
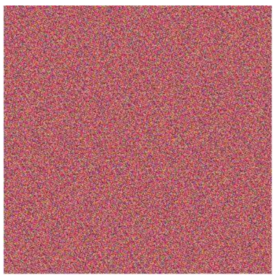

(c)

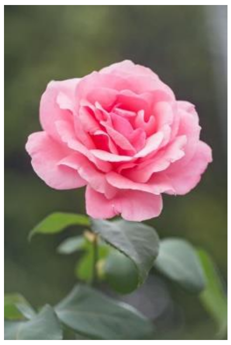
Fig 4.5 (a) shows the plain image that we took for the encryption, 4.5 (b) shows the encrypted image that we received as output of this image encryption algorithm and 4.5 (c) shows the decrypted image.

4.2 Key Sensitivity Analysis: An algorithm can be defined as a good image encryption algorithm if we do not get the plain image back out of the decryption process even if we make a slight change in the key. We also performed the key sensitivity analysis on this image encryption results. We changed the key value by one digit and in result we did not get the plain image back.

Fig 4.5 (b) shows the encrypted image and Fig 4.5 (c) shows the plain image that we received out of the decryption process. We used our original key (let us say Key1) to get the plain image in this decryption process. When we changed the key1 to Key2 then we got the following image out of the decryption process. And following results confirm the key sensitivity to get the plain image back from its original key.

(a)


(b)


(c)

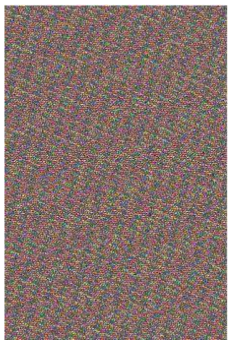
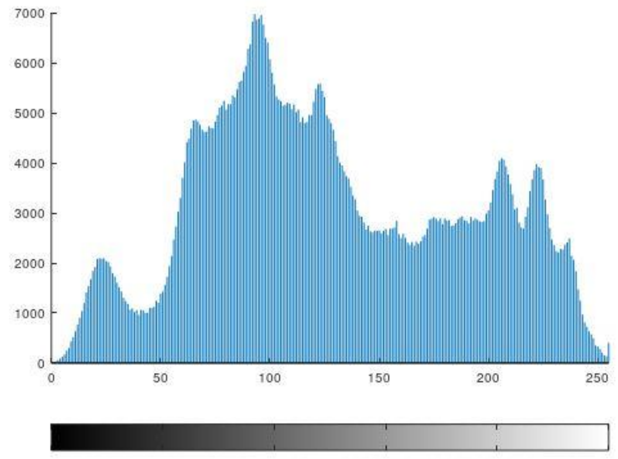
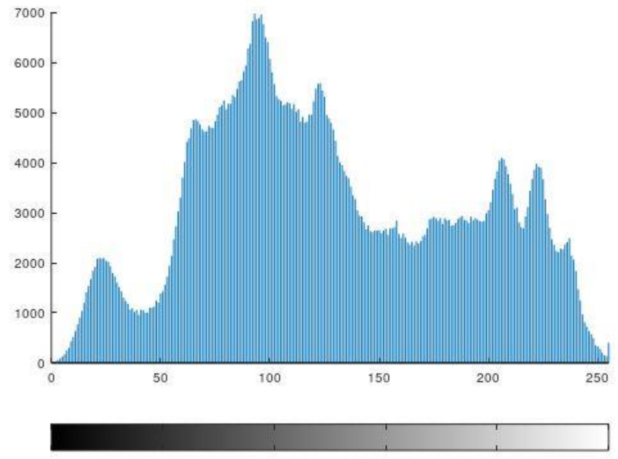

(d)

Fig 4.6 (a) shows the decrypted image that was received when we used the original key for "leena" image and figure 4.6 (b) shows the related decrypted image when we changed the key1 to key2. Figure 4.6 (c) shows the decrypted image that was received when we used the original key for "flower" image and figure 4.6 (d) shows the related decrypted image when we changed the key1 to key2.

4.3 Histogram Analysis: In this section we will see the different histograms that we have created out of the outputs of this image encryption algorithm. Following histograms belongs to the "leena" image and its encrypted image.


(a)


(b)

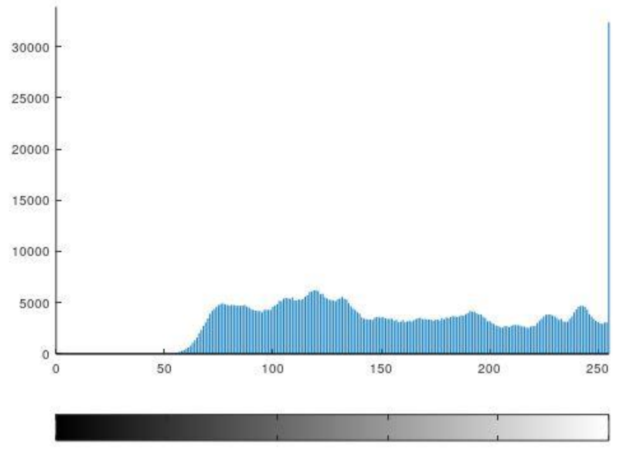
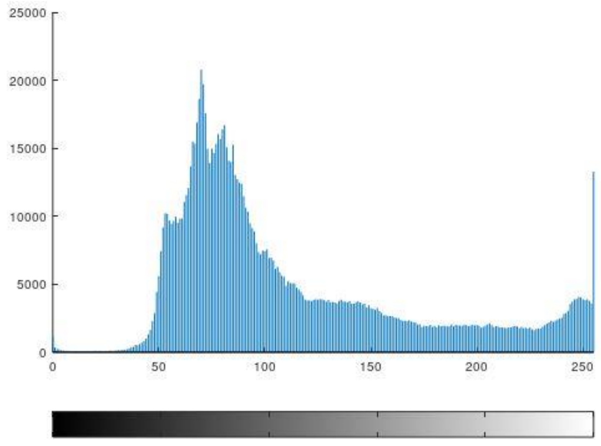
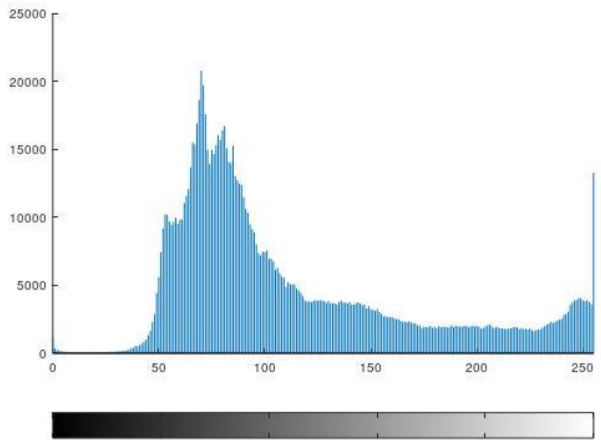

(c)

Fig 4.7 (a) shows the histogram of plain "leena" image, figure 4.7 (b) shows the histogram of encrypted image that we received after step 1 execution and figure 4.7 (c) shows the histogram of after value transformation step.

Following histograms belong to the "flower" image and its encrypted image.


(a)


(b)

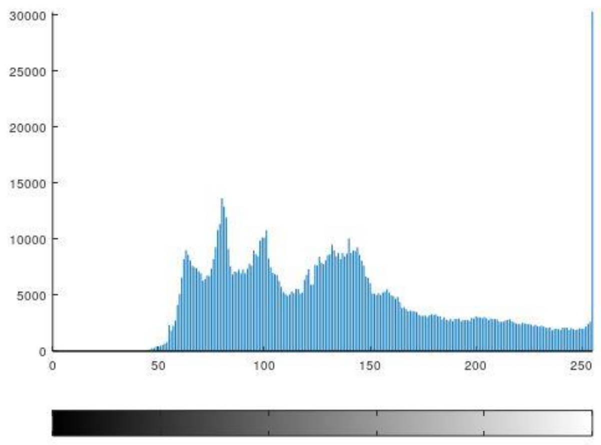
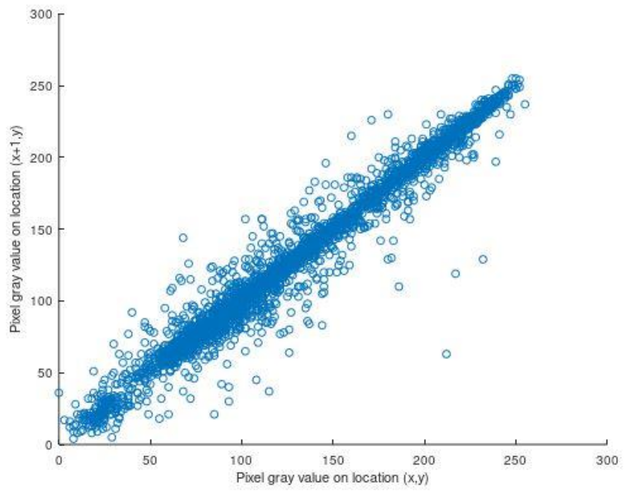

(c)

Fig 4.8 (a) shows the histogram of plain "leena" image, figure 4.8 (b) shows the histogram of encrypted image that we received after step 1 execution and figure 4.8 (c) shows the histogram of after value transformation step.

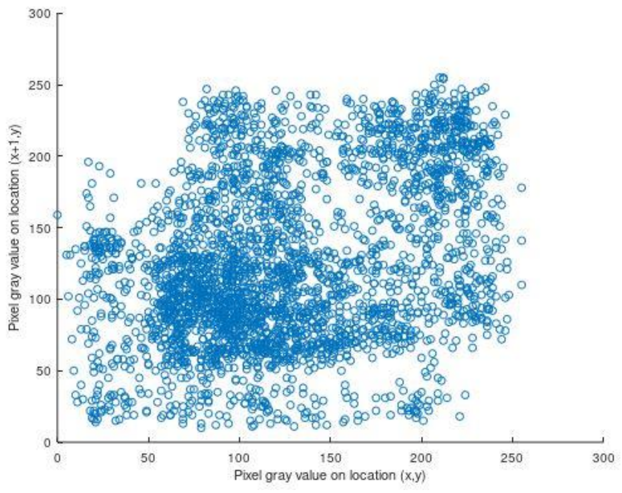
From above images it is evident the histogram of plain image and the encrypted image (which we got after execution of step 1) appeared similar and had no difference. However, when we performed the visual transformation [25] step then the histogram changed and change in histogram is clearly visible in figure 4.7 (c) and figure 4.8 (c) for "leena" and "flower" image, respectively.

4.4 Time Complexity Analysis: We executed this algorithm on 12 images of different sizes and the average time taken by the encryption process alone varied between 80 to 115 seconds on a i5 Pentium Processor with 16 GB Ram on Windows 10 Home Edition.

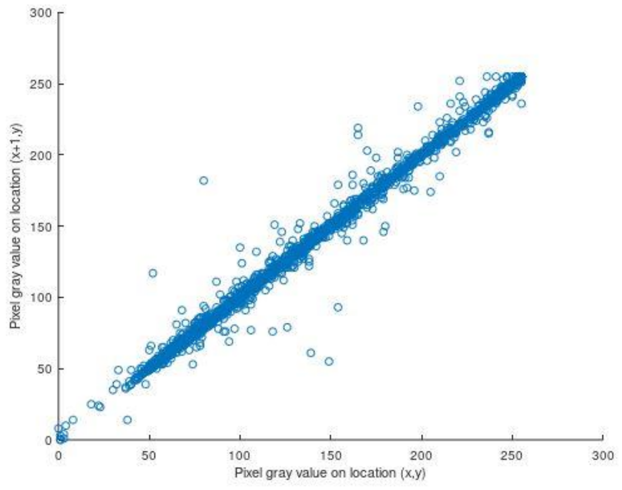
4.5 Adjacent Pixel Correlation Analysis: We performed the adjacent pixel correlation analysis on the plain image and the final encrypted image and following are the results.



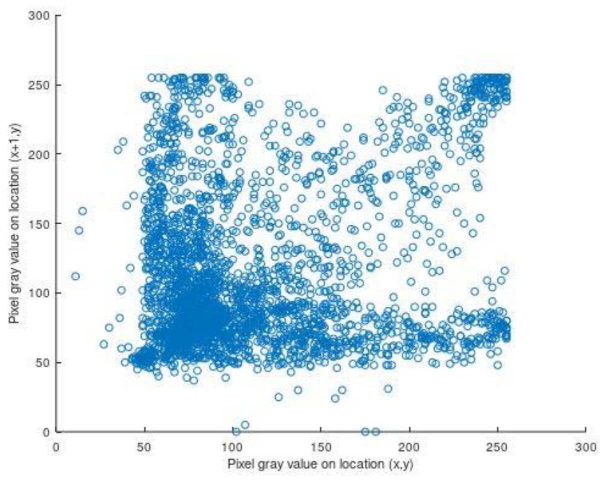
(a)



(b)



(c)



(d)

Fig 4.9 (a) shows the correlation between pixels of "leena" plain image and figure 4.9 (b) shows the correlation between pixels of "leena" encrypted image of step 1. Figure 4.9 (c) shows the correlation between pixels of "flower" plain image and figure 4.9 (d) shows the correlation between pixels of "flower" encrypted image of step 1.

Above images of correlation analysis show the correlation between pixels started to decrease from the step 1 of this image encryption technique. This is a good analysis technique to understand how pixels are correlated with each other and if pixel values do not show signs of correlation then that image encryption algorithm can be termed as a good image encryption algorithm.

5. CONCLUSION

In this paper, we suggested a new image encryption algorithm which is based on the inter-pixel movement of original values along with the value transformation based on a predefined key. Value transformation technique which is part of image steganography helped to see the difference in histograms presented in this paper. A histogram which was taken after performing value transform clearly showed that the value of a pixel had changed from its original value. Also, this value is recoverable as this was derived using the key. Overall, this was a three-step process, and this worked well to encrypt the image. Key sensitivity analysis confirmed the change in key resulted in a very different image.

The main strength of this paper is the value transformation step because if we generally perform the inter-pixel permutation algorithm then that alone does not confuse the decryption process much. In this algorithm, at the decryption side, one really needs to guess when to stop the inter-pixel permutation to perform the value transformation step. Any wrong guess will ruin the complete efforts. This make it difficult to crack this algorithm even using the brute force attack [26].

FUTURE RESEARCH

In future, this paper can be used as a base paper to perform an extended research. This research can be combined with a plane of a stego image passed either from the sender or a fixed stego at the received side. Or this research analogy can be used as base to create a more comprehensive image encryption technique.

REFERENCES

[1] Encryption - https://en.wikipedia.org/wiki/Encryption

[2] Narasimhan Aarthie and Rengarajan Amirtharajan, "Image Encryption: An Information Security Perceptive", Journal of Artificial Intelligence, Volume 7 (3): 123-135, 2014.

[3] Ratnakirti Roy, Shabnam Samima, Suvamoy Changder, "A map-based image steganography scheme for RGB images", Int. J. Information and Computer Security, Vol. 7, Nos. 2/3/4, 2015, Copyright © 2015 Inderscience Enterprises Ltd.

[4] Steganography: Hiding an image inside another - https://towardsdatascience.com/steganography-hiding-an-image-inside-another-77ca66b2acb1

[5] Histogram analysis - https://blog.minitab.com/blog/3-things-a-histogram-can-tell-you.

[6] Amnesh Goel, Nidhi Chandra, "A technique for image encryption based on explosive n*n block displacement followed by inter-pixel displacement of RGB attribute of a pixel ", Proceedings - International Conference on Communication Systems and Network Technologies, CSNT 2012.

[7] Zhi, L., Fen, S.A., and Xian, Y.Y. (2003), "A lsb steganography detection algorithm", 14th IEEE Proceedings on Personal, Indoor and Mobile Radio Communications, (PIMRC 2003), September, Vol. 3, pp.2780–2783.

[8] Amnesh Goel and Nidhi Chandra, "A Prototype Model for Secure Storage of Medical Images and Method for Detail Analysis of Patient Records with PACS," 2012 International Conference on Communication Systems and Network Technologies, Rajkot, 2012, pp. 167-170, doi: 10.1109/CSNT.2012.217.

[9] PACS (picture archiving and communication system) - https://searchhealthit.techtarget.com/definition/picture-archiving-and-communication-system-PACS - Last accessed on 29 June 2020.

[10] Rui Zhang and Di Xiao, "A secure image permutation–substitution framework based on chaos and compressive sensing", International Journal of Distributed Sensor Networks 2020, Vol. 16(3) The Author(s) 2020 DOI: 10.1177/1550147720912949 journals.sagepub.com/home/dsn.

[11] Arwa Benlashram, Maryam Al-Ghamdi, Rawan AlTalhi and Pr. Kaouther Laabidi, "A novel approach of image encryption using pixel shuffling and 3D chaotic map", Arwa Benlashram et al 2020 Journal of Physics: Conference Series 1447 012009.

[12] Eugenijus Margalikas & Simona Ramanauskaitė "Image steganography based on color palette transformation in color space", EURASIP Journal on Image and Video Processing 2019, 82 (2019), https://doi.org/10.1186/s13640-019-0484-x

[13] Karthikeyan B, Asha S, Poojasree B, "Gray Code Based Data Hiding in an Image using LSB Embedding Technique", International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-8, Issue-1, May 2019.

[14] Arun Kumar Singh, Juhi Singh, Dr. Harsh Vikram Singh, "Steganography in Images Using LSB Technique", International Journal of Latest Trends in Engineering and Technology (IJLTET), Vol. 5 Issue 1 January 2015, ISSN: 2278-621X.

[15] S. Sugathan, "An improved LSB embedding technique for image steganography," 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), Bangalore, 2016, pp. 609-612.

[16] PSNR - https://in.mathworks.com/help/vision/ref/psnr.html - Last accessed on 26th April 2020

[17] Amnesh Goel, Dr. Rakesh Bhujade, A Functional Review of Image Encryption Techniques, International Journal of Scientific & Technology Research Volume 8, Issue 09, September 2019, ISSN 2277-8616.

[18] Muhammad Asim, Varun Jeoti, "On Image Encryption: Comparison between AES and a Novel Chaotic Encryption Scheme" IEEE - ICSCN 2007, MIT Campus, Anna University, Chennai, India. Feb. 22-24, 2007. pp.65-69.

[19] Jean De Dieu Nkapkop, Joseph Yves Effa, Monica Borda, Laurent Bitjoka, Alidou Mohamadou, "A Secure and Fast Chaotic Encryption Algorithm Using the True Accuracy of the Computer" Informatica 40 (2016) 437–445.

[20] Ali Al-Haj, Hiba Abdel-Nabi, "Digital Image Security Based on Data Hiding and Cryptography" 2017 IEEE 3rd International Conference on Information Management 978-1-5090-6306-2/17 © 2017.

[21] How do I split a color image into its 3 RGB channels? - https://www.mathworks.com/matlabcentral/answers/91036-how-do-i-split-a-color-image-into-its-3-rgb-channels - Last accessed on 28 June 2020.

[22] Matlab - Image Processing Toolbox - https://www.mathworks.com/products/image.html - Last accessed on 19 June 2020.

[23] GNU Octave - https://www.gnu.org/software/octave/

[24] Sara Tedmori and Nijad Al-Najdawi, "Lossless Image Cryptography Algorithm Based on Discrete Cosine Transform", September 2012International Arab Journal of Information Technology 9(9):471-478.

[25] Arindrajit Seal, Shouvik Chakraborty, Kalyani Mali, "A New and Resilient Image Encryption Technique Based on Pixel Manipulation, Value Transformation and Visual Transformation Utilizing Single–Level Haar Wavelet Transform", Proceedings of the First International Conference on Intelligent Computing and Communication pp 603-611, Part of the Advances in Intelligent Systems and Computing book series (AISC, volume 458).

[26] Brute Force Attacks Defined - https://www.forcepoint.com/cyber-edu/brute-force-attack - Last accessed on 24th June 2020.