Implementation of LSB Steganography and its evaluation for various File Formats

Kartik Malik[1], Rahulkumar Ankola[2], Areeb Afzal Kanth[3], Aditi Akhauri[4]

[1]Computer Science and Engineering, [2]Computer Science and Engineering, [3]Computer Science and Engineering with specialization in Blockchain Technologies, [4]Computer Science and Engineering with specialization in Information Security

[1]kartik.malik2020@vitstudent.ac.in, [2]rahulkumar.ankola2020@vitstudent.ac.in, [3]areebafzal.kanth2020@vitstudent.ac.in, [4]aditi.akhauri2020@vitstudent.ac.in

Abstract

Steganography is derived from the Greek word "steganos" literally meaning “Covered” and "graphy", meaning “Writing”, which makes it covered writing. Steganography refers to the science of "invisible" communication. For hiding confidential information on various file formats, there are a wide variety of steganographic techniques, some of which are very complex than others and they all have their strong and weak points. Embedding using the Least Significant Bit (LSB) strategies suggest that data can be hidden in small parts of the cover image and the human eye can’t detect the hidden data in the cover file. This process can be used to hide images in 24-Bit, 8-Bit and even Grayscale format. Text files can also be hidden in various image file formats using this method. This paper describes how to embed using the LSB techniques and introduce it to various file formats, in order to embed both text files and image files.

Keywords: Steganography, Least Significant Bit (LSB), Text files, Image files

1. Introduction

Digital content now poses major challenges to its content developers, compilers, distributors and users. Damage, withdrawal or certain alterations of the embedded message is required in order to improve durability and robustness systems for digital content processing so that the organizing and processing of the content becomes simpler and frictionless.

Cryptography has been created as a shield to protect and maintain the secrecy of communication with many different ways designed and developed overtime to encrypt and decrypt data, in order to keep the message private. Unfortunately, it is sometimes not enough to only keep the content of the message private; it may also be necessary to maintain the presence of message secret. The method used to do this, is called steganography.

The transition from cryptography to steganography is due to the need of concealing the secret data’s presence in the image as steganographic image which in turn allows it to embed the secret data into the cover images. In conceptual terms, steganography helps in transmission of the messages without it being visible to the naked eye. Steganography has been used for thousands of years to transfer data without unwanted interruptions, illegal viewing of the data and modifications on the data by unauthorized people (attackers). It is an art of concealing information inside information. The main purpose of Steganography is most concerned with the protection of the contents embedded or hidden. Images are often considered as a good or in fact, ideal medias to hide information because of the redundant nature of the space created while storing these images. In Steganography, the secret messages are carried and transmitted by anonymous cover carriers in a way that it is difficult to predict existence of the secret messages. The anonymous carriers include images, audio, video, text or any other code represented digitally. Hidden message may be a plain text, an encrypted text or anything that can be represented in the form of a bit-stream.

2. Literature Survey

SERIAL NO.	**AUTHOR**	**YEAR OF PUBLISHING**	**TITLE**	**METHODOLOGY**	**METRICS**	**DRAWBACKS**
1.	V. Lokeswara Reddy, Dr. A. Subramanyam, Dr. P. Chenna Reddy	2010	Implementation of LSB Steganography and its evaluation for various File Formats	· The Objective of this study is to implement Least Significant Bit Steganography on various file formats. · The original image and the image to be hidden is read and the image to be hidden is shifted by x bits. · The MSB of the cover image is set according to the image. The no. of LSB's to be considered accordingly. . The shifted hidden image and the result from the previous step is bitored which only makes changes in the original image.	Mean-Squared Error (MSE). Peak Signal-to-Noise Ratio Steganalysis Detection Payload Capacity Independence of the file format Perceptibility Percentage Distortion less resultant image	· It is vulnerable to steganalysis and hence not secure at all. . Has low-security since in this, the watermark is embedded into single bit of original message only.

2.	Nan-I-Wu, Min-Shiang Hwang	2017	A novel LSB data Hiding scheme with the lowest distortion	. During encoding of the data, 3 secret bits are embedded in three pixels of the original image. · The XOR operation is performed between the least significant bits of the original image. . The pixels are then adjusted after comparing the XOR values with secret bit values.	Peak Signal-to-Noise Ratio Mean-Squared Error No. of pixels with overflow and underflow conditions. K-Parameter Payload Capacity RS Analysis Expected no. of modification per pixel (ENMPP)	The method is comparatively complex. This technique successfully survives the statistical RS attack. However, low hiding capacity remains the major issue for this technique as it only hides 1 bit in each pixel of the cover/original image.
3.	Firas A. Jassim, Hind E. Qassim	2012	Five Modulus Method for Image Compression	· The cover image is divided into N blocks of block size k*k pixels, where 'K' is the size of window. . Then each pixel in the blocks is modified such that the pixel of block is divisible by 5.	Peak Signal to Noise Ratio (PSNR). Mean Squared Error (MSE) Compression Ratio (CR) Root Mean Square Error (RMSE)	The limitation of this method is the hiding capacity, being below 1 bit per pixel in some cases.
4.	S. Batra, R. Rishi	2010	Insertion of the message in 6th ,7th and 8th bit of pixel values and retrieval in case intruder changes the least significant bit	· A pseudo random location(l) in the cover object using secret key is found for inserting the message bit b and then we check for the pixel value at the location.	Histogram Analysis Mean Squared Error (MSE) Peak Signal-to Noise Ratio	This method does not provide 100 % message insertion rate.

			of image pixels.	. After this, the 6th ,7th and 8th bits at the location are checked.	(PSNR)	

3. System Design

3.1 Algorithm for Hiding (Steganography)

1. Read the original image and the image which is to be hidden in the original image

2. Shift the image to hide in the cover image by X bits.

3. And the original image or cover image with 240 which is 11110000 So four MSB's set to 0. Because of this, only four LSB's considered further.

4. The shifted hidden image and the result of step 3 are bitored. This makes changes only in the X LSB bits so that the image is hidden in the original image. This image can be called as the stego image

3.2 Algorithm for Steganalysis

1. The stego image is bit shifted by 4 bits since it was shifted by 4 bits to insert it into the original image.

2. The image is the ANDED with 255 i.e., 11111111, which gives the original image. It is ANDED with 255 because initially all the LSB's were made 0. Now it is recovered back.

3. To get it to Unit8 format we, convert it back to unit8 which is the extracted image.

3.3 Algorithm for LSB Embedding (For embedding text files)

1.Take input of text that is to be hidden and create a function to convert the text to binary format.

2. For the conversion of the text to binary digits, 8-bit representation of the text is taken into account.

3. Input the image in any file format where you wish to embed your message.

4. Perform the XOR condition between the 2nd LSB of each pixel (cover image) and bits of message.

5. Replace the resultant with the LSB of the cover image.

6. Repeat step 4 and 5 for every pixel (red green blue) of the cover image.

7. And the data is embedded on the cover image.

8. For retrieving the message from the embedded image, perform the XOR condition between the LSB and 2nd LSB of the image. The resultant is the bit of secret message encoded.

9. Repeat step 8 for every pixel until the message is encoded.

3.4 Algorithm for LSB Embedding (For embedding image files)

1. Input original image in any format.

2. Input watermark in any File format.

3. Resize the watermark according to original image.

4. Merge the first 6 bits of the original image and first 2 bits of the watermark image.

5. Get watermarked image.

6. To decode the embedded image from the watermark image, take the last two bits of the watermarked image and add six zeros to it. (So that it will be an 8-bit pixel and adding zeros will increase the brightness of the image).

7. Calculate PSNR and MSE value of watermarked image.

Similarly, we can merge any number of bits adding up to 8, where bits of original image >= bits in watermarked image, for example, merging 4 bits of watermarked image and 4 bits of original image.

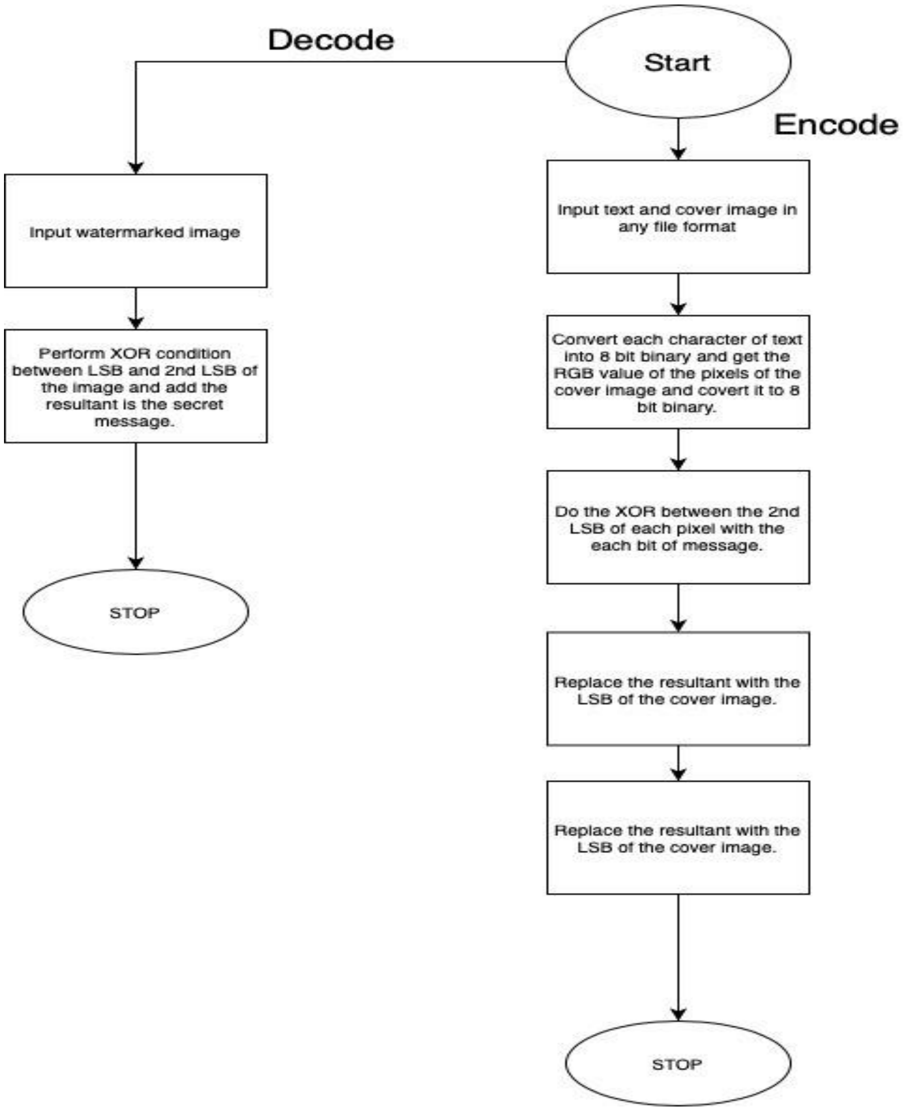

Decode
Start
Encode
Input watermarked image
Perform XOR condition between LSB and 2nd LSB of the image and add the resultant is the secret message.
STOP
Input text and cover image in any file format
Convert each character of text into 8 bit binary and get the RGB value of the pixels of the cover image and covert it to 8 bit binary.
Do the XOR between the 2nd LSB of each pixel with the each bit of message.
Replace the resultant with the LSB of the cover image.
Replace the resultant with the LSB of the cover image.
STOP

Fig. 1: *Flowchart for the algorithm used in text in image steganography.*

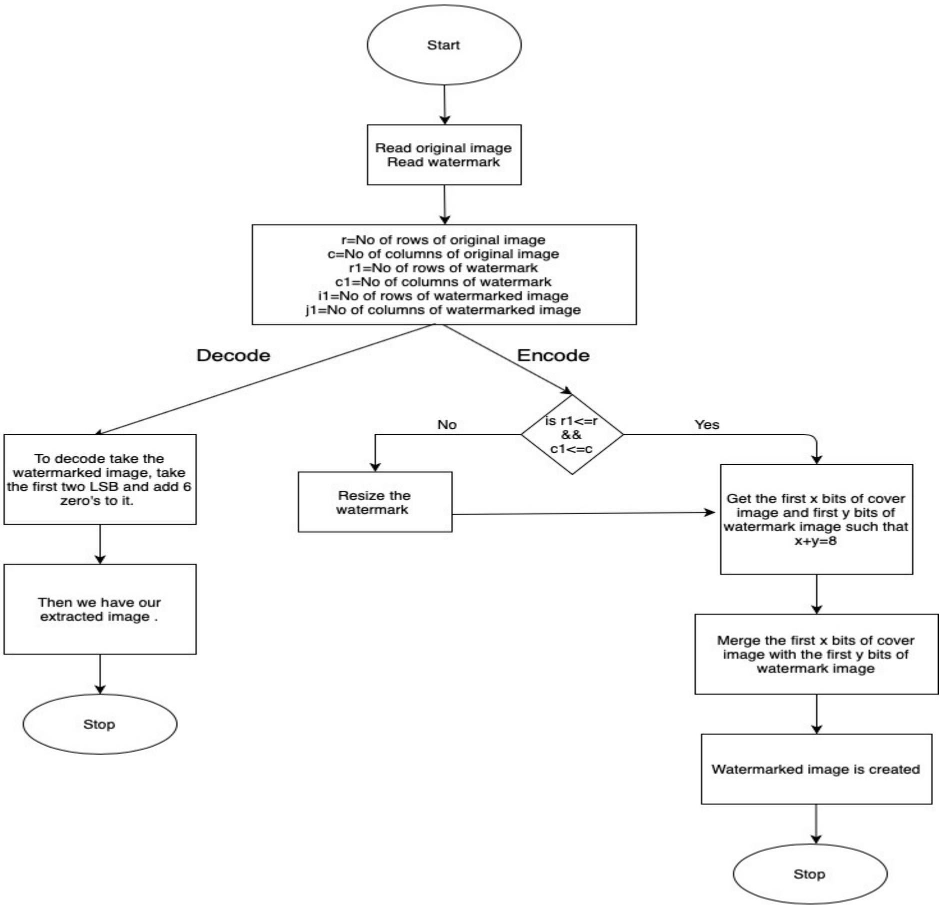

Start
Read original image
Read watermark
r=No of rows of original image
c=No of columns of original image
r1=No of rows of watermark
c1=No of columns of watermark
i1=No of rows of watermarked image
j1=No of columns of watermarked image
Decode
Encode
is r1<=r && c1<=c
No
Yes
To decode take the watermarked image, take the first two LSB and add 6 zero's to it.
Then we have our extracted image .
Stop
Resize the watermark
Get the first x bits of cover image and first y bits of watermark image such that x+y=8
Merge the first x bits of cover image with the first y bits of watermark image
Watermarked image is created
Stop

Fig.2: *Flowchart for the algorithm used in image in image steganography.*

4. Experimental Results and Analysis

There are many steganographic algorithms available. One has to choose the best available algorithm according to the given application. The following features should be checked while selecting a specific file format to carry out Steganography. Steganography says that a secret message should be hidden and must result in an image with the least distortion possible. It should be almost impossible to detect the distortion with naked eyes. The amount of data embedded in the cover also plays a crucial role. The algorithm used for steganography determines how much data can be embedded an image that leads to only slight distortion of the image. Steganalysis is a way to find the hidden information in the picture by applying various attacks on it. The Steganographic algorithm should always make sure that the Steganalysis algorithms applied fail. Namely, Steganography algorithms should not be prone to steganalysis attacks. While transferring the embedded image from one system to another, it is possible that an attacker could detect the secret information on the image and try to delete it. He/she can even modify the image. The modifications may include rotating, sharpening, adding noise etc. of the images. The modifications done may cause the image to distort. Selected steganographic algorithms should always be able to overcome such deceptions and make sure that the data reaches the destination in the needed format.

The implementation of the steganography and steganalysis for both image and text files have been done in python language. For the user's convenience, a website has been created for the same that is linked with our python code(backend) wherein we use HTML, CSS and JavaScript. We have used .png image file format for embedding the information, although formats like .bmp, .jpeg, etc. can also be used.

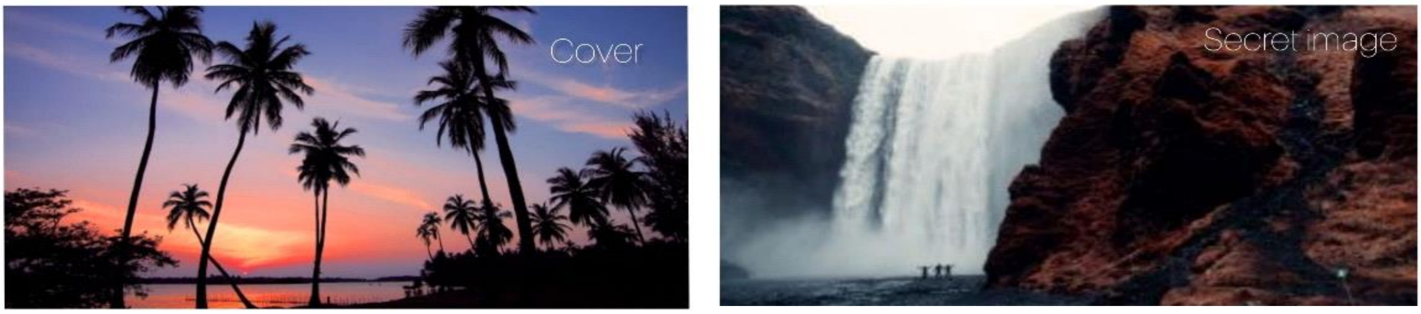
For the implementation of the image and text steganography, we have made use of two libraries, OpenCV and numpy. OpenCV is mainly used for reading and writing images whereas numpy is used to manipulate the images which are converted in the form of arrays. For the analysis of the quality of the image by using image quality measurements and robustness methods, the same libraries have been used, however, numpy has been used extensively for the comparison of the input and output images. The naïve method to implement this is by comparing every pixel of the original image with the processed image and in order to do that we would need several loops to iterate through every element. But in order to reduce the runtime and make our algorithm more efficient, we use numpy to compare the matrices of the images and perform arithmetic operations on them. Finally, in order to link our website with our python code, we make use of flask and create a local server and carried out routing of the respective web pages.

The IQM (Image Quality Measurements) used to analyse our processed images include Mean square error (MSE), Root Mean square error (RMSE), Signal-to-noise ratio (SNR), Peak signal-to-noise ratio (PSNR), Average Difference (AD), Normalized cross correlation (NCC), Image Fidelity (IF), Structural Content (SC), Normalised Absolute Error (NAE), Structural similarity index measure (SSIM), Structural dissimilarity index measure (DSSIM) and Universal Image Quality Index (UIQI). Several robustness attacks are carried out on the embedded images processed in both image in image and text in image steganography and these IQM values are compared for each attack carried out. The robustness measures used for analysis are cropping, rotating, Gaussian noise, Poisson noise, Speckle noise, Salt and Pepper noise, median filter, mean filter, sharpening and histogram equalization.

.

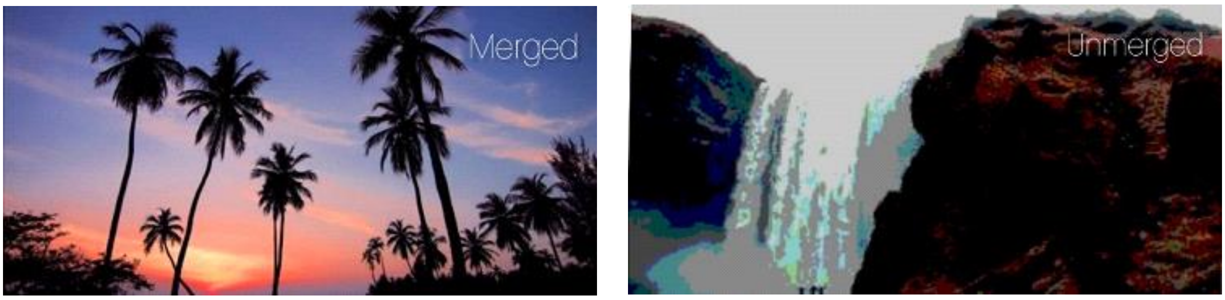
(a) *(b)*

Fig.3: *(a) Original image used and (b) embedded image found in text in image steganography*



(a) *(b)*

Fig.4 :*(a) cover and (b) secret images used as input*



(a) *(b)*

Fig.4: (a) *the merged and (b) unmerged result found in image steganography.*

The IQM values calculated for each robustness attack carried out on text in image steganography:

Input image (.png)	**MSE**	**RMSE**	**SNR**	**PSNR**	**AD**	**NCC**	**IF**	**SC**	**NAE**	**SSIM**	**DSSIM**	**UIQI**
Embedded image	1.09	1.044	24.874	47.757	139.342	0.63	0.37	0.99	0.537	0.0355	0.482	0.024
After Cropping	1.21	1.1	24.365	47.303	154.624	0.642	0.358	0.987	0.56	0.051	0.474	0.017
After Rotating	308.206	17.556	0.361	23.242	380.349	0.75	0.25	0.99	1.465	0.021	0.489	0.01
After adding Gaussian Noise	280.038	16.734	0.777	23.659	370.831	0.737	0.263	0.969	1.428	0.045	0.477	0.013
After Adding Poison Noice	30227.822	173.862	-19.555	3.327	-259.574	10190335.005	-10190334.005	358.792	1.0	454414.841	-227206.921	456504.262
After adding Speckle Noice	104.618	10.228	5.053	27.935	217.089	0.537	0.463	0.82	0.836	0.029	0.485	0.0
After adding Salt &Pepper Noice	14.214	3.77	13.722	36.604	142.607	0.625	0.375	0.95	0.549	0.031	0.484	0.02
Median filter	29.196	5.403	10.596	33.478	225.791	0.673	0.327	0.989	0.87	0.031	0.484	0.02
Mean filter	56.254	7.5	7.747	30.629	317.909	0.697	0.303	0.995	1.225	0.019	0.49	0.007
Sharpening	99.691	9.985	5.262	28.144	335.697	0.718	0.282	0.984	1.293	0.016	0.492	0.005
Histogram Equalization	302.435	17.391	0.443	23.324	453.363	0.749	0.251	0.97	1.746	0.014	0.493	0.007

The IQM values calculated for each robustness attack carried out on image in image steganography:

Input image (.png)	**MSE**	**RMSE**	**SNR**	**PSNR**	**AD**	**NCC**	**IF**	**SC**	**NAE**	**SSIM**	**DSSIM**	**UIQI**
Embedded image	9.377	3.062	14.519	38.41	49.463	0.424	0.576	0.621	0.172	0.031	0.484	0.01
After Cropping	9.647	3.106	14.66	38.287	60.299	0.431	0.569	0.632	0.187	0.012	0.494	0.005
After Rotating	303.803	17.43	-0.586	23.305	379.497	0.411	0.589	0.621	1.319	0.026	0.487	0.005
After adding Gaussian Noise	280.704	16.754	-0.243	23.648	361.112	0.875	0.125	1.15	1.255	0.025	0.487	0.006
After Adding Poison Noice	39756.296	199.39	-21.755	2.137	-280.711	16489987.627	-16489986.627	606.525	0.978	1352522.802	-676260.901	1356041.299
After adding Speckle Noice	89.845	9.479	4.705	28.596	139.229	0.338	0.662	0.509	0.484	0.009	0.495	0.003
After adding Salt &Pepper Noise	19.361	4.4	11.37	35.262	59.882	0.427	0.573	0.598	0.208	0.017	0.491	0.011
Median filter	63.947	7.997	6.181	30.073	156.53	0.43	0.57	0.627	0.544	0.009	0.495	0.002
Mean filter	95.277	9.761	4.45	28.341	203.007	0.791	0.209	1.025	0.706	0.014	0.493	0.007
Sharpening	130.861	11.439	3.071	26.963	240.987	0.404	0.596	0.583	0.838	0.015	0.492	0.009
Histogram Equalization	252.826	15.901	0.211	24.103	457.144	0.757	0.243	1.085	1.589	0.009	0.495	0.004

5. Conclusion

In this project we make use of the LSB steganography in two separate ways for embedding the text file and the image file respectively in the cover image. One algorithm makes use of the XOR operation in order to embed the secret text while the other method has simply embedded the RGB values of the secret image into the cover image by following the required steps of LSB.

After analysing the embedded images by carrying out several robustness attacks, we find out that for several image quality measurements (IQM), the values of the embedded image and the attacked images are similar, but for some measurements (like MSE in many cases) the results show huge discrepancies from the attacked images IQM values. We realise that LSB steganography algorithm has the potential to withstand a fair number of attacks as could be seen by the comparison done of the IQM results between the embedded and attacked images, but an improvement in the algorithm still becomes a need in order to provide better security to our information hidden in these image files.

Additionally, any image file format, including .jpeg and .bmp can be used as an input in the algorithm, thus extending it to multiple file formats. Here we have taken an example solely of a .png file format to understand the steganography algorithm for embedding both text and image files on a cover image.

The proposed project theme is useful for managing private messages within a non-confidential image. This algorithm can also be used in areas where high security is required, where it is difficult to extract the encrypted file. Areas where the LSB steganography method can be used are, Banks, Hospitals, Police Department, etc.

References

[1] V. Lokeswara Reddy, Dr. A. Subramanyam, Dr. P. Chenna Reddy, 2011, “Implementation of LSB Steganography and its Evaluation for various File Formats”, Int J. Advanced Networking and Applications Volume: 02, Issue: 05, Pages: 868-872

[2] Nan-I Wu, Min-Shiang Hwang, 2017, “A Novel LSB Data Hiding Scheme with the Lowest Distortion”, Imaging Science Journal, Vol.65, Issue: 6, Pages: 371-378.8p.

[3] Firas A. Jassim, Hind E. Qassim, 2013, “Five Modulus Method for Image Compression”, Signal & Image Processing: An International Journal (SIPIJ), Vol.3, No. 5.

[4] Sudhir Batra, Rahul Rishi and Rajkumar, 2010, “Insertion of message in 6th, 7th and 8th bit of pixel values and its retrieval in case intruder changes the least significant bit of image pixels”, International Journal of Security and Its Applications Vol. 4, No. 3.

[5] Touhid Bhuiyan, Afjal H. Sarower, Md. Rashed Karim, Md. Maruf Hassan, 2019, “An Image Steganography Algorithm using LSB Replacement through XOR Substitution”, 2019 International Conference on Information and Communications Technology (ICOIACT), 44-49.

[6] Jessica Fridrich, Miroslav Goljan, Rui Du, 2001, “Reliable detection of LSB steganography in color and grayscale images”, Multimedia and security: new challenges, 27-30.

[7] Nadeem Akhtar, Pragati Johri, Shahbaaz Khan, 2013, “Enhancing the security and quality of LSB based image steganography”, International Conference and Computational Intelligence and Communication Networks, 385-390.

[8] Shamim Ahmed Laksar, Kattamanchi Hemachandran, 2012, “Hiding Capacity data hiding using LSB Steganography and Encryption”, International Journal of Database Management Systems 4(6), 57.

[9] Mamta Juneja, Parvinder S Sandhu, Ekta Walia, 2009, “Application of LSB based steganographic technique for 8-bit color images”, World Academy of Science, Engineering and Technology 50, 423-425.

[10] Mohammed Mahdi Hashim, Mohd. Shafry Rahim, Fadil Abass Johi, Mustafa Sabah Taha, Hassan Salman Hamad, 2018, “Performance evaluation measurement of image steganography techniques with analysis of LSB

based on variation images formats”, International Journal of Engineering & Technology, 7(4) (2018) 3505-3514.

[11] Mahbuba Begum, Mohammad Shorif Uddin, 2020, “Analysis of Digital Image Watermarking Techniques through Hybrid Methods”, Advances in Multimedia, Volume 2020, Article ID 7912690, 12 pages.