International Journal of Innovative Research in Computer Science & Technology (IJIRCST)
ISSN (online): 2347-5552, Volume-10, Issue-5, September 2023
https://doi.org/10.55524/ijircst.2023.11.5.1
Article IDIRP1401, Pages 1-9
www.ijircst.org

A Password and Least Significant Bit Substitution Based Steganography Technique for Image Hiding

Diotima Nag[1], Unmesh Mandal[2]*, Goutam Das[3], and Rimpi Saha[4]

[1]PG Scholar, Department of Computer Science, University of Calcutta, Technology Campus, JD-2, Sector –III, SALT LAKE, Kolkata, West Bengal, India

[2]Assistant Professor, Department of Computer Science, Bidhan Chandra College, Rishra, 31, G. T. Road (East), Rishra, Hooghly, West Bengal, India

[3]SACT-I, Department of Computer Science, Bidhan Chandra College, Rishra, 31, G. T. Road (East), Rishra, Hooghly, West Bengal, India

[4]SACT-II, Department of Computer Science, Bidhan Chandra College, Rishra, 31, G. T. Road (East), Rishra, Hooghly, West Bengal, India

Correspondence should be addressed to Unmesh Mandal; unmesh.mandal@gmail.com

ABSTRACT- The proposed work introduces an image steganography technique using text-based password that acts as the symmetric key of encryption and decryption. It contains two components, an encoder, and a decoder. An encoder takes a secret image that needs to be hidden inside another cover image. The Combined image is encrypted with an eight characters' password. The starting position of the cover image from where the secret image starts to hide depends on the password. Then the (Least Significant Bit) LSB Substitution technique is used to hide the entire secret image inside the cover image. The decoder takes the encrypted image and the password to retrieve the secret image back. The proposed work produces an encrypted image by keeping 99% information of the cover image and 100% information of the secret image, thus it provides lossless retrieval of the secret image. The proposed work includes dynamicity of position in steganography that ensures more security of image data over the network than the existing state of art methods.

KEYWORDS- Cryptography, Steganography, Least Significant Bit Substitution, Image Hiding.

I. INTRODUCTION

In the field of network security, cryptography and steganography are two main techniques that are generally used in two different situations. Whenever it is necessary that the meaning of the secret message must not become understandable or interpretable by others we use cryptography, but if we try to hide the existence of the secret image from the public in the way of transmitting then we use steganography.[1] Steganography can be defined as a technique that helps to hide data behind another media so that any changes in the original media cannot be detectable by any eavesdropper.[2] The original media file can be anything like text, image, audio, or video files and also the secret message can be any kind of multimedia file. The original media file in which we want to hide our secret message is called the cover file and the message that needs to be hidden from the outside world is termed the secret file. [3] Types of steganography exist according to the types of cover files as text steganography uses a text cover file, image steganography uses an image cover file, and audio steganography uses any type of audio file as a cover file, etc. [4] There exist a number of algorithms in the field of image steganography. The most famous algorithm is the Least Significant Bit (LSB) substitution. In this technique, every bit of the secret message is populated to the Least Significant Bit of the cover file [5]. But one problem with this technique is that once an eavesdropper becomes successful to intercept the embedded file it is easy to trace out the whole secret message [6]. There are other techniques that proved to be useful in image steganography. Emam et al.[7] proposed a steganography method where the secret message is hidden in the random location of the cover image using a Pseudo Random Number Generator.[7] However, some Pseudo Random Number Generators may suffer from improper implementation and also issues with backdoors.[8] Swain proposed a steganography method that divides the image into non-overlapping blocks and used the LSB substitution technique and pixel value differencing technique.[9] All these techniques mainly focused on the cover and the secret image and manipulated them. However, our proposed technique mainly focuses on the password that will be a user input and depending on the nature of the password the steganography process will be done. The proposed work finally results very less degradation of cover image with a more secure technique of steganography.

II. METHODOLOGY

A. *Password to Key Mapping*

To hide a color image (secret image) into another color image (cover image), at first, we have to decide the location of the cover image from where the hiding of the secret image gets started. This is done with the help of a password of 8 characters. As well as here we use the same password of 8 characters for recovering the hidden image. For this, at first, we convert the characters of the password into their corresponding ASCII values, and then the ASCII values of the characters of the password are converted into 8-bit binary numbers (show in figure 1).

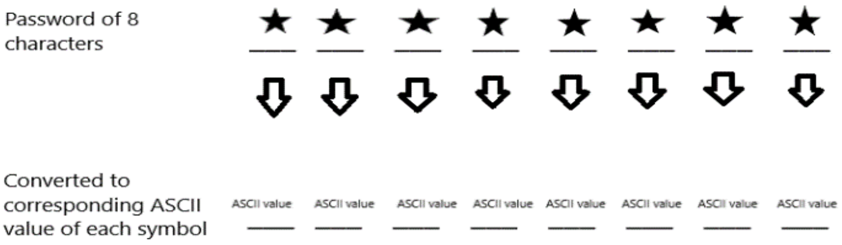


Figure 1: Mapping of Characters to Ascii Value in a Password

An 8-bit binary representation of each character of a password is used to construct a binary matrix by arranging the binary value of a character of the password in a row and the next row of the matrix is the binary value of the next character of the password and so on. In this way, we construct a binary matrix of order 8 X 8(show in figure 2).

After that, we take the main diagonal (1st diagonal) which is from the top left to bottom right, and we take the anti-diagonal (2nd diagonal) from top right to bottom left. The decimal values equivalent to the binary numbers contained in the 1st and 2nd diagonals are considered as the starting row number and column number of the cover image respectively from where the secret image starts to hide.

B. *Steganography Process*

After finding the starting position of hiding the secret image in the cover image, split each pixel of the secret image and cover image into R, G, and B colors. Represent each of the values of R, G, and B color into a binary number. Here each LSB position of a pixel value of the cover image is substituted by the pixel value of the secret image. For substitution R, G, and B color value of the cover image is substituted by the R, G, and B color value of the secret image. So, 8-bit of a pixel of the secret image is substituted to the LSB position of 8 consecutive pixels of the cover image. The following figure 4. illustrates this procedure.

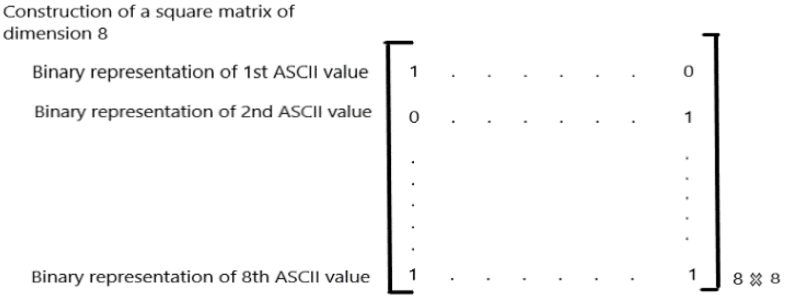

Construction of a square matrix of dimension 8
Binary representation of 1st ASCII value
Binary representation of 2nd ASCII value
Binary representation of 8th ASCII value
1 0
0 1
1 1
8 ✕ 8

Figure 1: Formation of 8 × 8 binary matrix

C. *Difference Image*

The Difference image is calculated for the demonstration of the quality of the proposed encryption process. It represents the closeness of the Cover image and the Encrypted image. Let the cover image is c and the combined image or encrypted image is c'. Then the Difference image d can be defined as,

dijk = {cijk- cijk': i ∈ [0, w), j ∈ [0, h) and k ∈ {R, G, B}} Where w is the width of the cover image and h is the height of the cover image.

Each pixel of the Difference image consists of subtraction of pixel values at ith row and jth column of the corresponding cover and combined image where 0≤i<w and 0≤j<h of k channel.

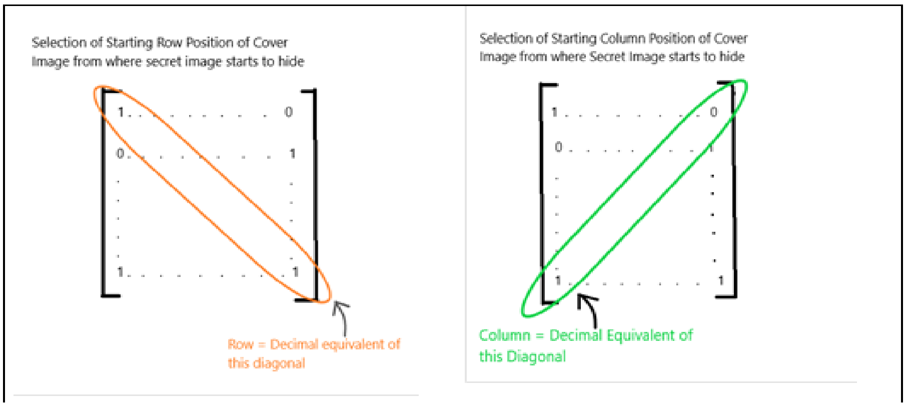

Selection of Starting Row Position of Cover Image from where secret image starts to hide
1 0
0 1
1 1
Row = Decimal equivalent of this diagonal
Selection of Starting Column Position of Cover Image from where Secret Image starts to hide
1 0
0 1
1 1
Column = Decimal Equivalent of this Diagonal

Figure 3: Formation of Key

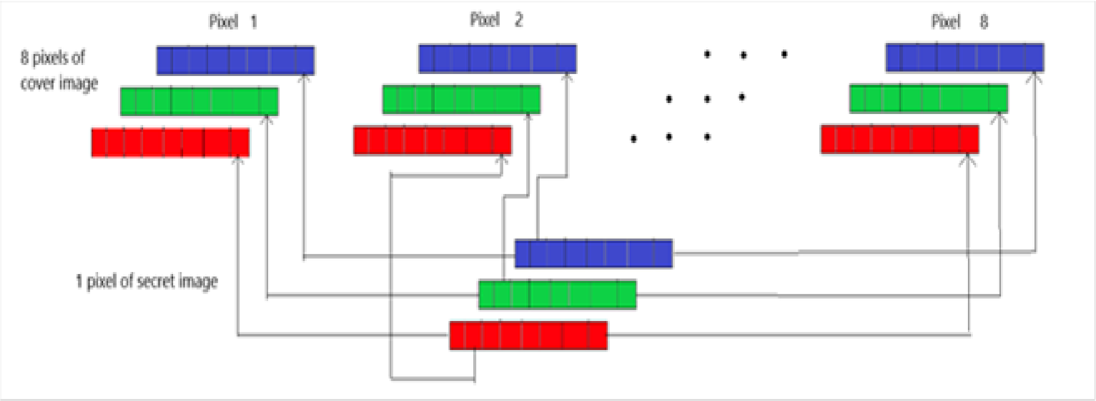
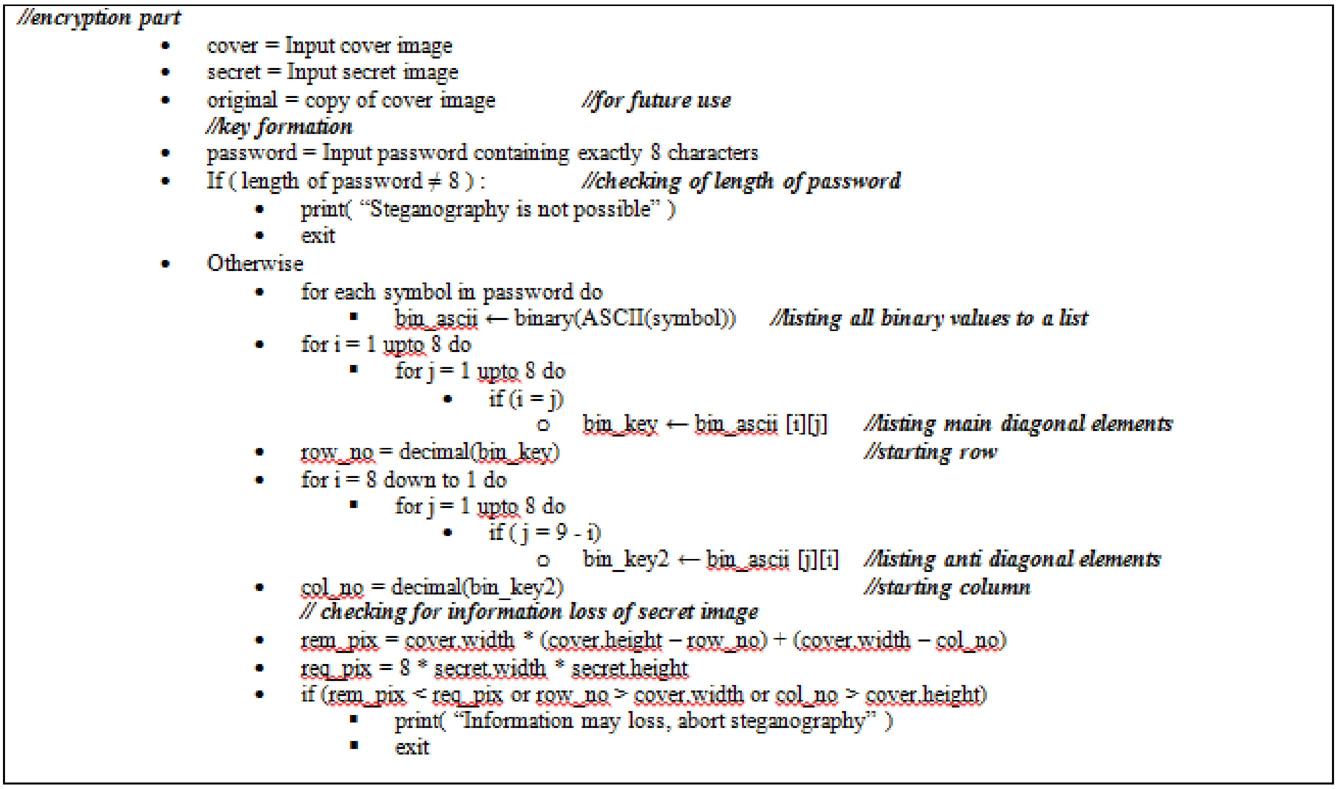
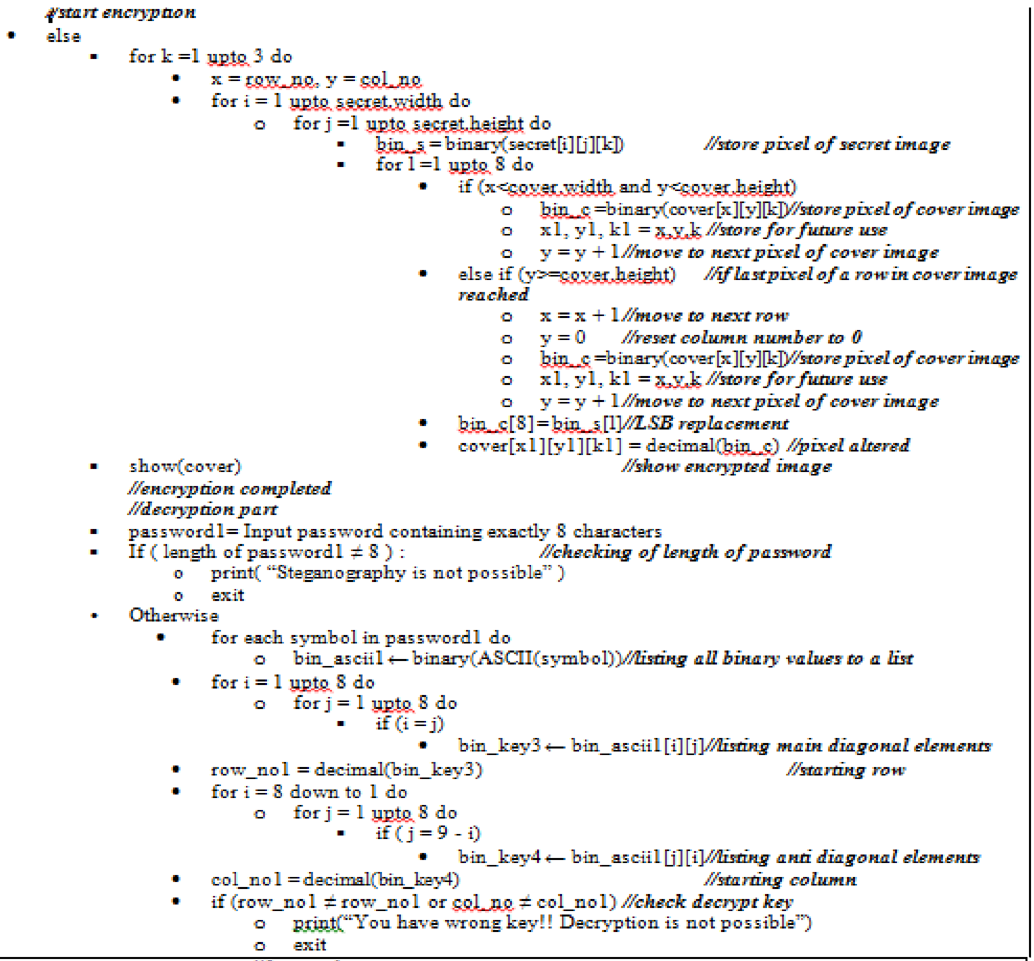


Figure 4: Illustration of the steganography proces

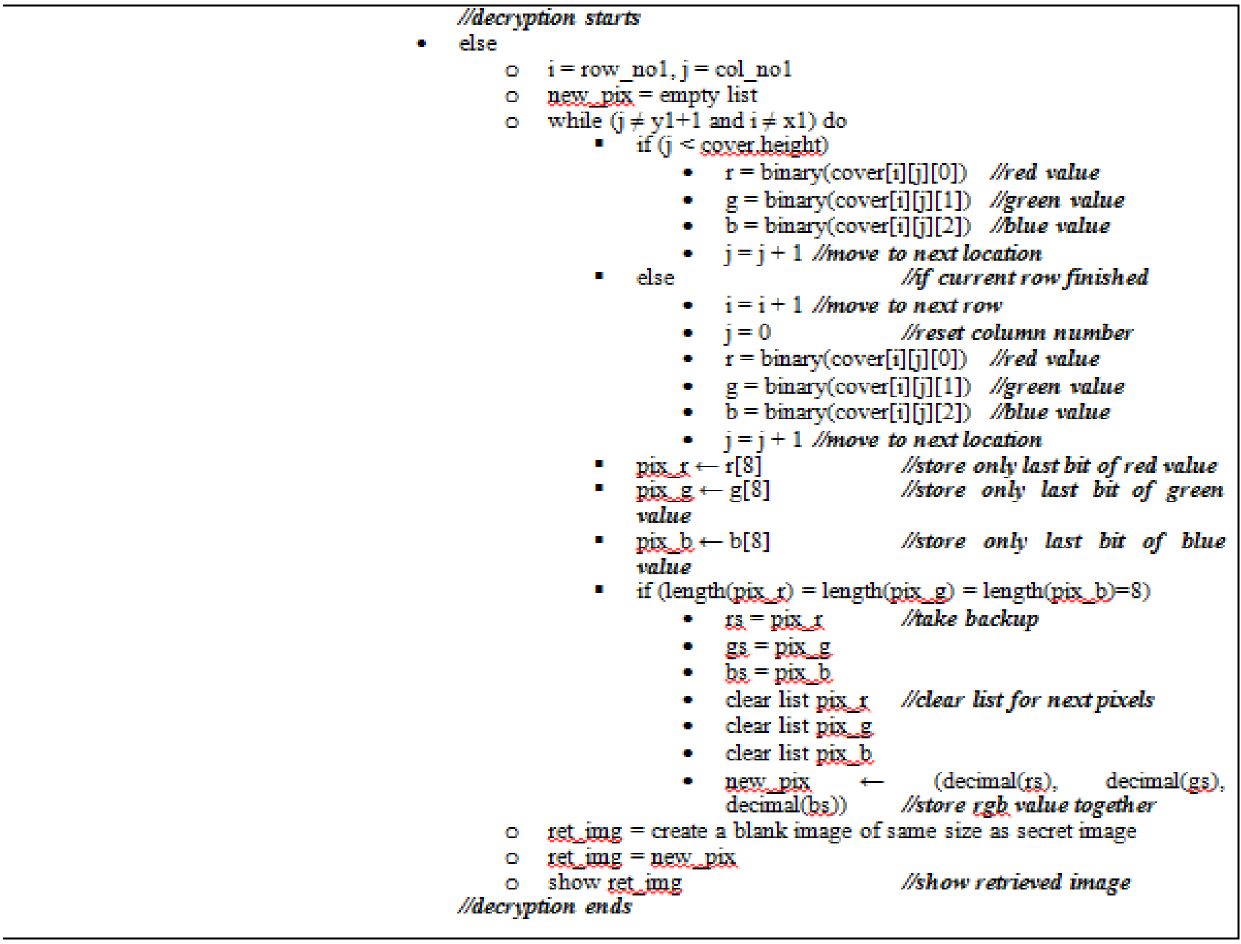
D. Pseudocode of The Proposed Method



```
//encryption part
    •  cover = Input cover image
    •  secret = Input secret image
    •  original = copy of cover image          //for future use
       //key formation
    •  password = Input password containing exactly 8 characters
    •  If ( length of password ≠ 8 ) :          //checking of length of password
         •  print( "Steganography is not possible" )
         •  exit
    •  Otherwise
         •  for each symbol in password do
              ▪  bin_ascii ← binary(ASCII(symbol))     //listing all binary values to a list
         •  for i = 1 upto 8 do
              ▪  for j = 1 upto 8 do
                   •  if (i = j)
                        o  bin_key ← bin_ascii [i][j]        //listing main diagonal elements
         •  row_no = decimal(bin_key)                       //starting row
         •  for i = 8 down to 1 do
              ▪  for j = 1 upto 8 do
                   •  if ( j = 9 - i)
                        o  bin_key2 ← bin_ascii [j][i]       //listing anti diagonal elements
         •  col_no = decimal(bin_key2)                      //starting column
            // checking for information loss of secret image
         •  rem_pix = cover.width * (cover.height – row_no) + (cover.width – col_no)
         •  req_pix = 8 * secret.width * secret.height
         •  if (rem_pix < req_pix or row_no > cover.width or col_no > cover.height)
              ▪  print( "Information may loss, abort steganography" )
              ▪  exit
```



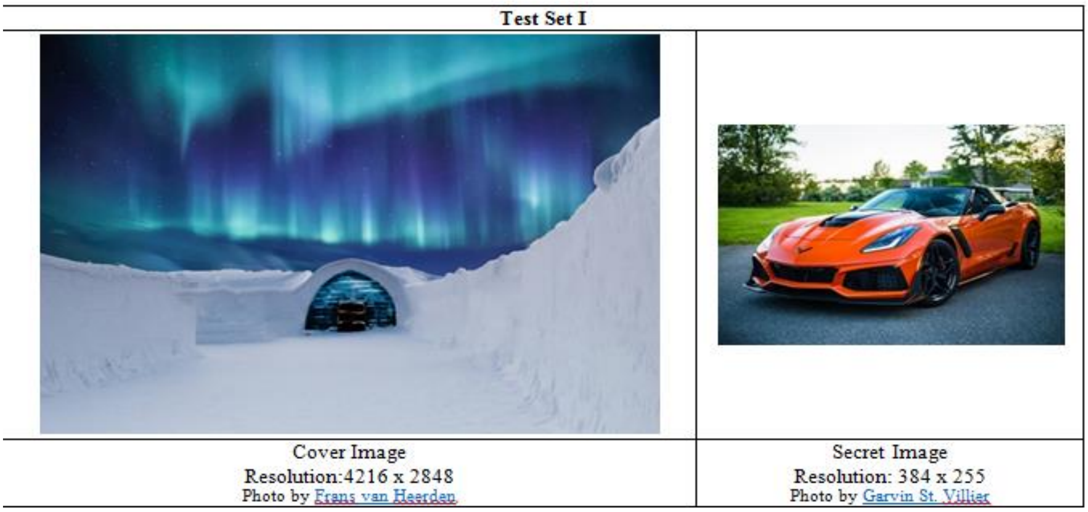
```
    #start encryption
•   else
        ▪   for k =l upto 3 do
                •   x = row_no, y = col_no
                •   for i = l upto secret.width do
                        o   for j =l upto secret.height do
                                ▪   bin_s = binary(secret[i][j][k])          //store pixel of secret image
                                ▪   for l =l upto 8 do
                                        •   if (x<cover.width and y<cover.height)
                                                o   bin_c =binary(cover[x][y][k])//store pixel of cover image
                                                o   x1, y1, k1 = x,y,k //store for future use
                                                o   y = y + 1//move to next pixel of cover image
                                        •   else if (y>=cover.height)     //if last pixel of a row in cover image
                                            reached
                                                o   x = x + 1//move to next row
                                                o   y = 0      //reset column number to 0
                                                o   bin_c =binary(cover[x][y][k])//store pixel of cover image
                                                o   x1, y1, k1 = x,y,k //store for future use
                                                o   y = y + 1//move to next pixel of cover image
                                        •   bin_c[8]=bin_s[l]//LSB replacement
                                        •   cover[x1][y1][k1] = decimal(bin_c) //pixel altered
        ▪   show(cover)                                              //show encrypted image
            //encryption completed
            //decryption part
        ▪   password1= Input password containing exactly 8 characters
        ▪   If ( length of password1 ≠ 8 ) :                    //checking of length of password
                o   print( "Steganography is not possible" )
                o   exit
        •   Otherwise
              •     for each symbol in password1 do
                        o   bin_ascii1 ← binary(ASCII(symbol))//listing all binary values to a list
                •   for i = 1 upto 8 do
                        o   for j = 1 upto 8 do
                                ▪   if (i = j)
                                        •   bin_key3 ← bin_ascii1[i][j]//listing main diagonal elements
                •   row_no1 = decimal(bin_key3)                                        //starting row
                •   for i = 8 down to 1 do
                        o   for j = 1 upto 8 do
                                ▪   if ( j = 9 - i)
                                        •   bin_key4 ← bin_ascii1[j][i]//listing anti diagonal elements
                •   col_no1 = decimal(bin_key4)                              //starting column
                •   if (row_no1 ≠ row_no1 or col_no ≠ col_no1) //check decrypt key
                        o   print("You have wrong key!! Decryption is not possible")
                        o   exit
```



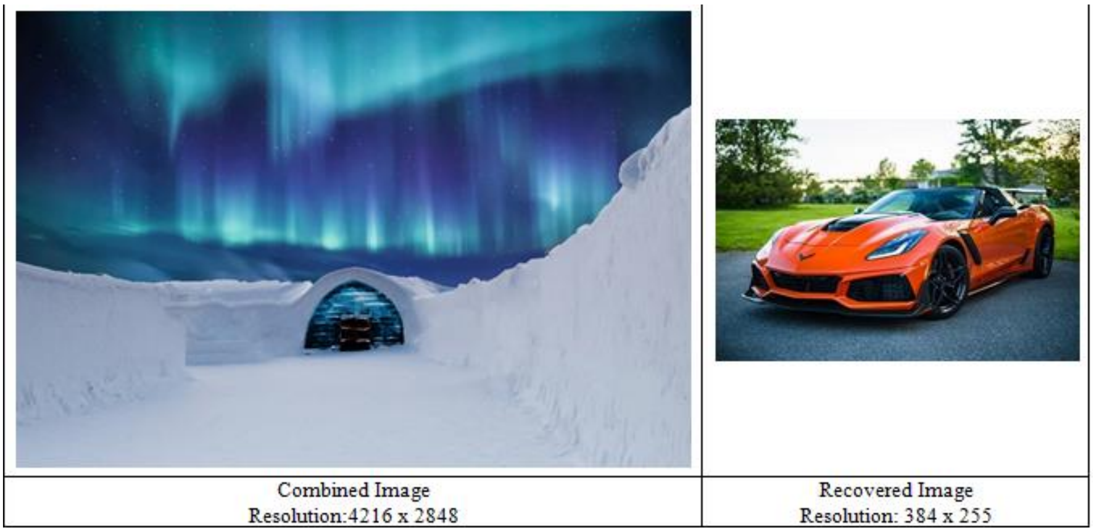
```
            //decryption starts
        •   else
                o   i = row_no1, j = col_no1
                o   new_pix = empty list
                o   while (j ≠ y1+1 and i ≠ x1) do
                        ▪   if (j < cover.height)
                                •   r = binary(cover[i][j][0])   //red value
                                •   g = binary(cover[i][j][1])   //green value
                                •   b = binary(cover[i][j][2])   //blue value
                                •   j = j + 1 //move to next location
                        ▪   else                             //if current row finished
                                •   i = i + 1 //move to next row
                                •   j = 0                    //reset column number
                                •   r = binary(cover[i][j][0])   //red value
                                •   g = binary(cover[i][j][1])   //green value
                                •   b = binary(cover[i][j][2])   //blue value
                                •   j = j + 1 //move to next location
                        ▪   pix_r ← r[8]                     //store only last bit of red value
                        ▪   pix_g ← g[8]                     //store only last bit of green
                            value
                        ▪   pix_b ← b[8]                     //store only last bit of blue
                            value
                        ▪   if (length(pix_r) = length(pix_g) = length(pix_b)=8)
                                •   rs = pix_r               //take backup
                                •   gs = pix_g
                                •   bs = pix_b
                                •   clear list pix_r         //clear list for next pixels
                                •   clear list pix_g
                                •   clear list pix_b
                                •   new_pix    ←    (decimal(rs),    decimal(gs),
                                    decimal(bs))             //store rgb value together
                o   ret_img = create a blank image of same size as secret image
                o   ret_img = new_pix
                o   show ret_img                     //show retrieved image
            //decryption ends
```

III. EXPERIMENTAL RESULTS

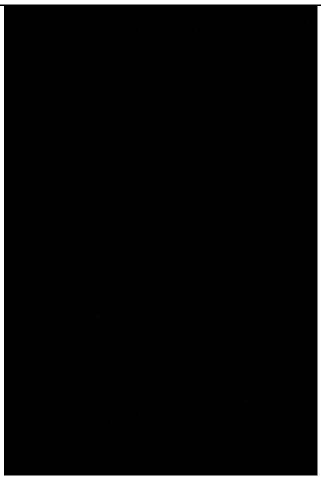
The proposed method is tested on multiple Cover and Secret images with different set of Passwords. For each set samples, difference images and histograms are calculated to justify the quality of the proposed encryption technique



Test Set I	
Cover Image Resolution:4216 x 2848 Photo by Frans van Heerden	Secret Image Resolution: 384 x 255 Photo by Garvin St. Villier

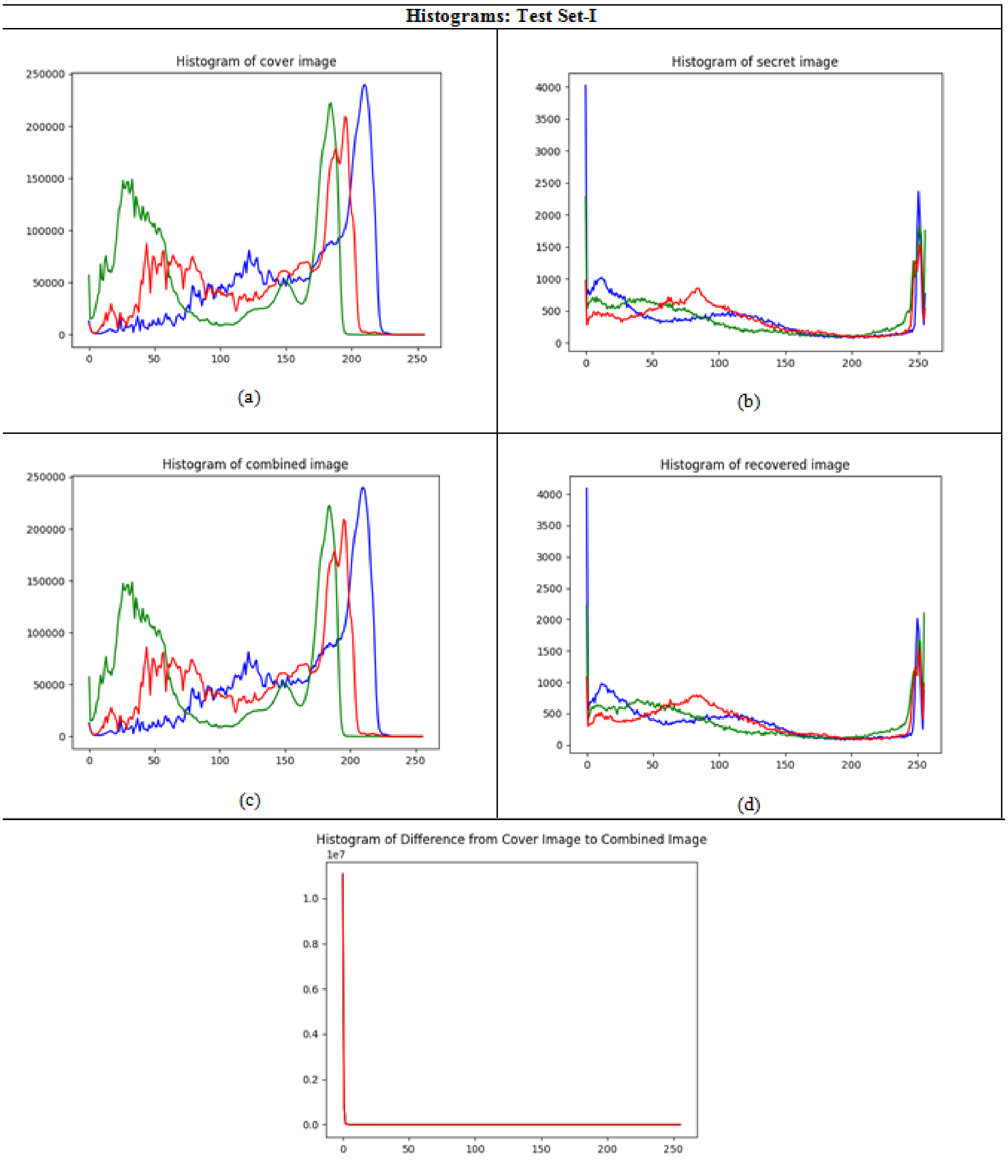


Combined Image Resolution:4216 x 2848	Recovered Image Resolution: 384 x 255



Difference Image
Resolution:3456 x 5184

Histograms: Test Set-I



(a)

(b)

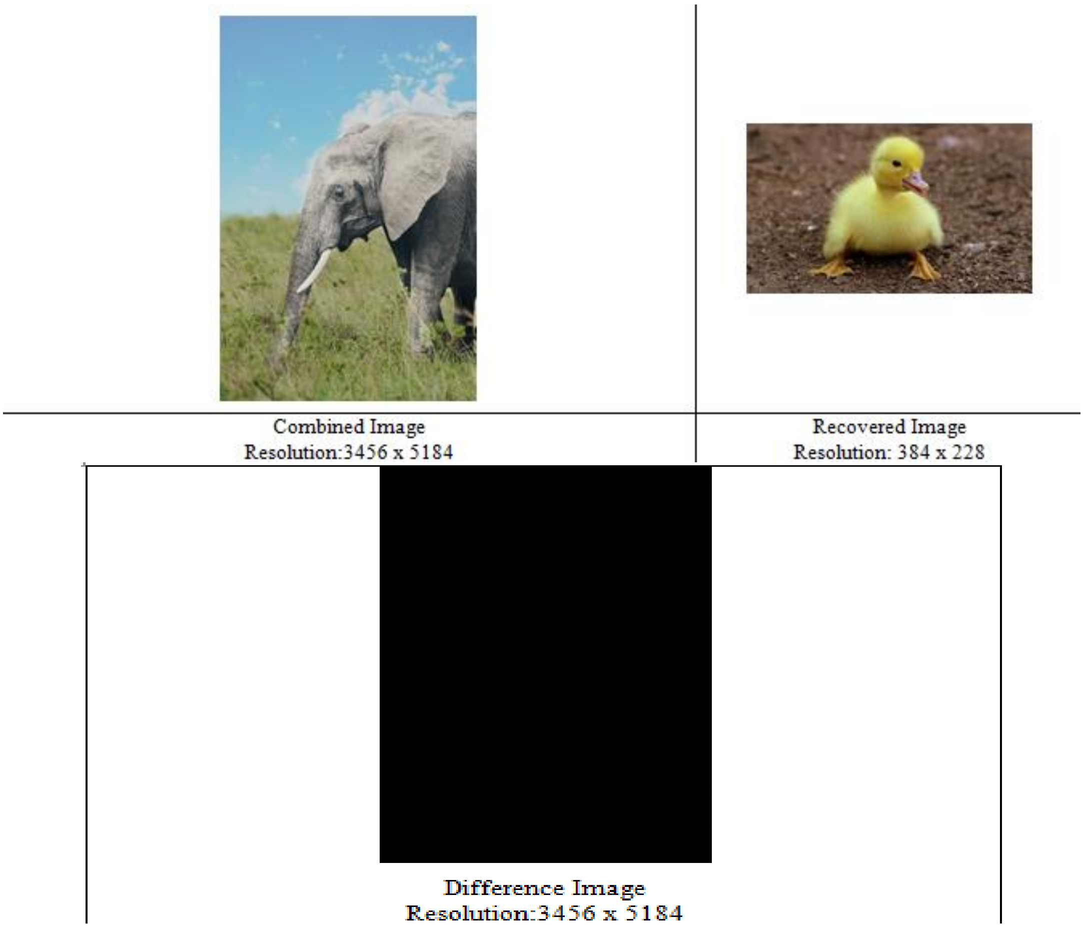
(c)

(d)

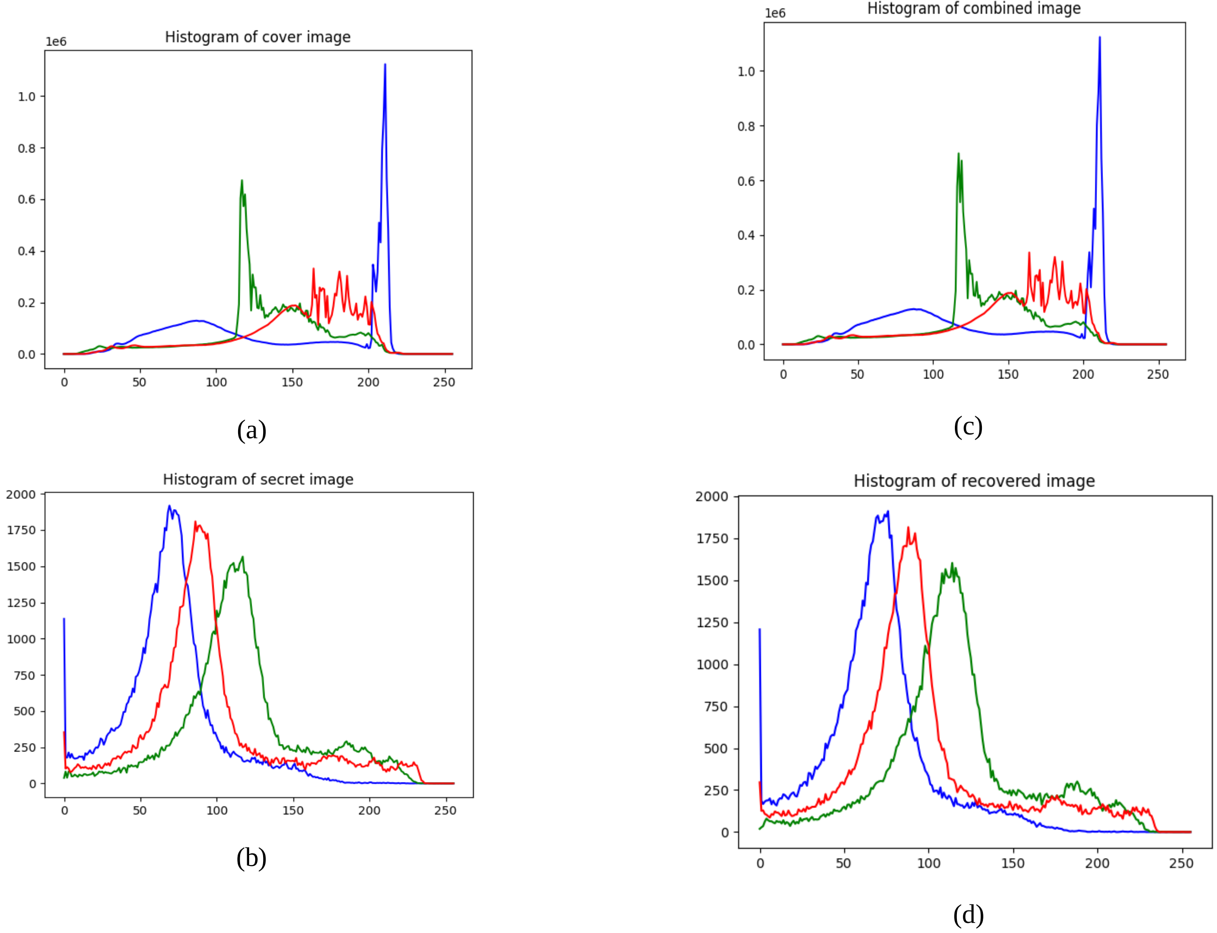
(e)

RGB Histogram of the a. Cover Image, b. Secret Image, c. Combined Image, d. Recovered Image, e. Difference Image

Test Set II	
Cover Image Resolution: 3456 x 5184 Photo by Saifuddin Ratlamwala	Secret Image Resolution: 384 x 228 Photo by Pixabay



Histograms: Test Set-II

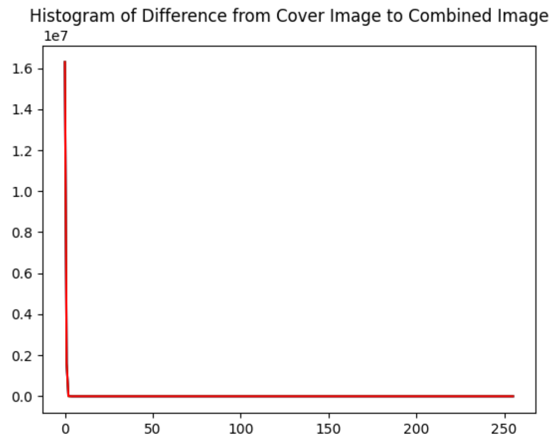


(a)

(c)

(b)

(d)



(e)

Figure 5: RGB Histogram of the a. Cover Image, b. Secret Image, c. Combined Image, d. Recovered Image, e. Difference Image

IV. DISCUSSIONS

From each of the output set it can easily be inferred that the degradation of original cover file is very less as the histogram of each degraded image shows that pixel intensity of all the pixels in all three channels remains near about zero. That is the actual beauty of LSB substitution technique. Also, from the stego image it is also very much difficult from where the encryption of the secret image has started. The role of the password here is very crucial as the encryption technique completely depend upon the password. It becomes very much difficult to trace out the decimal equivalent of the main and the anti-diagonal elements of characters of the password. Thus, our proposed work provides much more security to the secret message and it also ensures encryption with a very little degradation in cover media file as here we use LSB substitution technique.

V. CONCLUSION

Here we show a new method to hide a secret image into a cover image using a manual password that decides the location from where the secret image gets started to hide. The use of LSB substitution technique makes this method more effective as it ensures less quality degradation in cover media file. In future, we can use this kind of methodology in almost every kind of steganography techniques like text steganography, audio steganography, video steganography etc. Also we can try techniques other than LSB substitution so that more difficulty arises at the time of decryption and it may become much more secure method.

CONFLICTS OF INTEREST

The authors declare that they have no conflicts of interest.

ACKNOWLEDGMENT

We are grateful to the reviewers and facilitators whose constructive comments were useful in improving the content on this document. This work was supported by Bidhan Chandra College, Rishra by providing the platforms and means.

REFERENCES

[1] N. F. Johnson and S. Jajodia, "Exploring steganography: Seeing the unseen," in Computer, vol. 31, no. 2, pp. 26-34, Feb. 1998, https://doi.org/10.1109/MC.1998.4655281

[2] Nashat, D., Mamdouh, L. An efficient steganographic technique for hiding data. J Egypt Math Soc 27, 57 (2019). https://doi.org/10.1186/s42787-019-0061-6

[3] S. N. Kishor, G. N. K. Ramaiah and S. A. K. Jilani, "A review on steganography through multimedia," 2016 International Conference on Research Advances in Integrated Navigation Systems (RAINS), Bangalore, India, 2016, pp. 1-6, doi: https://doi.org/10.1109/RAINS.2016.7764373

[4] F. A. P. Petitcolas, R. J. Anderson and M. G. Kuhn, "Information hiding-a survey," in Proceedings of the IEEE, vol. 87, no. 7, pp. 1062-1078, July 1999, doi:https://doi.org/10.1109/5.771065

[5] Abbas Cheddad, Joan Condell, Kevin Curran, Paul Mc Kevitt, Digital image steganography: Survey and analysis of current methods, Signal Processing, Volume 90, Issue 3, 2010, Pages 727-752, ISSN 0165-1684, https://doi.org/10.1016/j.sigpro.2009.08.010

[6] X. Yu and N. Babaguchi, "An Improved Steganalysis Method of LSB Matching," 2008 International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Harbin, China, 2008, pp. 557-560, doi: https://doi.org/10.1109/IIH-MSP.2008.344

[7] Emam, Marwa & Ali, Abdelmgeid & A., Fatma. (2016). An Improved Image Steganography Method Based on LSB Technique with Random Pixel Selection. International Journal of Advanced Computer Science and Applications. 7. http://dx.doi.org/10.14569/IJACSA.2016.070350

[8] Rafik Hamza, A novel pseudo random sequence generator for image-cryptographic applications, Journal of Information Security and Applications, Volume 35, 2017, Pages 119-127, ISSN 2214-2126, https://doi.org/10.1016/j.jisa.2017.06.005

[9] Gandharba Swain, A Steganographic Method Combining LSB Substitution and PVD in a Block, Procedia Computer Science, Volume 85, 2016, Pages 39-44, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2016.05.174

ABOUT THE AUTHORS

Diotima Nag is a graduate student currently pursuing Masters of Science degree in Computer Science at University of Calcutta. She has completed her BSc degree in Computer Science at Bidhan Chandra College, Rishra, in 2021. She received several scholarships in recognition of her academic behaviour to her studies. Her primary area of interest lies in Artificial intelligence, machine learning, Deep learning, Recommendation system.

Unmesh Mandal is an Assistant Professor at the Department of Computer Science, Bidhan Chandra College, Rishra, Hooghly, West Bengal, India. He received M.Sc. degree in Computer and Information Science from University of Calcutta in 2014 and M.Tech. degree in Computer Science and Engineering from the same university in 2016. He has teaching experience of more than 6 years. His research interest includes

Artificial Intelligence, Machine Learning, Deep Learning, Time Series Analysis and Forecasting.

Goutam Das is a State Aided College Teacher (SACT-I) at the Department of Computer Science, Bidhan Chandra College, Rishra, Hooghly, West Bengal, India. He received M.Sc. degree in Computer and Information Science from University of Calcutta in 2013 and M.Tech. degree in Computer Science and Engineering from the same university in 2015. He has teaching experience of more than 5 years. His research interest includes Image Processing, Artificial Intelligence, Machine Learning and Deep Learning.

Rimpi Saha is a State Aided College Teacher (SACT-II) at the Department of Computer Science, Bidhan Chandra College, Rishra, Hooghly, West Bengal, India. She received M.Sc. degree in Computer Science from University of Burdwan in 2017 as a Gold Medallist. She has teaching experience of more than 5 years.