Indonesian Journal of Electrical Engineering and Computer Science
Vol. 14, No. 3, June 2019, pp. 1412~1419
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v13.i3.pp1412-1419

A new encrypted method in image steganography

Sabyasachi Pramanik[1], Dr. R. P. Singh[2], Ramkrishna Ghosh[3]
[1]Department of Computer Science & Engineering, Haldia Institute of Technology, India
[2]Department. of Computer Science & Engineering, Sri Satya Sai University of Technology & Medical Sciences, India
[3]Department of Information Technology, Haldia Institute of Technology, India

Article Info

Article history:

Received Oct 19, 2018
Revised Nov 17, 2018
Accepted Feb 26, 2019

Keywords:

ASCII
CAPTCHA
Cover image
Stego image

ABSTRACT

Steganography is data hiding technique in internet. Here we send CAPTCHA codes within a cover image using image steganography. CAPTCHA are the crazy codes. They are used in human response test. The word is actually acronym for: "Completely Automated Public Turning test to tell Computers and Humans Apart". It is a type of challenge-response test used in computing to determine whether or not the user is human. Websites implement CAPTCHA codes into their registration process due to spam. This is the utility of CAPTCHA codes. Here we generate CAPTCHA codes are and later send them in an encrypted version. Actually CAPTCHA codes are embedded ASCII into cover image with an encrypted form resulting stego image and thus attackers can not fetch the actual CAPTCHA resulting in a secured transmission of confidential data via internet using image steganography

Corresponding Author:

Sabyasachi Pramanik,
Department of Computer Science & Engineering,
Haldia Institute of Technology,
ICARE Complex, HIT Campus, Dist Midnapore (E), PIN-721657, India.
Email: sabyalnt@gmail.com

1. INTRODUCTION

1.1. Background

Here CAPTCHA codes have been formed by at least one small letter, one capital letter, one number and one special character and it is minimum of 8 characters and maximum of 16 characters. CAPTCHA are used to check that the person registering or trying to comment is a real live human being as opposed to a computer program attempting to spam the site.

1.2. The Problem

If CAPTCHA codes are embedded directly using LSB [1] embedding technique then it is easy for the attackers to extract the CAPTCHA. So, a new technique has been implemented to change the actual CAPTCHA into an encrypted version and then that encrypted version has been embedded into cover image [2] resulting in a stego image [3].

1.3. The Proposed Solution

We embed CAPTCHA into a 24 bit color image [4] as our cover image. We use LSB of Red, Green and Blue pixels [5] of 24 bit cover image in image steganography [6]. Embedding technique is based on an innovative algorithm that counts the frequency of 0's and 1's. We give priority to the lesser number of occurrences of 0's or 1's present. At first, we store the flag's value. If lesser number of 0's is present then flag's value will be 0, otherwise 1. Then we embed how many lesser number of 0's or 1's is present. We decide using this value total number of pixels needed to store the bit positions of 0's or 1's. Thus an ASCII [7] value is not directly embedded into the LSB of cover image, rather flag's value, number of occurrence of lesser 0's or 1's, positions where lesser number of 0's or 1's are taken place would be

embedded which makes attacker impossible to retrieve the actual data [8] which is the ASCII value of the embedded character. A layer of encryption [9] has been taken place here. At the receiver's end at first we extract flag's value. Based on the flag's [10] value we get the lesser occurrences of 0's or 1's. In this technique [11] then how many 0's or 1's are present would be extracted. After this the positions where lesser number of 0's or 1's are present that information would be extracted. This is the process to extract the whole information [12] about the embedded ASCII. Thus we gather whole information to retrieve the original ASCII that was sent via CAPTCHA through communication channel in internet

2. THE PROPOSED METHOD

2.1. Generation of CAPTCHA

a) Generate a random number [13] between 8 and 16. Let it be n. (We consider that our CAPTCHA would be minimum of 8 characters and maximum of 16 characters)
b) Initialize a string str as str=" "
c) Perform following task for n times
d) Generate random numbers between 97 and 122 (ASCII of a to z) and place it into the variable char.
e) Concatenate, str=str+char
f) Generate random number between 48 and 57 (ASCII of 0 to 9) and place it into the variable char
g) Concatenate, str=str+char
h) Generate random number between 65 and 90 (ASCII of A to Z) and place it into the variable char
i) Concatenate, str=str+char
j) Generate random number between 33 and 47 or 58 and 64 or 91 and 96 or 123 and 126 (ASCII of Special Characters) and place it into variable char.
k) Concatenate, str=str+char
l) Randomly we choose any step from 3 to 7 to generate any kind of char and keep concatenating.
m) Repeat the step 8 until n number of characters has been generated.
n) Terminate the CAPTCHA codes with the ASCII value dot (.) as the end of CAPTCHA code.
o) Jumble up characters of CAPTCHA for a random number of times for repositioning of characters and thus the final CAPTCHA code is ready.

2.2. Embedding [14] Technique

a) As said, we generate a CAPTCHA of minimum 8 characters.
b) CAPTCHA ensures to be generated by combining at least one capital letter, one small letter, one number and one special character.
c) Check the length of the CAPTCHA.
d) Take character one by one and do the following
e) Take the ASCII value of the character
f) Convert the ASCII value into equivalent binary value.
g) Check number of 0's and 1's in this binary value.
h) If the number of 0s is less than number of 1's then flag's value would be 0 otherwise flag's value would be 1.
i) Let the flag value be 0 and then we count how many 0's are available
j) Modify the LSB of red pixel of RGB cover image to store the flag value.
k) Embed the number of 0's into LSB of Green and Blue components of pixel
l) Embed the position of the occurrence of 0's at the LSB of next RGB pixels as per requirements.
m) Repeat the following steps until all the characters get embedded into cover image.

2.3. Extraction [15] Technique

a) Read first red, green and blue pixel of stego image.
b) Extract flag bit based on the LSB of red pixel
c) Extract LSB of green and blue pixel to calculate how many 1's or 0's are present afterward.
d) Read next pixels' LSBs to find the positions of 0's or 1's
e) Based on this value, ASCII can be obtained. Thus relevant character, digit or special character can be found out.
f) Repeat the step 1 to 4 for rest of the pixels until the ASCII value 46 of dot (.) is found which indicates the end of CAPTCHA code. Thus CAPTCHA codes are extracted via a secure steganography.

3. RESEARCH METHOD

To construct CAPTCHA codes we consider the ASCII value is of 7 bits in binary representation.' ASCII range for printable characters is from 0 to 127 and 127 needs a maximum of 7 bits to represent it in binary. Let us assume the generated CAPTCHA is B9@m#~2q. It has a length of 8 characters except dot. Dot must be here the terminating character [16].

First we deal with B. Its ASCII value is 66. Binary [17] equivalent of 66 is 1000010. In 1000010, there are five 0's and two 1's. As the number of 1's is less, so we take the flag as 1. 1's are present in 2^{nd} & 7^{th} bit position. Now we embed flag 1, then the number of 1's i.e., two, then bit [18] positions of 1's, which are 2^{nd} & 7^{th} positions respectively. We embed this value by LSB modification of RGB colour cover image. Thus the ASCII value 66 is encrypted as 6, 2, and 7. 6 come from 110, where 1 is the flag value and 10 is the number of 1's (i.e., two). 2 & 7 come from the position of 1's. Next, 9 is the number that is used to construct the CAPTCHA.

Its ASCII value is 57. Binary equivalent of 57 is 0111001. In 0111001, there are three 0's and four 1's. So, the flag is 0. 0's are present in second, third and seventh bit position. Now we have to embed flag 0, then number of 0's i.e., three, then bit positions of 0's i.e. second, third and seventh values by LSB modification of RGB cover image. Thus the ASCII value 57 is encrypted [19] as 3, 2, 3, 7, where 3 means 011 denoting 0 as flag, 3 as number of 0's. 0's are present in 2^{nd}, 3^{rd} & 7^{th} position, resulting in 2, 3 & 7.

Next character is @. Its ASCII value is 64. Binary equivalent of 64 is 1000000. In 1000000, there are six 0's and one 1. As the number of 1's is less, so we take the flag as 1. 1 is present in 7^{th} bit position. Now we embed flag 1, then the number of 1's i.e., one, then bit positions of 1's, which is 7^{th}. We embed this value by LSB modification of RGB coloured cover image. Thus the ASCII value 64 is encrypted as 5, 7. 5 come from 101, where 1 is the flag value and 01 is the number of 1's. 7 come from the positions of 1.

Next character is m. Its ASCII value is 109. Binary equivalent of 109 is 1101101. In 1101101, there are two 0's and five 1's. As the number of 0's is less, so we take the flag as 0. 0's are present in 2^{nd} & 5^{th} bit position. Now we embed flag 0, then the number of 0's i.e., two, then bit positions of 0's, which are 2^{nd} & 5^{th}. We embed this value by LSB modification of RGB coloured cover image. Thus the ASCII value 109 is encrypted as 2, 2 and 5. 2 come from 010, where 0 is the flag value and 10 is the number of 0's. 2 & 5 come from the positions of 0's.

Next character is #. Its ASCII value is 35. Binary equivalent of 35 is 0100011. In 0100011, there are four 0's and three 1's. As the number of 1's is less, so we take the flag as 1. 1's are present in 1^{st}, 2^{nd} & 6^{th} bit position. Now we embed flag 1, then the number of 1's i.e., three, then bit positions of 1's, which are 1^{st}, 2^{nd} & $6^{th.}$ We embed this value by LSB modification of RGB colour cover image. Thus the ASCII value 35 is encrypted as 7, 1, 2 and 6. 7 come from 111, where 1 is the flag value and 11 is the number of 1's. 1, 2 & 6 come from the positions of 1's.

Next character is ~. Its ASCII value is 126. Binary equivalent of 126 is 1111110. In 1111110, there are one 0 and six 1's. As the number of 0's is less, so we take the flag as 0. 0 is present in 1^{st} bit position. Now we embed flag 0, then the number of 0's i.e., one, then bit positions of 0's, which is at 1^{st} position. We embed this value by LSB modification of RGB colour cover image. Thus the ASCII value 126 is encrypted as 1, 1. 1 comes from 001, where 0 is the flag value and 01 is the number of 0's. 1 comes from the positions of 0's.

Next digit is 2. Its ASCII value is 50. Binary equivalent of 50 is 0110010 (as considered 7bit representation). In 0110010, there are four 0's and three 1's. As the number of 1's is less, so we take the flag as 1. 1 is present in 2^{nd}, 5^{th} and 6^{th} bit positions. Now we embed flag 1, then the number of 1's i.e., three, then bit positions of 1's, which is 2^{nd}, 5^{th} and 6^{th}. We embed this value LSB modification of RGB colour cover image. Thus the ASCII value 50 is encrypted as 7, 2, 5 and 6. 7 come from 111, where 1 is the flag value and 11 is the number of 1's. 2, 5 and 6 come from the positions of 1's.

Next character is q. Its ASCII value is 113. Binary equivalent of 113 is 1110001. In 1110001, there are o 3 0's and four 1's. As the number of 0's is lesser than number of 1's, so we take the flag as 0. 0 is present in 2^{nd}, 3^{rd}, 4^{th} bit positions. Now we embed flag 0, then the number of 0's i.e., three, then bit positions of 0's, which are 2^{nd}, 3^{rd}, 4th. We embed this by LSB modification of RGB colour cover image. Thus the ASCII value 113 is encrypted as 011, 010, 011, 100 i.e. 3, 2, 3, 4. 0 comes from 011, where 0 is the flag value and 11 is the number of 0's. 010, 011, 100 come from the positions of 0's those are 2, 3, 4 respectively.

Finally we embed dot (.). Its ASCII value is 46. Binary equivalent is 0101110. In 0101110, there are three 0's and four 1's. As the number of 0's is less, so we take flag as 0. 0's are present in 1^{st}, 5^{th} and 7^{th} bit positions. Now we embed flag 0, then the number of 0's i.e., three, then bit positions of 0's, which are 1^{st}, 5^{th} and 7^{th}. We embed this value by LSB modification of RGB colour cover image. Thus the ASCII value 46 is encrypted as 3, 1, 5 and 7. Here, 3 come from 011, where 0 is the flag value and 11 is the number of 0's. Then 1, 5 and 7 are the positions of the 0's. ASCII value 46 indicates dot (.). Therefore, it is the end of CAPTCHA code.

Thus the CAPTCHA B9@m#~2q. is embedded in an encrypted form into LSB's of 24 bit RGB cover image as 627323757225712611725632343157

During extraction first pixel's RGB [20] would generate 6 whose binary equivalent is 110 indicating flag's value 1 and total number of 1's are two(10).Then next two pixel's LSB would be retrieved to find three positions of 1's i.e., second and seventh position respectively. Thus instead of ASCII 66, the encrypted value 6, 2, 7 has been retrieved.

6,2,7 indicate a value which has total two 1's at second and seventh position, that means other positions are left as 0's. We know printable ASCII range is up to 127 and 127 is the ASCII value of DEL that cannot be used to create CAPTCHA. So, maximum value is 126 whose binary equivalent is 01111110 that can be used to construct the CAPTCHA. Thus from 6, 2, 7 we can understand flag value is 1, no of 1's are two and 1's are present in 2^{nd} and 7^{th} position. So we get 1000010 and its equivalent ASCII 66 i.e. the ASCII of digit B.

Next pixel's RGB would generate 3 whose binary equivalent is 011 indicating flag's value 0 and total number of 0's are three (11). Then next we retrieve three pixel's LSB to find three positions of 0's i.e. second, third and seventh position respectively. Thus the encrypted value 3, 2, 3, 7 has been retrieved. Thus finally we get 011, 010, 011, 111.

3, 2, 3, 7 indicates a value which has total three 0's at second, third and seventh position that means other positions are left as 1's except MSB which is always 0 here as printable ASCII range is up to 127 and 127 itself is the ASCII value of DEL that cannot be used to create CAPTCHA. So, maximum value is 126 whose binary equivalent is 01111110. Thus finally we get 0111001 from 011,010,011,111 which is equivalent to ASCII 57 i.e., the ASCII of digit 9.

Next pixels' decimal value 5, 7 extracted as 101, 111. RGB would generate decimal 5 whose binary equivalent is 101 indicating flag's value 1 and total number of 1's is one (01). Then we retrieve next pixel's LSBs to find the position of 1's i.e., 7. Thus encrypted value 5, 7 is considered as 1000000 (value which contains 1 at 7^{th} bit and rest values are filled up with 0's) i.e. 64 in decimal, the ASCII of @

Next we extract binary 010, 010, 101 i.e. in decimal 2, 2, 5. Here flag is 0 and total number of 0's are two (10) at position 2 (010) and position 5(101). So, we get 1101101 (value which contains 0 at 2^{nd} and 5^{th} bit positions. Rest of the values are filled up by 1's) i.e., in decimal 109, the ASCII of m.

Next we extract binary 111, 001, 010, 110 i.e. in decimal 7, 1, 2, 6. Here flag is 1 and total number of 1's are three (11) at position 1 (001), 2(010) and 6(110). So, we get 0100011 (value which contains 1 at 1^{st}, 2^{nd} and 6^{th} bit positions. Rest of the values are filled up by 0's) i.e. in decimal 35, the ASCII of #.

Next we extract [21] binary 001, 001 i.e., in decimal 1, 1. Here flag is 0 and total number of 0 is one (01) at position 1 (001). So, we get 1111110 (value which contains 0 at 1^{st} bit position. Rest of the values are filled up by 1's) i.e., in decimal 126, the ASCII of ~.

Next we extract 111, 010, 101 and 110 i.e. in decimal 7, 2, 5 and 6. Here flag is 1 and total number of 1's are 3(11) at positions 2(010), 5(101) and 110(6). So, we get 0110010 (value which contains 1 at 2^{nd}, 5^{th} and 6^{th} bit positions. Rest of the values are filled up by 0's) i.e. in decimal 50, the ASCII of 2.

Next we extract 011, 010, 011 and 100 i.e. in decimal 3, 2, 3 and 4. Here flag is 0 and total number of 0's are 3(11) at positions 2(010), 3(011) and 4(100). So, we get 1110001 (value which contains 0 at 2^{nd}, 3^{rd}, 4^{th} bit positions. Rest of the values are filled up by 1's) i.e. in decimal 113, the ASCII of q.

Finally we extract 011,001,101,111, Here 0 is the flag value, number of 0's are 11 in binary i.e., 3 in decimal, Three 0's are present in 1^{st}, 5^{th} and 7^{th} position. So we get 3, 1, 5 and 7. From here we can understand that flag value is 0. Three numbers of 0's are present at 1^{st}, 5^{th} and 7^{th} position i.e., we retrieve 0101110 and its equivalent ASCII 46 i.e. the ASCII of dot (.). By getting dot (.) we can understand that it is the end of CAPTCHA codes. Thus we extract the LSBs of all participating pixels to retrieve back our CAPTCHA codes B9@m#~2q.

4. RESULTS AND DISCUSSION

For comparing stego image with cover image the commonly used measures are Mean-Squared Error, Peak Signal-to-Noise Ratio.

4.1. PSNR & MSE

In this new encrypted method PSNR [22], MSE, RMSE are standard measurements [23] used to test the quality of the stego images. MSE measures the average of the squares of the errors. The error is the amount by which the pixels value is the difference between the stego image and the cover image. RMSE stands for Root Mean Square Error. PSNR is the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the degree of exactness of its representation. The signal here is the cover image, and the noise is the error caused by the embedded bits into stego image. If the value of PSNR is

high then the quality of the stego image would be good enough. Let the cover image's pixels be represented as C (i, j) and the stego image pixels as S (i,j) for fixed image [24] size of M x N.

$$PSNR = 10 * \log_{10} \frac{255 * 255}{MSE}$$

Where MSE denotes the mean square error, which is given as

$$MSE = \frac{1}{M * N} \sum_{i=1}^{M} \sum_{j=1}^{N} ([S(i,j) - C(i,j)])^2$$

Comparison of PSNR, MSE difference between LSB embedded stego image and new encrypted method embedded stego image as shown in Table 1. A higher PSNR value denotes that difference between cover image and the stego image is more invisible to the human eye interpretation. Cover image and stego image as shown in Figure 1 & 2.

Table 1. Comparison of PSNR, MSE Difference between LSB Embedded Stego Image and New Encrypted Method Embedded Stego Image

Tools	24 bit colour image (LSB)	24 bit colour image by new encrypted method in image steganography (Our Technique)
PSNR	41.22	42.56
MSE	4.72	3.57
RMSE	2.17	1.88

Figure 1. Cover image

Figure 2. Stego image

4.2. Standard Deviation

It is a measure that is used to quantify the amount of variation or dispersion of a set of data values of pixels. A low standard deviation indicates that the pixel values tend to be close to the mean, while a high standard deviation indicates that the pixel values are spread out over a wider range of values. Comparison of SD, variance difference between LSB embedded stego image and new encrypted method embedded stego image as shown in Table 2.

Table 2. Comparison of SD, Variance Difference between LSB Embedded Stego Image and New Encrypted Method Embedded Stego Image

Tools	24 bit colour image	24 bit colour image by new encrypted method in image steganography (Our Technique)
SD	2.1	1.79
VARIANCE	4.41	3.2041

The other Image quality parameters are Normalized Cross Correlation, Average Difference and Maximum Difference. Let the original image's pixels be represented as C (i, j) and the stego image pixels as S(i,j) for fixed image size of M x N. Normalized Cross Correlation (NCC) is defined as:

$$NCC = \sum_{i=1}^{M}\sum_{j=1}^{N}([C(i,j)].[S(i,j)]) \Big/ \sum_{i=1}^{M}\sum_{j=1}^{N}([S(i,j)])^2$$

When the absolute value of the normalized correlation coefficient equals one, then there exist a linear relation between the two samples (cover image & stego image), while on the other hand, when the value of the normalized correlation coefficient equals zero, then the two samples have no linear relation. Average Difference (AD) is defined as:

$$AD = \frac{1}{M * N}\sum_{i=1}^{M}\sum_{j=1}^{N}([S(i,j) - C(i,j)])$$

AD is simply the average difference of pixels between the cover image and stego image. Maximum Difference (MD) is defined as:

$$MD = Max(|S(i,j) - C(i,j)|)$$

MD is the maximum difference of pixels between the cover image and stego image. Comparison of NCC, AD and MD difference between LSB embedded stego image and new encrypted method embedded stego image as shown in Table 3.

Table 3. Comparison of NCC, AD and MD Difference between LSB Embedded Stego Image and New Encrypted Method Embedded Stego Image

Tools	24 bit colour image (LSB)	24 bit colour image by new encrypted method in image steganography (Our Technique)
NCC	0.9965	0.9998
AD	0.013	0.002
MD	3.13	2.95

5. CONCLUSION

Thus we use a new technique to generate an encrypted version of CAPTCHA and then embed it into cover image generating stego image. Attackers cannot have any clue how to retrieve the CAPTCHA codes. Here no need of the use of any encryption [25] or decryption key to embed or extract the CAPTCHA codes. Thus embedding algorithm itself generates the encrypted version of CAPTCHA codes and then embed it into cover image and extraction algorithms itself extracts the actual CAPTCHA codes without the help any key. In future we can use encryption and decryption key along with this algorithm to provide more robustness in the field of image steganography.

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to Prof. R. P. Singh, my research supervisor, for his patient guidance and enthusiastic encouragement. I would also like to thank Prof. Samir Kumar Bandyopadhyay for his advice and assistance in keeping my progress on schedule. I would also like to extend my thanks to the technicians of the laboratory of the Computer Science & Engineering department for offering me the resources in running the program. Finally, I wish to thank my parents and wife for their support and encouragement throughout my study.

REFERENCES

[1] Elaf Ali Abbood, Rusul Mohammed Neamah and Shaymaa Abdulkadhm, "Text in Image Hiding using Developed LSB and Random Method", International Journal of Electrical and Computer Engineering, Vol. 8, No. 4, August 2018, pp. 2091~2097.

[2] N. Krishnaveni and Sudhakar Periyasamy, "A Novel and Innovative Approach for Image Steganography with Chaos", Indonesian Journal of Electrical Engineering and Computer Science, Vol. 11, No. 1, July 2018, pp. 263~267.

[3] Sharif Shah Newaj Bhuiyan, Norun Abdul Malek, Othman Omran Khalifa and Farah Diyana Abdul Rahman, "An Improved Image Steganography Algorithm based on PVD", Indonesian Journal of Electrical Engineering and Computer Science, Vol. 10, No. 2, May 2018, pp. 569~577.

[4] Soumen Bhowmik and Arup Kumar Bhaumik, "A New Approach in Color Image Steganography with High Level of Perceptibility and Security", 2016 International IEEE Conference on Intelligent Control Power and Instrumentation (ICICPI), 21-23 Oct. 2016, ISBN: 978-1-5090-2638-8, DOI: 10.1109/ICICPI.2016.7859718.
[5] Meenakshi S Arya, Meenu Rani and Charndeep Singh Bedi, "Improved Capacity Image Steganography Algorithm using 16-Pixel Differencing with n-bit LSB Substitution for RGB Images", International Journal of Electrical and Computer Engineering, Vol. 6, No. 6, December 2016, pp. 2735~2741.
[6] Sabyasachi Pramanik and S. K. Bandyopadhyay, "An Innovative Approach in Steganography", Scholars Journal of Engineering and Technology, pp. 276-280, 2014.
[7] Raihan Sabirah Sabri, Roshidi Din and Aida Mustapha, "Analysis Review on Performance Metrics for Extraction Schemes in Text Steganography", Indonesian Journal of Electrical Engineering and Computer Science, Vol. 11, No. 2, August 2018, pp. 761~767.
[8] Alaa A. Jabbar Altaay, Shahrin Bin Sahib and Mazdak Zamani, "An Introduction to Image Steganography Techniques", 2012 International IEEE Conference on Advanced Computer Science Applications and Technologies (ACSAT), 26-28 Nov. 2012, DOI: 10.1109/ACSAT.2012.25.
[9] Radha S. Phadte and Rachel Dhanaraj, " Enhanced Blend of Image Steganography and Cryptography", 2017 International IEEE Conference on Computing Methodologies and Communication (ICCMC), 18-19 July 2017, DOI: 10.1109/ICCMC.2017.8282682.
[10] Rupali Jain and Jayshree Boaddh, "Advances in Digital Image Steganography", 2016 International IEEE Conference on Innovation and Challenges in Cyber Security, DOI: 10.1109/ICICCS.2016.7542298
[11] A. Soria-Lorente and S. Berres, "A Secure Steganographic Algorithm Based on Frequency Domain for the Transmission of Hidden Information", Hindawi, Security and Communication Networks, Volume 2017, Article ID 5397082, 14 pages, https://doi.org/10.1155/2017/5397082.
[12] Munesh Chandra Trivedi, Shivani Sharma and Virendra Kumar Yadav, "Analysis of Several Image Steganography Techniques in Spatial Domain: A Survey", ICTCS '16 Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies, Article No. 84., March 04 - 05, 2016, doi>10.1145/2905055.2905294.
[13] Hanizan Shaker Hussain, Roshidi Din, Mohd Hanif Ali and Nor Balqis, "The Embedding Performance of Stego SVM Model in Image Steganography", Indonesian Journal of Electrical Engineering and Computer Science, Vol. 12, No. 1, October 2018, pp. 233~238.
[14] Mamta Juneja, "A Covert Communication Model-Based on Image Steganography", International Journal of Information Security and Privacy, 2014 Pages: 19.
[15] Ebrahim Alrashed and Suood Alroomi,"Hungarian Puzzled Text with Dynamic Quadratic Embedding Steganograph.", International journal of electrical and computer engineering, Vol.7, No.2, April 2017, pp. 799 ~ 809 Vol 12, No 2 November 2018, pp. 716~721.
[16] Reihane Saniei and Karim Faez, "The Security of Arithmetic Compression Based Text Steganography Method", International Journal of Electrical and Computer Engineering, Vol. 3, No. 6, December 2013, pp. 797-804.
[17] Jeevan K. M and S. Krishnakumar, "An Image Steganography Method Using Pseudo Hexagonal Image", International Journal of Pure and Applied Mathematics, Volume 118 No. 18 2018, 2729-2735, ISSN: 1311-8080 (printed version); ISSN: 1314-3395 (on-line version).
[18] Usha B. A, "High Efficient Data Embedding in Image Steganography Using Parallel Programming", Applications of Security, Mobile, Analytic, and Cloud (SMAC) Technologies for Effective Information Processing and Management, 2018 Pages: 14.
[19] Mustafa Cem Kasapbas and Wisam Elmasry, "New LSB-based colour image steganography method to enhance the efficiency in payload capacity, security and integrity check", Sådhanå, https://doi.org/10.1007/s12046-018-0848-4, May 2018.
[20] Mohammad Rasoul PourArian1 and Ali Hanani, " Blind Steganography in Color Images by Double Wavelet Transform and Improved Arnold Transform", Indonesian Journal of Electrical Engineering and Computer Science Vol. 3, No. 3, September 2016, pp. 586 ~ 600 DOI: 10.11591/ijeecs.v3.i2.pp586-600.
[21] Sevierda Raniprima, Bambang Hidayat and Nur Andini, "Digital Image Steganography with Encryption based on Rubik's Cube Principle", 2016 International IEEE Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC), 13-15 Sept. 2016, DOI: 10.1109/ICCEREC.2016.7814972.
[22] Munesh Chandra Trivedi, Shivani Sharma and Virendra Kumar Yadav, "Analysis of Several Image Steganography Techniques in Spatial Domain: A Survey", ICTCS '16 Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies, Article No. 84., March 04 - 05, 2016, doi:10.1145/2905055.2905294.
[23] Shine Zhang, Liang Yang, Xihao Xu and Tiegang Gao, "Secure Steganography in JPEG Images Based on Histogram Modification and Hyper Chaotic System", International Journal of Digital Crime and Forensics, January 2018.
[24] Sabyasachi Pramanik and S. K. Bandyopadhyay, "Image Steganography Using Wavelet Transform and Genetic Algorithm", International Journal of Innovative Research in Advanced Engineering, Vol. 1 pp. 1-4, 2014.
[25] Sabyasachi Pramanik and S. K. Bandopadhyay, "Application of Steganography in Symmetric Key Cryptography with Genetic Algorithm", International Journal of Computers and Technology, Vol. 10, No 7, pp 1791-1799, 2013.

BIOGRAPHIES OF AUTHORS

	Sabyasachi Pramanik is an Assistant Professor in the Department of Computer Science & Engineering, Haldia Institute of Technology. He is pursuing PhD from SSSUTMS, Bhopal. He has 13 years of experience in Teaching and Administration. His research interests include Image Processing and Steganography. He has published many journals of international repute.
	Dr. R.P. Singh is former Director and Prof. Electronics and Communication at Maulana Azad National Institute of Technology, (MANIT) Bhopal. Dr. Singh Graduated and Post Graduated in Electronic Engineering from Institute of Technology (now IIT), B.H.U. Varanasi in 1971 and 1973, respectively. He did his Ph.D. from Barakatullah University Bhopal in 1991. He has 39 years of teaching, research, and administrative experience in Maulana Azad College of Technology (MACT)/MANIT out of which 22 years as Professor. He has worked as Professor In-charge Academic and Chairman Admission Committee Dean (Academic) & Dean (R/D) at MACT /MANIT, Bhopal. He has published 125 papers in National / International reputed and indexed Journals including SCI. He has been member of Executive Committee, Institution of Engineers (I) M.P. Circle. He was Chairman of Computer Society of India. Bhopal. He was member of Board of Studies, and Research Degree committee of many Universities. He has chaired Technical Sessions of various National and International Conferences. He has been Consulting Editor of Journal of Institution of Engineers and reviewer in many International/National Journals.
	Ramkrishna Ghosh is an Assistant Professor in the Department of Information Technology, Haldia Institute of Technology. He is pursuing PhD from KIIT, Bhubaneswar. He has published many journals of international repute. He has authored various books on programming languages like C/C++.