Hiding of CAPTCHA in a colour image using FNP Algorithm

D.S. Haritha
Department of Computer Science and Engineering
Vignan's Institute of Engineering for Women
Visakhapatnam, Andhra Pradesh, India.
harithahani234@gmail.com.

K. VijayaLakshmi
Department of Computer Science and Engineering
Vignan's Institute of Engineering for Women
Visakhapatnam, Andhra Pradesh, India.
vijjusree1999@gmail.com

D. Vasantha
Department of Computer Science and Engineering
Vignan's Institute of Engineering for Women
Visakhapatnam, Andhra Pradesh, India.
vasanthavashu@gmail.com

P. Bhargavi
Department of Computer Science and Engineering
Vignan's Institute of Engineering for Women
Visakhapatnam, Andhra Pradesh, India.
pothina1999@gmail.com

M. Mamatha Laxmi
Assistant Professor
Department of Computer Science and Engineering
Vignan's Institute of Engineering for Women
Visakhapatnam, Andhra Pradesh, India.
mamathamantri@gmail.com

ABSTRACT- Steganography is data hiding technique in internet. Now a days, website became the identity for many businesses. Many companies offer a lot of free services to the users. Every user wants to register on the site. So, for avoiding spam registrations on the site the CAPTCHA is developed. The word CAPTCHA is actually acronym for "Completely Automated Public Turing Test to tell Computers and Human Apart". Generally, computers are not capable of solving captcha. But hackers are looking for ways to bypass this security measure using sophisticated bots. To foolproof websites many CAPTCHA techniques are introduced. But these can be time consuming and frustrate the users. So users may switch to websites without CAPTCHAs. One solution to solve this problem is to display the CAPTCHA only when required to block the bots. We propose a steganography technique in which we send CAPTCHA codes within a cover image. Actually, CAPTCHA codes are embedded into cover image in an encrypted form resulting stego image and thus attackers also cannot fetch the actual CAPTCHA resulting in a secured transmission of confidential data via internet using image steganography.
Keywords: ASCII, CAPTCHA, Cover image, Stego image.

1. INTRODUCTION

1.1 Background

The CAPTCHA is a type of challenge test used in computing to identify whether the user is human or not. CAPTCHA comes in several sizes and of different types. These all works quit well against spam. Some are harder to solve, some are fun and some will benefit you monetarily on your website. There are many types of CAPTCHAs but the most widely used are word solving, Audio, 3D, Math solution, Drang and Drop, Jquery Slider, Tic Tac Toe etc. Online businesses use forms for registration and signups to provide services to their users. Bots usually target such forms and fill them with junk information which bias the acquisition flow metrics for the business. CAPTCHA is usually implemented to stop such spam registrations from bots. but there are certain sophisticated bots that do bypass CAPTCHA and end up spamming forms. Eventhough CAPTCHA do block simple bots from spamming website, there are certain sophisticated bots which started bypassing them by using outsourced teams that can even solve in real time.

1.2 The Problem

To foolproof websites many CAPTCHA techniques like text or image identification, assortment of images, from which users must select all the images that show a particular object etc have been developed. But these can be time consuming and frustrate the users. Sometimes it is very difficult to read and time consuming to decipher. So users may switch to websites without CAPTCHAs. One solution to solve this problem is to display the CAPTCHA only when required to block the bots. This can be done by embedding the CAPTCHA in an image and on clicking a generate but CAPTCHA should be generated. In traditional approach, CAPTCHA codes are embedded directly using LSB technique. We propose a new embedding technique called FNP to embed the CAPTCHA in the image. Here F stands for Flag bit value, N stands for Number of occurrences of 0's and 1's and P stands for Position of bits.

II. LITERATURE SURVEY

The art of data hiding in digital media ie., steganography and watermarking, aims to embed secret data into cover image with purpose of identification, copyright protection, and annotation. The main constraint factors of this process are message data quantity, necessity of invariability of embedded data under distortions like lossy compression, third party removal, or modification. Data hiding techniques fall into three categories of cryptography, steganography, and watermarking. Watermarking and particularly steganography tend to conceal presence of hidden data while cryptography makes data gibberish.

2.1 Data Hiding In Audio Signal- Least significant bit (LSB) coding: LSB is the simplest way to embed information in a digital audio file. By substituting the least significant bit of each sampling point with a binary message, LSB coding allows for a large amount of data to be encoded. Among many different data hiding techniques proposed to embed secret message within audio file, the LSB data hiding technique is one of the simplest methods for inserting data into digital signals in noise free environments, which merely embeds secret message-bits in a subset of the LSB planes of the audio stream. A data message is hidden within a cover signal in the block called embeddor using a stego key, which is a secret set of parameters of a known hiding algorithm. This proposed system is to provide a good, efficient method for hiding the data from hackers and sent to the destination in a safe manner. This proposed system will not change the size of the file even after encoding and also suitable for any type of audio file format. Low-bit encoding embeds secret data into the least significant bit (LSB) of the audio file.

2.2 Hiding Text Within Image Using LSB Replacement: Least significant bit(LSB) is the most popular stegnography technique. It hides the secret message in RGB image. In this, a modified method of data hiding by LSB substitution method is developed to embed a secure text in the gray image, so that the interceptors will not notice about the existence of that text. A text image is embeded in a gray image using a random key. The hiding process included concealing the plain text using column by column technique until the entire text is completed.

2.3 An Image Stegnography Method Using Pseudo Hexagonal Image: Hexagonal image processing is a new developing area of image processing, in which the images are represented using hexagonal shaped pixels rather than square pixels. In this, a method for hiding a secret image represented in Pseudo hexagonal structure is proposed. The pseudo hexagonal pixel is constructed from the square pixel of original image using JK algorithm. Using this method

the entire secret image without any loss is embedded in the cover image. Since we are hiding the entire bits of secret image using the JK algorithm, this technique is capable of preserving the MSE, PSNR and SSIM index of the secret image. This indicates that, the quality of a secret image which is hidden in the cover image is preserved 100%.

MOTIVATION

Steganography can be defined as the hiding the data like file, image, video or any message to the other file, image, video or message. In Steganography the useless bits are actually replaced by the useful bits in order to hide the required file into any of the files or data. If a good cryptographic cipher is used, it is likely that no one, not even a government entity, will be able to read it.

All steganography requires is a *cover text*, which is where data will be hidden, a *message* that is made up of data, an *algorithm* that decides how to hide the data, and frequently, a *key* that will be used to randomize the placement of the data and perhaps even encrypt it. This hidden information can be plain text, cipher text, or even images. The carrier files may be any image, audio and video.

The applications like Stegan PEG, Open Stego and so on are used to fulfill the purpose of wrapping up one file into another. The applications used for Steganography hides the bits of the required file into another file in a manner so that the original file does not lose its characteristics. It can be considered pretty more secure than encryption or hashing as in these cases the attacker can sniff at least the junks but in the case of Steganography, they won't be able to detect if anything important has been transmitted. It is usually applied at a place where the data has to be sent secretly.

It plays a vital role in cybersecurity by allowing the legitimate users or peers to send the data in a way which is highly secured so that it could be protected from the hacker or malicious users who are intended to harm or abuse the system. It can be done using software that is available in the market for free or paid.

III. PROPOSED ALGORITHM

In our proposed system we embed CAPTCHA into a 24-bit colour image which is our cover image. We use LSB of Red, Green and Blue pixels of 24-bit cover image and the embedding technique is based on an FNP algorithm which 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 no 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 value of CAPTCHA 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 of lesser number of 0's or 1's would be embedded which makes attacker impossible to retrieve the actual data which is the ASCII value of the embedded character. A layer of encryption has been taken place here. At the receiver's end at first, we extract flag's value. Based on the flag's value we get the lesser occurrences of 0's or 1's, next how many 0's or 1's is present would be extracted. After this the positions of lesser number of 0's or 1's would be extracted. This is the process to extract the whole information about the embedded ASCII. Thus, we gather whole information to retrieve the original ASCII that was sent via CAPTCHA through communication channel in internet.

3.1 Generation of CAPTCHA:

Step 1: Generate a random number of size between 8 and 16. Let it be n. (We consider that our CAPTCHA would be minimum of 8 characters and maximum of 16 characters).

Step 2: Initialize a string str as str=""

Step 3: Perform following task for n times
Step 4: Generate random numbers between 97 and- 122 (ASCII of a to z) and place it into the variable char.
Step 5: Concatenate, str=str+char
Step 6: Generate random number between 48 and 57 (ASCII of 0 to 9) and place it into the variable char
Step 7: Concatenate, str=str+char
Step 8: Generate random number between 65 and 90 (ASCII of A to Z) and place it into the variable char
Step 9: Concatenate, str=str+char
Step 10: 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.
Step 11: Concatenate, str=str+char
Step 12: Randomly we choose any step from 3 to 7 (where these steps) to generate any kind of char and keep concatenating.
Step 13: Repeat the step 8 until n number of characters has been generated.
Step 14: Terminate the CAPTCHA codes with the ASCII value dot (.) as the end of CAPTCHA code.
Step 15: Jumble up characters of CAPTCHA for a random number of times for repositioning of characters and thus the final CAPTCHA code is ready.

3.2 Embedding Algorithm
Step 1: Check the length of the CAPTCHA. Take one by one character and do the following
Step 2: Take the ASCII value of the character.
Step 3: Convert the ASCII value into equivalent binary value.
Step 4: Check number of 0's and 1's in this binary value.
Step 5: If the number of 0's is less than number of 1's then flag value would be 0 otherwise flag value would be 1.
Step 6: Let the flag value be 0 and then we count how many 0’s are available
Step 7: Modify the LSB of red pixel of RGB cover image to store the flag value.
Step 8: Embed the number of 0's into LSB of Green and Blue components of pixel
Step 9: Embed the position of the occurrence of 0's at the LSB of next RGB pixels as per requirements.
Step 10: Repeat the following steps until all the characters get embedded into cover image.
Step 11: Stego image is generated.

3.3 Extraction Technique
Step 1: Read first red, green and blue pixel of stego image.
Step 2: Extract flag bit based on the LSB of red pixel.
Step 3: Extract LSB of green and blue pixel to calculate how many 1's or 0's are present.
Step 4: Read next pixel's LSBs to find the positions of 0's or 1's.
Step 5: Based on this value, ASCII can be obtained. Thus, relevant character, digit or special character can be found out.
Step 6: Repeat the step 1 to 4 where are these steps or 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.

IV. EXPERIMENT AND RESULT

To construct CAPTCHA codes, we consider the ASCII value in 7 bits 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 B9AmdP2q. It has a length of 8 characters except dot. Dot must used as a terminating character. First, we deal with B. Its ASCII value is 66. Binary 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 2nd & 7th bit position. Now we embed flag 1, then the number of 1's i.e., two, then bit positions of 1's, which are 2nd & 7th 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 as 3, 2, 3, 7, where 3 means 011 denoting 0 as flag, 3 as number of 0's. 0's are present in 2nd, 3rd & 7th position, resulting in 2, 3 & 7.

Next character is A. Its ASCII value is 65. Binary equivalent of 65 is 1000001. In 1000001, 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 is present in 1st and 7th bit position. Now we embed flag 1, then the number of 1's i.e., two, then bit positions of 1's, which is at 1st and 7th. We embed this value by LSB modification of RGB coloured cover image. Thus, the ASCII value 65 is encrypted as 6,1,7. 6 come from 110, where 1 is the flag value and 10 is the number of 1's.

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 2nd & 5th bit position. Now we embed flag 0, then the number of 0's i.e., two, then bit positions of 0's, which are 2nd & 5th. 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 d. Its ASCII value is 100. Binary equivalent of 100 is 1100100. In 1100100, 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 3rd,6th & 7th bit position. Now we embed flag 1, then the number of 1's i.e., three, then bit positions of 1's, which are 3rd,6th & 7th. We embed this value by LSB modification of RGB colour cover image. Thus, the ASCII value 100 is encrypted as 7, 3, 6 and 7. 7 come from 111, where 1 is the flag value and 11 is the number of 1's. 3, 6 & 7 come from the positions of 1's.

Next character is P. Its ASCII value is 80. Binary equivalent of 80 is 1010000. In 1010000, 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 is present in 5 & 7th bit position. Now we embed flag 1, then the number of 1's i.e., five and seven, then bit positions of 1's, which is at 5th &7th position. We embed this value by LSB modification of RGB colour cover image. Thus, the ASCII value 80 is encrypted as 6,2,5. 1 comes from 110, where 1 is the flag value and 10 is the number of 1'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 2nd, 5th and 6th bit positions. Now we embed flag 1, then the number of 1's i.e., three, then bit positions of 1's, which is 2nd, 5th and 6th. We embed this value LSB modification of RGB colour cover image. Thus, the ASCII value 50 is encrypted as 7, 2, 5

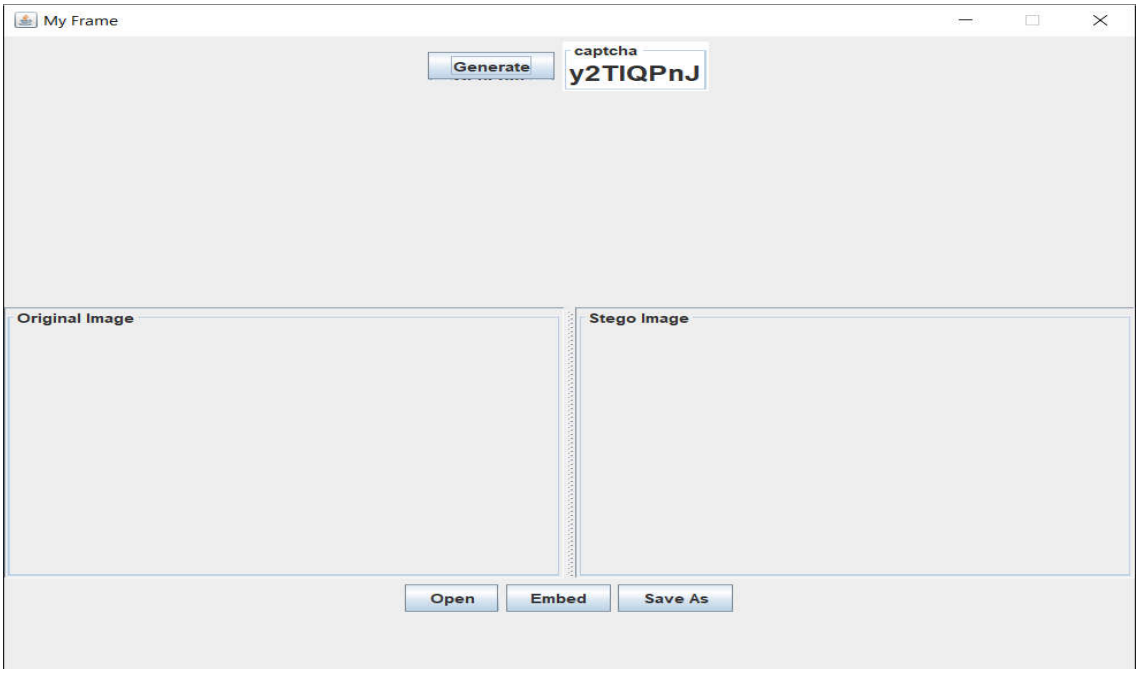
and 6. 7 come from 111, where 1 is the flag value and 11 is the number of 1's. 2, 5 and 6 come from the positions of 1's.

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 2nd, 3rd, 4th bit positions. Now we embed flag 0, then the number of 0's i.e., three, then bit positions of 0's, which are 2nd, 3rd, 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.

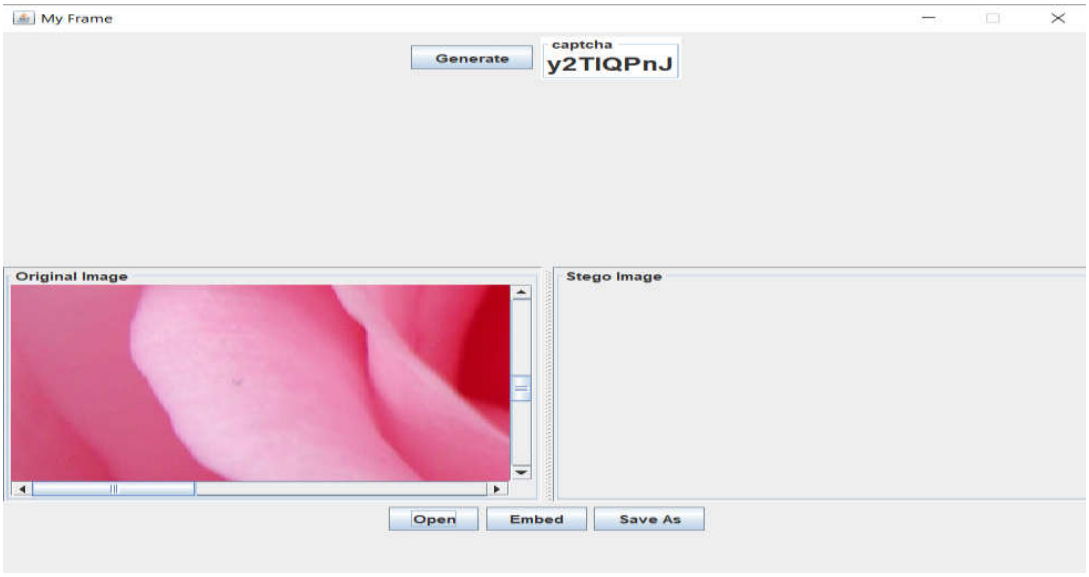
Finally, we embed dot (.). Its ASCII value is 46. Binary equivalent is 0101110. In 0101110, there are three 0's and four 1's. As the number of 0's is less, so we take flag as 0. 0's are present in 1st, 5th and 7th bit positions. Now we embed flag 0, then the number of 0's i.e., three, then bit positions of 0's, which are 1st, 5th and 7th. 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 B9AmdP2q. is embedded in an encrypted form into LSB's of 24 bit RGB cover image as 627323761722573676257256323 4 During extraction first pixel's RGB 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. 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 7th bit and rest values are filled up with 0's) i.e. 65 in decimal, the ASCII of Next we extract binary 110, 001, 111 i.e. in decimal 2, 1, 7. Here flag is 1 and total number of 1's are two(10) at position 1(001) and position 7(111). So, we get 1000001 (value which contains 0 at 1^{st} and 7^{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 1st, 2nd and 6th bit positions. Rest of the values are filled up by 0's) i.e. in decimal 35, the ASCII of d. Next, we extract binary 111,011,110,111 i.e., in decimal 7,3,6,7. Here flag is 1 and total number of 1 is one (01) at position 3 (011) and position 6 (110) and position 7 (111). So, we get 1100100 (value which contains 1 at bit position. Rest of the values are filled up by 0's) i.e., in decimal 80, the ASCII of P. Next, we extract 110, 101, 111 i.e. in decimal 6, 5, 7. Here flag is 1 and total number of 1"s are 2(01) at positions 5(101), 7(111). So, we get 1010000 (value which contains 1 at 5th and 7th bit positions. Rest of the values are filled up by 0"s) i.e. in decimal

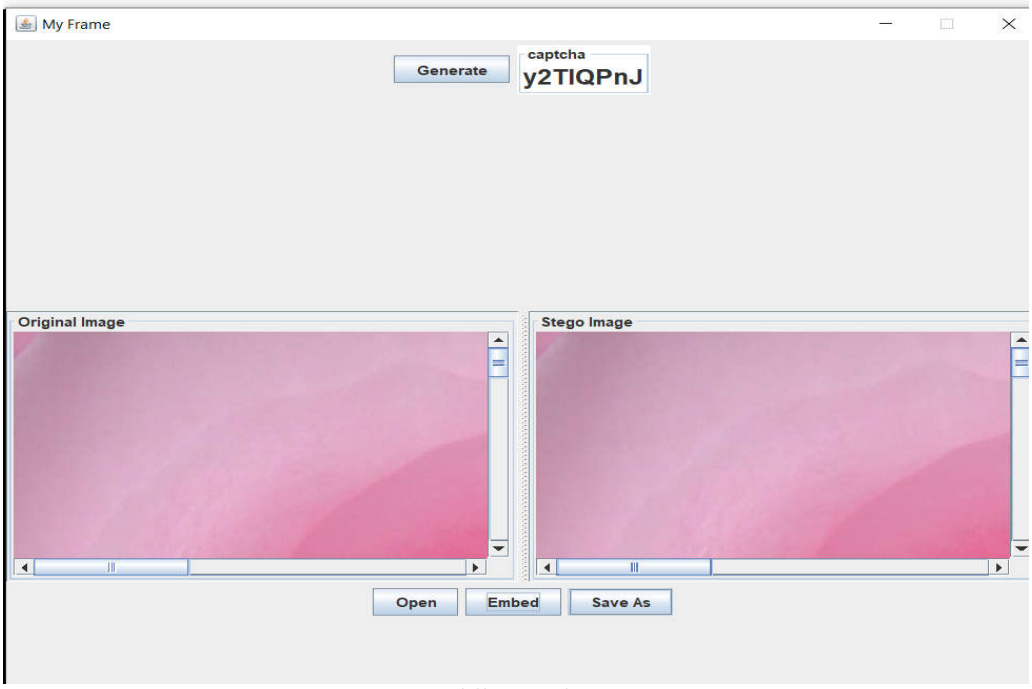
50, the ASCII of 2. Next, we extract 011, 010, 011 and 100 i.e. in decimal 3, 2, 3 and 4. Here flag is 0 and total number of 0"s are 3(11) at positions 2(010), 3(011) and 4(100). So, we get 1110001 (value which contains 0 at 2nd, 3rd, 4th bit positions. Rest of the values are filled up by 1's) i.e. in 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 1st, 5th and 7th 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 1st, 5th and 7th 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 B9AmdP2q.



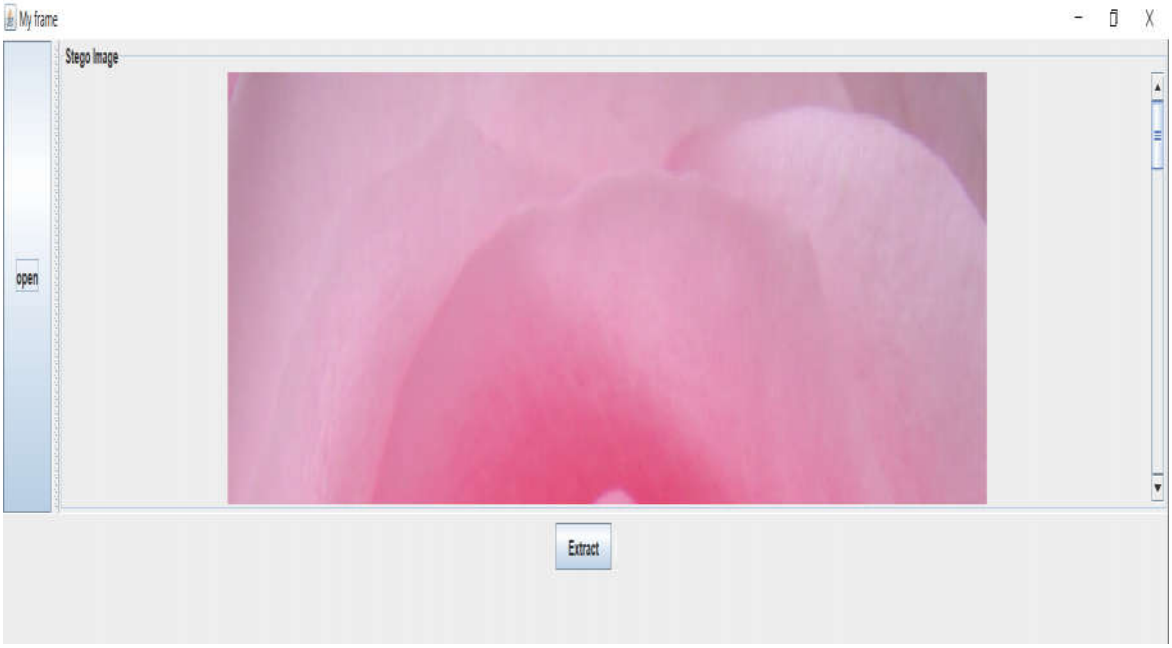
Generation of captcha at embedding side



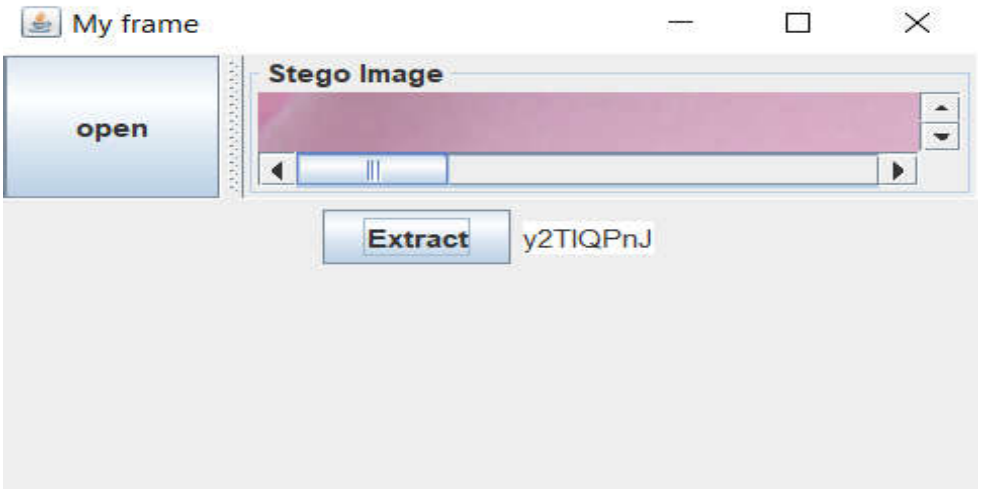
Selected pixels of image



Emedding an image



Extaction of image from file



Extraction of captcha from stego image

V. CONCLUSION

In this paper we discussed how to generate CAPTCHA and then embed it into cover image to generate stego image. CAPTCHA can be displayed only when we require to block bots. Attackers cannot have any clue of how to retrieve the CAPTCHA codes. We don't require any key to extract or embed 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.

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).