Indonesian Journal of Electrical Engineering and Informatics (IJEEI)
Vol. 8, No. 3, September 2020, pp. 525~531
ISSN: 2089-3272, DOI: 10.11591/ijeei.v8i3.831

A Unique Way to Generate Password at Random Basis and Sending it Using a New Steganography Technique

Sabyasachi Pramanik[1], R. P. Singh[2], Ramkrishna Ghosh[3], Samir K Bandyopadhyay[4]
[1]Department of Computer Science and Engineering, Haldia Institute of Technology, India
[2]Vice Chancellor, Sri Satya Sai University of Technology and Medical Sciences, India
[3]Department of Information Technology, Haldia Institute of Technology, India
[4]Advisor to Chancellor, JIS University, India

Article Info

Article history:

Received Nov 14, 2018
Revised Apr 30, 2019
Accepted Jun 23, 2020

Keywords:

Steganography
Password
LSB Technique
PSNR
MSE

ABSTRACT

Data hiding is a technique for the secure transmission of confidential data. Many data hiding techniques exist and steganography is the most important one. This paper presents a new steganography method in the spatial domain. We use steganography to send confidential information from sender to receiver. Here, we generate a password at random basis in a unique way based on system time and date. Then we send this confidential password using steganography by implementing a totally new embedding and extraction technique based on the exact length of bits in the binary representation of ASCII values. Here, confidential text information is embedded into cover image generating a stego image and sent to the receiver maintaining top-level secrecy.

Corresponding Author:

Sabyasachi Pramanik,
Department of Computer Science and Engineering,
Haldia Institute of Technology,
ICARE Complex, Hatiberia, Dist: Midnapore (E), West Bengal, India.
Email: sabyalnt@gmail.com

1. INTRODUCTION

1.1. Background

Data transmission is the most vital task on the Internet. Sometimes confidential data plays such a major role that it cannot be sent directly. So, confidential data hiding is one of the most important aspects of data transmission. This is accomplished by different steganography techniques.

1.2. The Problem

Here, text is the confidential information treated as a password and it is generated using an innovative method. If the generated password is embedded into the cover image using the LSB technique, huge LSB modification would be needed that can bring distortion in a stego image. It is a genuine problem. So, the password is embedded into a cover image generating a stego object using a new steganography embedding technique from sender's end and is later sent and extracted at receiver's end using steganography extraction technique. Here, both embedding and extraction techniques use much lesser LSB modification compared to direct LSB technique. Therefore much lesser image distortion takes place indeed.

1.3. The Proposed Solution

We use an innovative technique to generate a password that ensures uniqueness at each instance. This password is generated based on system date and time at a random basis and then sent as confidential text [1] into a cover image using steganography [2, 3, 4]. Embedding [5] and extraction techniques are also applied using an innovative approach which assures adequate space saving and lesser modification in a stego image

[6] compared to existing LSB [7, 8, 9] technique. Here, system date and time is taken to generate the password for each instance and then positions of the characters of the password are randomly changed to finalize the ultimate password which is chosen as confidential information. This confidential information is embedded into a cover image [10] using a unique embedding technique based on the exact length of bits in the binary representation of ASCII [11] values. This needs minimal modification at LSB of the cover image and assures space saving over the existing LSB technique.

2. RESEARCH METHOD

2.1. Step 1: Password Generation

A password is generated at random basis [12]. Inputs are taken from system date and time at the instance of the generation of passwords. Then an hour, minute and second is extracted from time. Similarly, day, month and year is extracted from date.

Next, initialize an integer no.1.

Initialize an integer flag.

no.1=hour mod 10

Convert the hour into words. For each of the instance, the flag's value would be either 0 or 1 based on the value of an hour. Here 0 stands for upper case conversion and 1 stands for lower case conversion in words. This is the resultant string. Now, randomly choose two positions of the string based on its length and pick up those two characters.

```
if (no.1 >=0 && no.1<=4) then
        {
                flag = 1
        }
if (no.1>=5 && no.1<=9) then
        {
                flag = 0
        }
```

Special Character Generation in password creation:

Generate a random [9] number from (33 to 47) or (58 to 64) or (91 to 96) or (123 to 126) and take it as ASCII of the generated special character. Concatenate no.1, the randomly chosen two characters of hour expressed in words either in an upper or lower case based on the flag's value (0 or 1) and the special character. This process would be repeated for the rest of the inputs like a second, minute, day, month and year. Then shuffle the position of characters of the concatenated string in random order for n number of times. Thus the final password would be generated. This final password ensures high-level security [13].

For e.g., let time and date be 9:13:56 AM and 25/12/2017 respectively. Here, no.1=9 mod 10 = 9

Here 9 is within the range of 5 and 9, so flag = 0.

Thus the upper case of 9 in words is NINE. This is the resultant string. Then any two positions of the string are chosen randomly based on its length. Say, the positions are 2 and 4. So the two characters I and E of NINE are picked up. Now, a random number has been chosen from the specified range (33 to 47) or (58 to 64) or (91 to 96) or (123 to 126). Say it is 35. Take it as the ASCII value of #. Now concatenate 9, I, E and # to construct the substring of final password string as 9IE#. By repeating this process for a second, minute, day, month and year, we construct the entire final password.

According to our example, the password is 9IE#3te@6SX!5FV^2wo&7VN:

Let us consider the final password is S#w7VE9@Voe^3N6:t!F5&I2X after shuffling [14] 3 times.

For embedding this password using the existing LSB technique, 24*8=192 bits are needed. Let us check how many bits are needed to embed the password using the following new embedding technique.

Step 2: Steganography Embedding Technique

We take advantage of non-printing ASCII range from (0 to 31) in our embedding technique. So we consider the values of (0 to 9) directly instead of their ASCII values to embed and using a lesser number of LSBs as these numbers are within the range (0 to 31). We define type 0 to type 7 based on the decimal numbers' exact length of bits on a binary representation. This is expressed using Table 1 given below.

Table 1. Decimal Range to Binary Range based on Type

Type	Decimal Range	Binary Range	The exact length of Bits in the binary representation
0	0 to 1	0 to 1	1
1	2 to 3	10 to 11	2
2	4 to 7	100 to 111	3
3	8 to 15	1000 to 1111	4
4	16 to 31	10000 to 11111	5
5	32 to 63	100000 to 111111	6
6	64 to 95	1000000 to 1011111	7
7	96 to 127	1100000 to 1111111	7

Here our password is expressed in digits, characters and special symbols. As the ASCII range (128 to 255) is non-printable characters, we don't need to consider this range for password generation.
We consider specific treatment so that space saving [15] can provide more desirable outcomes.
For Type 0, Decimal Range is 0 to 1. Here Customized Range is also (0 to 1). Additive value is 0 as (0+0) = 0 to (1+0) = 1
For Type 1, Decimal Range is 2 to 3. Here Customized Range is (0 to 1). Additive value is 2 as (0+2) = 2 to (1+2) = 3
For Type 2, Decimal Range is 4 to 7. Here Customized Range is (0 to 3). Additive value is 4 as (0+4) = 4 to (3+4) = 7
For Type 3, Decimal Range is 8 to 15. Here Customized Range is (0 to 7). Additive value is 8 as (0+8) = 8 to (7+8) = 15
For Type 4, Decimal Range is (16 to 31). Here Customized Range is (0 to 15). Additive value is 16 as (0+16) = 16 to (15+16) = 31.
For Type 5, Decimal Range is (32 to 63). Here Customized Range is (0 to 31). Additive value is 32 as (0+32) = 32 to (31+32) = 63
For Type 6, Decimal Range is (64 to 95). Here Customized Range is (0 to 31). Additive value is 64 as (0+64) = 4 to (31+64) = 95
For Type 7, Decimal Range is (96 to 127). Here Customized Range is (0 to 31). Additive value is 96 as (0+96) = 96 to (31+96) = 127
Above customization is shown using Table 2.

Table 2. Decimal Range to Customized Range with Additive Value based on Type

Type	Decimal Range	Customized Range (used for saving LSBs)	Additive value to get the exact decimal value	LSBs needed to embed value based on Customized Range
0	0 to 1	0 to 1	0	1
1	2 to 3	0 to 1	2	1
2	4 to 7	0 to 3	4	2
3	8 to 15	0 to 7	8	3
4	16 to 31	0 to 15	16	4
5	32 to 63	0 to 31	32	5
6	64 to 95	0 to 31	64	5
7	96 to 127	0 to 31	96	5

In the cover image, we consider the first three LSBs as type value and then take the number of LSBs needed to embed value based on customized range. For example, say, in our password, a digit is 7. Then its type is 2 and after customization, its equivalent number is 3. Three LSBs are used to store type 2. Next two LSBs are used to store customized value 3. Later the equivalent decimal value can be obtained as (3 + 4) =7, where 4 is the additive value. So in our embedding technique (3 + 2) = 5 bits are needed to store 7 whereas 8 bits are needed to store 7 in regular LSB technique.

In our password, the type can be of (0 to 7) only. So for type specification, we always consider 3 LSBs. This length is fixed in nature and then we use the required number of LSBs for embedding, based on the value of Customized Range. Thus we proceed using one after another character of the password. At first, we

use 3 LSBs to embed Type of the character and then the minimum required LSBs to embed the value of Customized Range. Then we again consider the type and value of the next character [16] and so on.

First, three LSBs' embedded information of type will signify the length of the character so that we can count the number of LSBs used for embedding. For embedding, our needed types are from Type 0 to Type 7 i.e., out of total no. of 8 types, we need 3 LSBs per type.

After this consideration, we create another table to have a final glance regarding LSB saving compared to the regular LSB technique where each character is expressed in 8 bits. It also gets embedded using 8 LSBs. Here, Table 3 shows the clear idea of using the LSBs to save space compared to regular LSB technique.

Table 3. Total No. of LSB needed to embed based on Type and Value 3

Type	LSB needed to embed Type	LSB needed to embed the value	The need for Total No. of LSB	LSB saving
0	3	1	4	4
1	3	1	4	4
2	3	2	5	3
3	3	3	6	2
4	3	4	7	1
5	3	5	8	0
6	3	5	8	0
7	3	5	8	0

Thus, we can see that in our technique, we can save LSBs to embed using Type 0 to Type 4 i.e., for 5 types but for Type 5 to Type 7 i.e., for 3 types the LSBs needed for embedding is the same as regular LSB technique.

2.2. Extraction Technique

According to [17],at first 3 LSBs would be used to check the type of value.
Next LSBs are extracted based on type.
After extraction, additive values are considered for having the exact values for calculation purpose based on the type.
Repeat the above steps until the end of the password string.
Thus embedded password would be extracted.

3. RESULTS AND DISCUSSIONS

To calculate Peak Signal to Noise Ratio (PSNR) [18, 19] we have to first calculate Mean Square Error (MSE) [20] between the cover image and the stego image. Given a noise-free $m \times n$ monochrome image I [21] and its noisy approximation K, MSE is defined as:

$$MSE = \frac{1}{m\,n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \quad (1)$$

The PSNR (in dB) is defined as:

$$PSNR = 10.log_{10} \frac{MAX^2}{MSE} \quad (2)$$

Here, MAX is the maximum possible pixel [22] value of the image. When the pixels are represented using 8 bits per sample, this is 255.
The higher the value of PSNR, the more the similarity between the hidden image and the cover image. In the existing LSB technique, we embed the password in the least significant bit of red, green and blue components of the cover image to generate a stego image. We calculate the PSNR of the stego image. Then we calculate the PSNR of the stego image generated from the Space Saver LSB technique i.e., our method. The results are displayed in Table 4.

Table 4. Comparison of PSNR of existing LSB technique and our Space Saver LSB technique

Stego Image using existing LSB technique			Stego Image using Space Saver LSB technique		
PSNR_RED (dB)	PSNR_GREEN (dB)	PSNR_BLUE (dB)	PSNR_RED (dB)	PSNR_GREEN (dB)	PSNR_BLUE (dB)
32.34	33.61	33.22	68.63	68.71	68.46

It can be seen that the PSNR value of our Space Saver LSB technique is much more improved compared with the existing LSB technique. So our Space Saver LSB technique is highly improved compared with existing LSB technique. Cover image and the stego image are shown below as Figure 1 and Figure 2 respectively.

Figure 1. Cover Image

Figure 2. Cover Image with Password i.e., Stego Image.

4. CONCLUSION

If we just simply consider embedding 8 characters of each of the type, from Type 0 to Type 7, we can see that total no. of LSBs saved is (4+4+3+2+1) = 14 out of 8*8=64 LSBs which are exactly needed for embedding in existing LSB technique. We refer our technique as a Space Saver LSB technique. Here we need (4 + 4 + 5 + 6 + 7 + 8 + 8 + 8) = 50 LSBs for embedding by using our Space Saver LSB Technique. This result is shown in Figure 3.

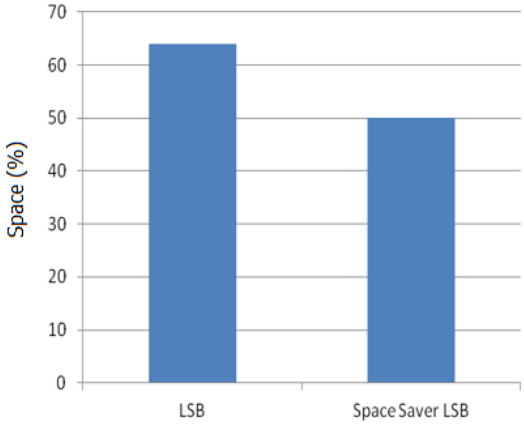

Figure 3. Techniques for embedding

So, (64-50)/64*100=21.875% space has been saved using Space Saver LSB compared to existing LSB Technique. Percentage of space saving would be varied in different instances.

ACKNOWLEDGEMENTS

I am indebted to Prof. Samir Kumar Bandyopadhyay, Advisor to Chancellor, JIS University, for numerous prospective comments on various drafts of the manuscript and for bringing to my attention gaps in my knowledge and logic. Finally, I wish to thank my parents for their support and encouragement throughout my study.

REFERENCES

[1] Firas A. Jassim, "A Novel Steganography Algorithm for Hiding Text in Image using Five Modulous Method", International Journal of Computer Applications, Vol. 72, No-17, June 2013.

[2] Christy Atika Sari, Giovani Ardiansyah, De Rosal Ignatius Moses Setiadi and Eko Hari Rachmawanto, "An Improved Security and Message Capacity using AES and Huffman Coding on Image Steganography", 2018 ICW TELCOMNICA International Conference.

[3] 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.

[4] Ebrahim Alrashed and Suood Alroomi, "Hungarian Puzzled Text with Dynamic Quadratic Embedding Steganography.", 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.

[5] Hanizan Shaker Hussain, Roshidi Din, Mohd Hanif Ali and Nor Balqis, "The Embedding Performance of StegSVM Model in Image Steganography", Indonesian Journal of Electrical Engineering and Computer Science, Vol. 12, No. 1, October 2018, pp. 233~238.
[6] 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 (IJECE), Vol. 6, No. 6, December 2016, pp. 2735- 27401.
[7] Ammar Y. Tuama **e**t al, "Randomized Pixel Selection for Enhancing LSB Algorithm Security against Brute-Force Atack", Journal of Mathematics and Statistics", 2017, DOI: 10.3844/jmssp.2017.127.138.
[8] Chunfang Yang et. al., "Pixel Group Trace Model-Based Quantitative Steganalysis for Multiple Least Significant Bits Steganography", IEEE Transaction Information Forensics and Security, Vol. 8, No. 1, January 2013.
[9] K. Thangadurai and G.Sudha Devi, "An analysis of LSB based image steganography techniques", International Conference on Computer Communication and Informatics (ICCCI), 3-5 Jan 2014, Coimbatore, India.
[10] Xinyi Zhou et. al., "An Improved method for LSB Based Color Image Steganography Combined with Cryptography". IEEE ICIS, June 2016.
[11] Reza Tavoli et. al., "A New Method for Text Hiding in the Image by Using LSB", International Journal of Advanced Computer Science and Applications, Vol. 7, No. 4, 2016.
[12] 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.
[13] Ammar Y. Tuama **e**t al, "Randomized Pixel Selection for Enhancing LSB Algorithm Security against Brute-Force Atack", Journal of Mathematics and Statistics", 2017, DOI: 10.3844/jmssp.2017.127.138.
[14] Sabyasachi Pramanik and Samir K. Bandyopadhyay, "Hiding Secret Message in an Image", International Journal of Innovative Science, Engineering and Technology, Vol 1 pp. 553-559, 2014.
[15] Sabyasachi Pramanik, R. P. Singh and Ramkrishna Ghosh, "Steganography with Modified LSB Technique Blended with Asymmetric Cryptography", IEEE International Conference on Inventive Computation Technologies (ICICT-2018), pp. 318-323.
[16] Sabyasachi Pramanik and Samir K. Bandyopadhyay, "Application of Steganography in Symmetric Key Cryptography with Genetic Algorithm", International Journal of Computers and Technology, Vol. 10, No 7, pp 1791-1799, 2013.
[17] 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.
[18] Sabyasachi Pramanik, R. P. Singh and Ramkrishna Ghosh, "A New Encrypted Method in Image Steganography", Indonesian Journal of Electrical Engineering and Computer Science, Vol. 14, No. 3, June 2019, pp. 1412~1419.
[19] Sabyasachi Pramanik and Samir K. Bandyopadhyay, "Hiding Secret Message in an Image", International Journal of Innovative Science, Engineering and Technology, Vol 1 pp. 553-559, 2014.
[20] 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.
[21] Bhanupriya Katre and Bharti, "Dynamic Key based LSB Technique for Steganography", International Journal of Computer Applications (0975 – 8887), Volume 167 – No.13, June 2017.
[22] Sabyasachi Pramanik and Samir K Bandyopadhyay, "An Innovative Approach in Steganography", Scholars Journal of Engineering and Technology, pp. 276-280, 2014.

BIOGRAPHIES OF AUTHORS

Sabyasachi Pramanik is an Assistant Professor in the Department of Computer Science & Engineering, Haldia Institute of Technology. He is pursuing Ph.D. from Sri Satya Sai University of Technology and Medical Sciences, Bhopal. He has 13 years of experience in Teaching and Administration. His research interests include Data Hiding, Image Processing, Steganography and Artificial Intellegence. 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 a member of Executive Committee, Institution of Engineers (I) M.P. Circle. He was Chairman of Computer Society of India. Bhopal. He was

a member of the 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 Ph.D. from KIIT, Bhubaneswar. He has published many journals of international repute. He has authored various books on programming languages like C/C++.

Prof. Samir Kumar Bandyopadhyay has Ph.D. Degrees in both Engineering and Medical Science. He has published 600 papers in National and International Journals. Seventy Scholars got their Ph.D. Degree under his supervision. He has 34 years of teaching experience in both areas. He was Registrar, University of Calcutta and also held the post of Vice-Chancellor in a State Aided University. Presently he is working as Advisor to Chancellor of JIS University, West Bengal, India.