Suresh Gyan Vihar University Journal of Engineering & Technology
(An International Bi-Annual Journal)
Vol . 1, Issue 2, 2015,pp.22-26
ISSN: 2395-0196

An Integrated Approach of Data Security on Server via Stegnography using 4 bit LSB

Udit Yadav[1], Akhilesh Pandey[2]

[1]Research. Scholar, Suresh Gyan Vihar University, Jaipur
[2]Asst. Professor, Computer Science & Engineering, Suresh Gyan Vihar University, Jaipur

Abstract – In this article the implementation and analysis of image steganographic process is carried out to secure the data on a network. LSB encoding technique is utilized to hide the data in the LSB of 24 bit RGB pixels of colored cover image. The intermediate cover image containing the data (stegano image) shows a good correlation with the original cover image meaning a good amount of hiding of data. If the size of the data file to be sent is more than the capacity of the cover image then multiple copies of cover image are used to embed the data. The correlation of recovered data file with the original data file comes out to be 1 meaning a perfect recovery.

Keywords: LSB, Steganographic, RGB, Data Security

I. INTRODUCTION

In recent times, the significance of ensuring the network data integrity has been highlighted by the following research works [1–5]. These techniques, while can be useful to ensure the storage correctness without having users possessing data, cannot address all the security threats in network data storage, since they are all focusing on single server scenario and most of them do not consider dynamic data operations. As a complementary approach, researchers have also proposed distributed protocols [6–8] for ensuring storage correctness across multiple servers or peers. Again, none of these distributed schemes is aware of dynamic data operations. As a result, their applicability in network data storage can be drastically limited. Steganography is the art and science of writing hidden messages in such a way that no one apart from the intended recipient knows of the existence of the message [9]. Due to growing need for security of data image steganography is gaining popularity [10]. The main goal of steganography is to communicate securely in a completely undetectable manner [11] and to avoid drawing suspicion to the transmission of a hidden data [12]. This idea of data hiding is not a novelty, it has been used for centuries all across the world under different regimes but to date it is still unknown to most people – is a tool for hiding information so that it does not even appear to exist. However Steganography operates at a more complex level as detection is dependent on recognizing the underlying hidden data. In the presented text the implementation and analysis of image steganographic process is carried out to secure the data on network. LSB encoding technique is utilized to hide the data in the LSB of 24 bit RGB pixels of colored cover image. The intermediate cover image containing the data (stegano image) shows a good correlation with the original cover image meaning a good amount of hiding of data. If the size of the data file to be sent is more than the capacity of the cover image then multiple copies of cover image are used to embed the data. The correlation of recovered data file with the original data file comes out to be 1, meaning a perfect recovery.

Steganography and encryption are both used to make sure data prudence. However the main discrimination among them is that with encryption any person can see that both parties are communicating in secret. Steganography hides the continuation of a clandestine communication and in the best case nobody can see that both parties are communicating in secret. This makes steganography appropriate for a number of responsibilities for which encryption isn't, such as copyright marking. Adding together encrypted copyright information to a file could be easy to eliminate but embedding it inside the stuffing of the file itself can avoid it being simply recognized and disinterested.

II. LSB TECHNIQUE FOR HIDING THE DATA

The least significant bit (in other words, the 8th bit) of some or all of the bytes inside an image is changed to a bit of the secret message. Digital images are mainly of two types (i) 24 bit images and (ii) 8 bit images. In 24 bit images we can embed three bits of information in each pixel, one in each LSB position of the three eight bit values for example:

Let us consider a data byte 1 0 1 1 0 1 0 1

Let the RGB component of 3 pixels read from cover image be:

	RED	GREEN	BLUE
Pixel – 1 (163, 232, 185)	10100011	11101000	10111001
Pixel – 2 (26, 243, 57)	00011010	11110011	00111001
Pixel – 3 (181, 30, 133)	10110101	00011110	10000101

The RGB component of these 3 pixels in the stegano image shall be:

	RED	GREEN	BLUE
Pixel – 1 (163, 232, 185)	10100011	11101000	10111001
Pixel – 2 (27, 242, 57)	00011011	11110010	00111001
Pixel – 3 (180, 31, 133)	10110100	00011111	10000101

Bits marked red are the bits from data byte (most significant bit to least significant bit) read. Here blue component of every third pixel remains unchanged. In this case, only four bits needed to be changed to insert the character successfully. The resulting changes that are made to the least significant bits are too small to be recognized by the human eye, so the message is effectively hidden. An automated system is designed to upload and download the data to network and from the network, the flow charts for the uploading the data and downloading the data are demonstrated in the Fig.1 and Fig.2 below

a) Encoding Process :

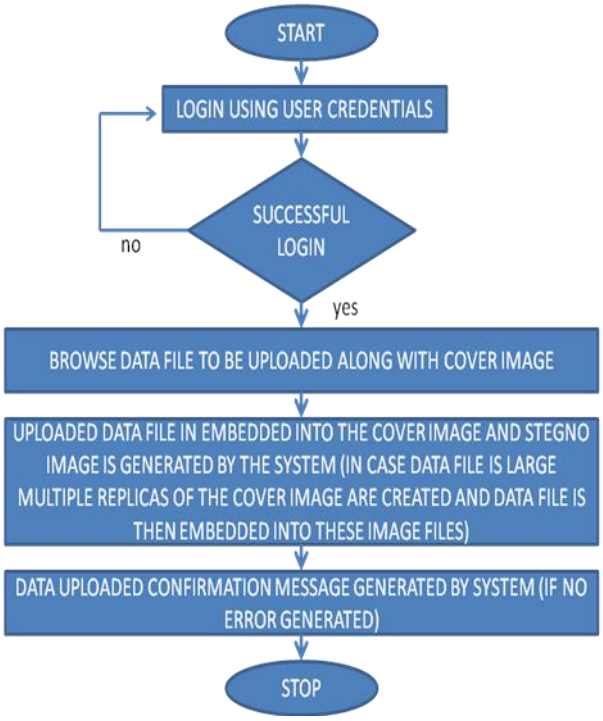


Fig.1 Flowchart of data up load on network

b) Decoding Process:

c) Data Extraction

d) The extraction process is as follows.

e) **Inputs** : Stego-image file, stego-key

f) **Output**: Secret message.

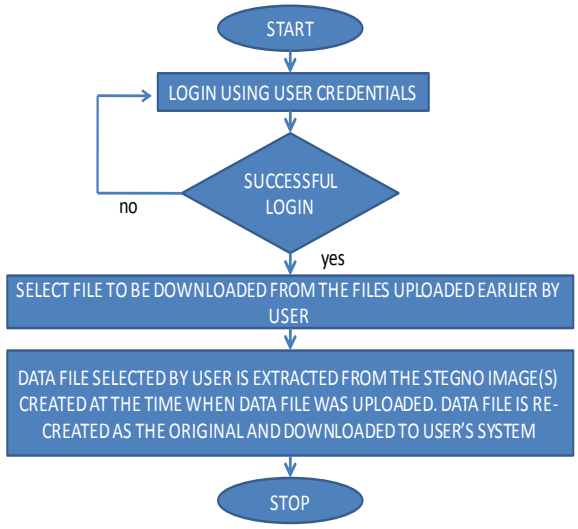


Fig.2 Flowchart of data down load on network

Procedure:

- Step 1: Extract the pixels of the stego image.
- Step 2: Now, start from first pixel and extract stego key characters from first component of the pixels. Follow Step3 up to terminating symbol, otherwise follow step 4.
- Step 4: If this extracted key matches with the key entered by the receiver, then follow Step 5, otherwise terminate the program.
- Step 5: If the key is correct, then go to next pixels and extract secret message characters from first component of next pixels. Follow Step 5 till up to terminating symbol, otherwise follow step 6.
- Step 6: Extract secret message [18, 20].

Decryptions (Size_data_file, Number_file)

// Here Size_data_file is original size of the data file, Number_file is the number of Stegno-images created while encryption.

- **Step 1.** Initialize byte_array[Size_data_file], byte_index = 0

II. ALGORITHM

//Initialize byte_array [] of bytes equal to number of bytes in the original data file.

- **Step 2.** For i = 1 to Number_file step by 1 Loop
- **Step 3**. Retrieve Stegno_image(i)

//Read i[th] Stegno_image

- **Step 4**. Initialize bit_array[8], bit_position = 0

//Array to be used for calculating each byte of the data file

- **Step 5**. For x=0 to Stegno_image(i).Width-1 step by 1Loop

- **Step 6**. For y=0 to Stegno_image(i).Height -1 step by 1 Loop

- **Step7.** Read pixel_color = Stegno_image(i).Pixel(x, y)

//Read value of Pixel(x,y) into color_pixel

- **Step 8.** Set red_array = pixel_color.Red

//Retrieve the Red component for color_pixel as a 8-bit array (range 0 -255)

Set green_array = pixel_color.Green

//Retrieve the Green component for color_pixel as a 8-bit array (range 0 -255)

Set blue_array = pixel_color.Blue

//Retrieve the Blue component for color_pixel as a 8-bit array (range 0 -255)

- **Step 9.** Set bit_array[bit_potision++] = red_array[7]

//Set bit_array[bit_position] to LSB of blue component of pixel, and increment bit_position by 1

Set bit_array[bit_potision++] = green_array[7]

//Set bit_array[bit_position] to LSB of blue component of pixel, and increment bit_position by 1

If bit_potision < 8 then

Set bit_array[bit_potision++] = blue_array[7]

//Set bit_array[bit_position] to LSB of blue component of pixel, and increment bit_position by 1

Else

bit_potision = 0;

//Reset bit_position

byte_array[byte_index++] = bit_array

//transfer bit_array to byte_array at position byte_index and increment byte_index by 1

End If

- **Step 10.** End Loop (y)
- **Step 11.** End Loop (x)
- **End 12.** If byte_index = Size_data_file then

Write byte_array[] as data file

//Save byte_array[] as data file.

Step 13. Exit

The algorithm used can be summarized as like we consider a cover image Image_1 of size 120*140, i.e. width of image is 120 pixels and height is 140 pixels, and a data file data.txt of size 200KB. Here total number of pixels in the image = 120 * 140 = 16800. In every pixel 3-bits of data can be embedded, one bit at least significant bit of RGB component of pixel, so total data bits that can be embedded in the cover image = 16800 * 3 = 50400 bits or 6300 bytes.

Total byte of the data file = 200 * 1024 = 204800

Here number of bytes in the data file is greater than byte that can be accommodated in the cover image. Thus we need multiple copies of the cover image to embed the data file, number of Ceiling (204800 / 6300) = 33. So 33 copies of cover image will be created rather denying the client as size of data file is larger than the number of bytes that can be accommodated in the cover image.

Now the data file will also be broken down in 33 small chunks by the process, where first 32 chunks will of size 6300 bytes each and last chunk will be of 3200 bytes (204800 – 6300 * 32). Now iterative process will be called to embed every bit of data file into the 33 copies of the cover image to generate the stegno images which shall then be uploaded over the network.

In such way the files having a large size can also be uploaded by parts practically having no limit over the data size to be uploaded. The Fig 3 below shows the process in a sequential manner. The complete process can be summarized as follows:

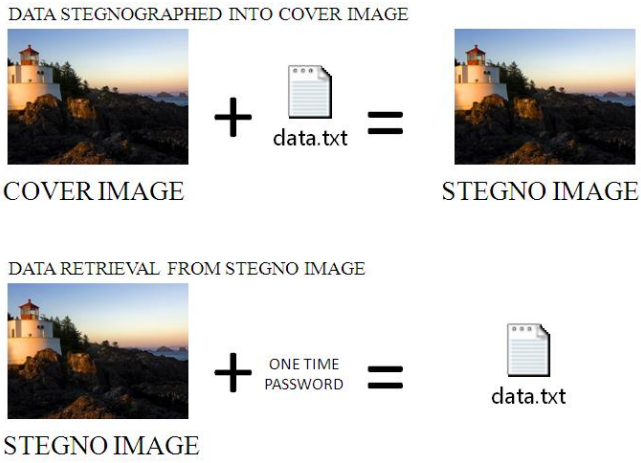


Fig.3 The process of image steganography

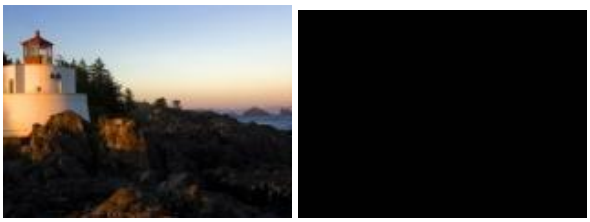
In this article an example is taken in which a text data file is hidden in an image. The text file is having a larger size then the capacity of the cover image so multiple copies of the cover image are created to hide the complete data.

III. ANALYSIS OF THE DESIGNED SYSTEM

In this section an example is taken in which a text data file is hidden in an image. The text file is having a larger size then the capacity of the cover image so multiple copies of the cover

image are created to hide the complete data. The difference images and stegano images at each and every step are analyzed. The data matrices of each image are produced in order to justify the process results and the graphs are drawn for each image RGB components. The correlation between each stegano image is calculated. The text file size used is having a size of 592 bytes. The cover image taken is having 3x15x21= 945 pixels. To hide one byte of data 9 (3 RGB Pixels) pixels are required so the given image can handle 945/ 9 = 105 bytes of data. According to the capacity of the cover image six replicas of cover image are needed to encrypt the complete date. Six Stegano images will be produced to hide the complete data.

The Stegano image 1 produced is shown in Fig. 5.2 (a). It is the enlarged copy of the actual Stegano image produced. The difference image is also created by taking the difference of the cover image and the Stegano 1 image. The difference image is almost black showing no significant difference in the pixel values of the cover image. This makes the hiding of the data at a good level.



(a) (b)

Fig. 4 (a) Stegano Image 1 (b) Difference from cover image

Here to estimate the similarity of the cover image with the stegano image the correlation of the RGB component is taken with respective components of each stegano image and the results of the correlation are summarized below: Correlation Coefficient shows amount of similarity between two matrices. Here the calculated value Correlation Coefficients of stegano-images with respect to Cover Image are:

Table 1. Correlation Coefficients Of All Image Components

corr2(R, R1) = 1
corr2(G, G1) = 1
corr2(B, B1) = 1
corr2(R, R2) = 1
corr2(G, G2) = 1
corr2(B, B2) = 1
corr2(R, R3) = 1
corr2(G, G3) = 1
corr2(B, B3) = 1
corr2(R, R4) = 1
corr2(G, G4) = 1
corr2(B, B4) = 1
corr2(R, R5) = 1
corr2(G, G5) = 1
corr2(B, B5) = 1
corr2(R, R6) = 1
corr2(G, G6) = 1
corr2(B, B6) = 1

Here R, G, B are matrices for Red, Green, Blue component of the Cover Image respectively, R1, G1,B1 are matrices for Red, Green, Blue component of the Stegno-Image(1) respectively.

To extend the possibility of mathematical verification of security of the method the entropy of the cover image and the different stegano images are calculated and the percent difference between the entropy of the cover image and stegano images are calculated

Table 2. Entropy Of Various Stegano Images Compared With The Original Image

		% Variation
Entropy of Cover-Image	7.3491	
Entropy of Stegno-Image(1)	7.3279	0.2885
Entropy of Stegno-Image(2)	7.3302	0.2572
Entropy of Stegno-Image(3)	7.3453	0.0517
Entropy of Stegno-Image(4)	7.3378	0.1538
Entropy of Stegno-Image(5)	7.3520	0.0395
Entropy of Stegno-Image(6)	7.3456	0.0476

From the above table of the entropies it very clear that the difference between entropies the cover image and the respective stegano images is very less also the Percentage of variation in Entropy of Cover-Image and Stegano –images is coming very less that is a very less of fraction showing a good amount matching of the images making the difference delectability very less.

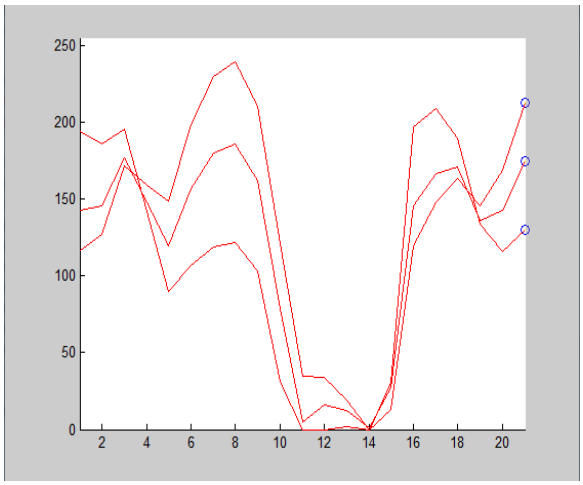


Fig.5 Plot of RGB component of the cover image

IV. CONCLUSIONS

In recent years, the extended growth in server storage provider industry continues to drive the requirements for more and more applications on the servers. These services can be utilized without any investment on bulky IT set up and a large demand to purchase the software. A large pool of the services provided by the serve providing firms without any investment on set up or IT infrastructure no need to hair IT professionals, no need to train the IT resources it is just pay and get the service at a very reasonable rate. The services provided by the servers are so cheap and straight forward but at the same time posses the threat of data insecurity. Because in this the data is available in an remote server, which clients simultaneously is used by several clients at a time in case of theft of the data or any expose of data to any false client knowingly or unknowingly may arise the questions on the data security.

An image steganographic process using the LSB encryption technique is used to secure the data over the networks in such scheme the data to be transferred is hidden in the LSBs of pixels of the cover image so that he data or information to be sent is in the form of LSBs of the cover image which cannot be detected without any prior information of availability of data in that image. The conclusions of the work carried out are summarized below

- An automatic system is designed to upload to data on the networks the data as soon as uploaded on the network are encrypted in the LSB,s of a cover image and the data is no longer available as the uploaded format but in the encrypted format now.
- The data containing capacity of the cover image is finite so if the data exceeds the capacity of the cover image a multiple copies of the cover image are created the data file is broken in the comestible size pieces and assembled back at the time of recovery.
- An stegano image is created with respect to each cover image embedded with data and the correlation of that stegano image comes out to be 1 in each case, proving that no one can detect the difference in stegano image and cover image.
- The entropy of each stegano image is having a very less difference with the entropy of the cover image again justifying the a good hiding possibility of data in images.

V. REFERENCES

[1]. Juels and J. Burton S. Kaliski, "PORs: Proofs of Retrievability for Large Files," *Proc. of CCS '07*, pp. 584–597, 2007.

[2]. H. Shacham and B. Waters, "Compact Proofs of Retrievability," *Proc. of Asiacrypt '08*, Dec. 2008.

[3]. K. D. Bowers, A. Juels, and A. Oprea, "Proofs of Retrievability: Theory and Implementation," Cryptology ePrint Archive, Report 2008/175, 2008, http://eprint.iacr.org/.

[4]. G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," *Proc. Of CCS '07*, pp. 598–609, 2007.

[5]. G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession," *Proc. of SecureComm '08*, pp. 1–10, 2008.

[6]. T. S. J. Schwarz and E. L. Miller, "Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage," *Proc. of ICDCS '06*, pp. 12–12, 2006.

[7]. M. Lillibridge, S. Elnikety, A. Birrell, M. Burrows, and M. Isard, "A Cooperative Internet Backup Scheme," *Proc. of the 2003 USENIX Annual Technical Conference (General Track)*, pp. 29–41, 2003.

[8]. K. D. Bowers, A. Juels, and A. Oprea, "HAIL: A High-Availability and Integrity Layer for Network Storage," Cryptology ePrint Archive, Repor 2008/489, 2008, http://eprint.iacr.org/.

[9]. R. Chandramouli and N. Memon,"Analysis of LSB based Image Steganography", IEEE ICIP, pp. 1022-1022, Oct. 2001.

[10]. R.J. Anderson, F.A.P. Petitcolas, "On The Limits of Steganography", IEEE Journal ofSelected Area in Communicafions, pp. 474-481, May 1998.

[11]. N.F. Johnson, S. Jajodia, "Staganalysis: The Investigation of Hiding Information", IEEE, pp. 113-116, 1998.

[12]. H. Hastur, Mandelsteg, ftp://idea.sec.dsi.unimi.it/pub/security/crypt/code/