A Method for Public Key Method of Steganography

Prof. Samir Kumar Bandyopadhyay
Dept. of Computer Science
& Engineering, University of Calcutta
92 A.P.C. Road, Kolkata – 700009, India

Sarthak Parui
Institute of Engineering & Management
Sector V, Salt Lake,
Kolkata-700091, India

ABSTRACT
Security of the digital information is becoming primary concern prior to transmitting the information itself via some media. Information security means protecting information and information systems from unauthorized access, use, disclosure, disruption, modification or destruction. In this paper, a public key method of Steganography is proposed under standard cryptographic assumptions. The byte location in LSB of which the secret bit is to be embedded is found out by public key of the receiver and receiver apply private key of itself to reconstruct the secret message following RSA assumptions.

General Terms
Steganography, Security.

Keywords
Communication, Security, Steganography

1. INTRODUCTION
Steganography is a technology that hides a message within an object, a text, or a picture. It is often confused with cryptography, not in name but in appearance and usage. The easiest way to differentiate the two is to remember steganography conceals not only the contents of the message but also the mere existence of a message. The original steganographic applications used "null ciphers" or clear text. A null cipher conveys that the message has not been encrypted in any way, whether it is using basic character shifting, substitution or advanced modern day encryption algorithm. So, the message is often in plain view but for a reason can either not be detected as being present or cannot be seen once detected. As is common with cryptography, steganography has its roots in military and government applications and has advanced in ingenuity and complexity.

Steganography is the science of hiding secret information by means of some carrier file [1]. The secret information in general is embedded into some media file like image or audio and thus it is transmitted so as to prevent an opponent from guessing that some secret information is being transmitted. So, the main objective of Steganography is not to let the opponent guess that any kind of information apart from the media file itself is transmitted. In spatial domain of Steganography by using Image as the carrier file, we, in general, invert the Least Significant Bit of a particular byte of the carrier image to embed a particular bit of Secret message [2,3,4]. This method is known as LSB (Least Significant Bit) masking method of Steganography.

In this paper, a public key method of Steganography is proposed under RSA cryptographic assumptions. In the proposed algorithm, RSA is used not to encrypt the secret message but to find the exact Byte location of the carrier file, in LSB of which a particular bit of the secret message is to be embedded. For a particular bit of the message file, we employ RSA encryption algorithm to generate a cipher. The LSB of the Red or Green or Blue value of a particular pixel represented by the cipher is then inverted only if the particular bit of the secret message is one. Likewise, we can embed the entire secret message into the carrier file and thereby we can send the embedded file called stego image and the carrier file to intended recipient. Upon receiving, recipient can apply XOR operation on the carrier image and the stego image to find the inverted LSBs. Then applying RSA Decryption algorithm on the location of a particular inverted LSB, we get the original position of the bit in the message file and thus can reconstruct the message file. Thus, this algorithm scatters the information at the time of hiding based on a particular public key-private key combination.

2. RELATED WORK
In modern Steganography, secret information can be embedded in any kind of multimedia file, viz. image, audio or video. We mostly use image file as the carrier prior to send the secret information via some media. Now if we modify the carrier image, its statistical properties get changed. Hence an opponent can detect the distortion from the statistical property of the resulting stego image. Embedding higher entropy data changes the histogram of the carrier image significantly. Thus, it is a common practice to embed data in the Least Significant Bits of the carrier image file.

The majority of today's steganographic systems use images as cover media because people often transmit digital pictures over email and other Internet communication. Several methods exist to employ the concept of Steganography as well as plenty algorithms have been proposed in this regard. To gather knowledge regarding our approach, we have concentrated on some techniques and methods which are described below.

In the field of image security, Miroslav Dobsicek [1] has developed an interesting application of steganography where the content is encrypted with one key and can be decrypted with several other keys, the relative entropy between encrypt and one specific decrypt key corresponds to the amount of information. Yusuk Lim, Changsheng Xu and David Dagan Feng, 2001, described the web based authentication system consists of two parts: one is a watermark embedding system and the other is authentication system. In case of watermark embedding system, it is installed in the server as application software that any authorized user, who has access to server, can generate watermarked image. The distribution can use any kind of network transmission such as FTP, email etc. Once image is distributed to externally, client can access to authentication web page to get verification of image [2].

Min Wu and Bede Liu, June, 2003, proposed [3] a new method to embed data in binary images, including scanned text, figures, and

signatures. The method manipulates "flippable" pixels to enforce specific block based relationship in order to embed a significant amount of data without causing noticeable artifacts. They have applied Shuffling before embedding to equalize the uneven embedding capacity from region to region. The hidden data can then be extracted without using the original image, and can also be accurately extracted after high quality printing and scanning with the help of a few registration marks.

Rehab H. Alwan, Fadhil J. Kadhim, and Ahmad T. Al- Taani, 2005, has explained a method with three main steps. First, the edge of the image is detected using Sobel mask filters. Second, the least significant bit LSB of each pixel is used.

Finally, a gray level connectivity is applied using a fuzzy approach and the ASCII code is used for information hiding. The prior bit of the LSB represents the edged image after gray level connectivity, and the remaining six bits represent the original image with very little difference in contrast. The given method embeds three images in one image and includes, as a special case of data embedding, information hiding, identifying and authenticating text embedded within the digital images [4]. In 2007, Nameer N. EL- Emam proposed an algorithmic approach to obtain data security using LSB insertion steganographic method. In this approach, high security layers have been proposed through three layers to make it difficult to break through the encryption of the input data and confuse steganalysis too [5].

S. K. Bandyopadhyay, Debnath Bhattacharyya, Swarnendu Mukherjee, Debashis Ganguly, Poulami Das in 2008 has proposed a heuristic approach to hide huge amount of data using LSB steganography technique. In their method, they have first encoded the data and afterwards the encoded data is hidden behind a cover image by modifying the least significant bits of each pixel of the cover image. The resultant stego-image was distortion less. Also, they have given much emphasis on space complexity of the data hiding technique [6].

There is also a good method proposed by G. Sahoo and R. K. Tiwari in 2008. Their proposed method works on more than one image using the concept of file hybridization. This particular method implements the cryptographic technique to embed two information files using steganography. And due to this reason they have used a stego key for the embedding process [7].

Unfortunately, modifying the cover image changes its statistical properties, so eavesdroppers can detect the distortions in the resulting stego-image's statistical properties. In fact, the embedding of high-entropy data (often due to encryption) changes the histogram of colour frequencies in a predictable way. So, in order to obtain more security in our prescribed method, we have embedded the secret information behind an image of 8 times the size of secret information file to hide any remarkable change in the final image and also it helps the secret information remain scattered throughout the carrier image which will make the changes in the histogram look like noise.

3. PROPOSED WORK

Prior to discuss about the algorithm in detail, it is better to mention the selection of the cover object and the information which is to be steganographed behind the cover. In general, secret information is like signatures, secret text, secret images, formulae etc in any convenient format. The cover file or the carrier file can be any kind of multimedia file. Here, we refer INFO_FILE as the secret message file and the cover object as COVER_IMG. Here, as a cover object 24-bit Bitmap Image is used. The size of COVER_IMG should be at least 8 times of that of the INFO_FILE.

As for RSA assumption, we determine 3 sufficiently large numbers e, d & n to have {e, n} as the public key and {d, n} as the private key. For any file, it is trivial to get the bit representation. For INFO_FILE, we obtain the bit representation at first. For the image file, i.e. for COVER_IMG, we obtain the pixel representation as an array of {R, G, B} values. For every bit of INFO_FILE, say, if location of the bit is P, then we obtain C=P^e mod n; then we determine a=C/3 and b=C%3. If Pth bit of INFO_FILE is 1, then at ath pixel of the COVER_IMG, we invert LSB of b; Now, if b=0, we invert LSB of B value, if b=1, then we invert LSB of G value, otherwise LSB of R value is inverted. From this new value, new pixel is formed. Likewise for every bit of INFO_FILE, we generate new pixel, if necessary, and thereby, we form the STEGO_IMG (the final image where the secret information is embedded). Both the COVER_IMG and the STEGO_IMG is sent to the intended recipient.

At the receiver end, we make logical XOR between the {R, G, B} values of STEGO_IMG & COVER_IMG. As a result of which we get the bits which were inverted. Say, we get G value of ath pixel is inverted. Then we get C=a*3 + 1, and original location of that bit in the INFO_FILE as P, where P=C^d mod n. Likewise, after processing the entire STEGO_IMG, we fill rest of the bits of generated INFO_FILE with zero to obtain the original INFO_FILE.

The detailed algorithm is described taking a 24-bit image as cover image (assuming public key {e, n} and private key {d, n} under RSA assumption is pre-determined):

3.1 MAIN ()

This is the main algorithm which will be called in the Sender's side and this will invoke other methods.

1. Select the file bearing the information or the message to be hidden as INFO_FILE.
2. Call CALCULATE_INFO_FILE (INFO_FILE) method which will be calculating the size of INFO_FILE as number of bytes and store it in variable named size_constraint.
3. Call SELECT_COVER_IMG (size_constraint) method which will select a cover image suitable as per constraints and it will return the image itself.
4. Call HIDE (INFO_FILE, COVER_IMG) method to embed secret information of INFO_FILE into COVER_IMG. This method will return the STEGO_IMG
5. Send both STEGO_IMG & COVER_IMG to intended recipient.

3.2 CALCULATE_INFO_FILE (INFO_FILE)

This function will calculate the size of the INFO_FILE in number of bytes and then the product 8*size.

Arguments: This function will take INFO_FILE as argument and finally it will output the required product 8*size.

1. Count the number of bytes of the INFO_FILE (say it is s).
2. Calculate 8*s.
3. Return the above product.

3.3 SELECT_COVER_IMG (size_constraint)

This algorithm will select the Cover image as per size_constraint.

Argument: This method takes size_constraint as the argument and finally it will output the COVER_IMG.

1. Select an image file and check to see if size of the image in terms of bytes >= size_constraint.
2. If so, then the image can be accepted and return the image; otherwise repeat step 1 again.

3.4 HIDE (INFO_FILE, COVER_IMG)

This algorithm embeds secret message into the cover image and prepare STEGO_IMG.

Argument: This method takes INFO_FILE and COVER_IMG as arguments and finally it outputs STEGO_IMG.

1. Compute pixel values of COVER_IMG as an array of {R, G, B} values.
2. Iterate through each bits of INFO_FILE starting from the 1st bit and execute following steps.
3. Let P be the position of the current bit of INFO_FILE under consideration. We determine C=P^e mod n, a=C/3, b=C%3, where {e, n} is the public key of the receiver.
4. If P^{th} bit of INFO_FILE is set to 0, then go to step 7.
5. Now, we are to deal with ath pixel. If b=0, then invert LSB of B value, if b=1, then invert LSB of G value, otherwise invert LSB of R value.
6. Compute new pixel value and replace it with the original value for that pixel in the pixel array.
7. If EOF is reached from INFO_FILE, then break, otherwise pick the next bit of INFO_FILE and go to step 3.
8. Obtain STEGO_IMG from the new pixel array and return it to invoker method.

3.5 RECOVERY (STEGO_IMG, COVER_IMG)

This method is invoked at the receiver end and used to recover the INFO_FILE from the STEGO_IMG & COVER_IMG received.

Argument: This method takes STEGO_IMG & COVER_IMG as the arguments and finally outputs recovered INFO_FILE.

1. We, at first, XOR the R, G, B values of STEGO_IMG with those of COVER_IMG.
2. As a result of XOR, we get the bits which were inverted.
3. We maintain a matrix to represent the INFO_FILE to be generated.
4. For each of the bits inverted, we process as stated in following steps.
5. We assume b=0 indicates B value, b=1 indicates G value and b=2 indicate R value.
6. Let LSB of b value of ath pixel is found to be inverted after XOR operation and is currently under consideration.
7. We compute C=a*3+b and P=C^d mod n, where {d, n} is the private key of the receiver.
8. We, now, set the Pth bit of the matrix for INFO_FILE to be generated as 1.
9. Go to step 6 until all the bits which were inverted are exhausted.
10. Fill other bits of the matrix for INFO_FILE with zero.
11. Thus byte by byte restore will result into extracted INFO_FILE.

4. RESULTS

Before moving into the implementation of STEGO_IMG, INFO_FILE and suitable COVER_IMG must be chosen. Here a handwritten signature in JPEG file is chosen as INFO_FILE, which is shown in Figure 1.

Figure 1. INFO_FILE [8493 bytes]

The module calculation in section III.B calculates the size of the INFO_FILE as 8493 bytes and size_constraint as 67944 bytes (=8493*8).

Maintaining the size_constraint, the module in section III.C chooses the cover object as COVER_IMG as in Figure 2.

Figure 2. COVER_IMG [492534 bytes]

Implementation of HIDE module as discussed in section III. D is thereby performed which produces the STEGO_IMG as in Figure 3.

Figure 3. STEGO_IMG [492534 bytes]

At the receiver side, both STEGO_IMG and COVER_IMG are passed as argument to RECOVERY module as discussed in section III.E which produces the extracted INFO_FILE at the receiver side as shown in Figure 4.

Figure 4. Extracted INFO_FILE [8493 bytes]

Another test is performed taking the following text file as input INFO_FILE.

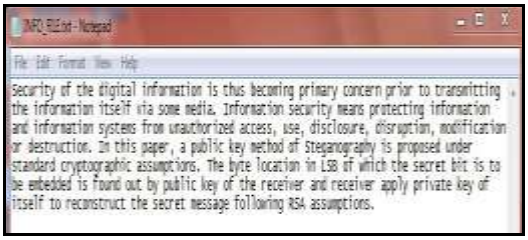

Figure 5. INFO_FILE [606 bytes]

The module calculation in section III.B calculates the size of the INFO_FILE as 606 bytes and size_constraint as 4848 bytes (=606*8).

Maintaining the size constraint, the module in section III.C chooses the cover object as COVER_IMG as in Figure 6.

Figure 6. COVER_IMG [723654 bytes]

Implementation of HIDE module as discussed in section III.D is thereby performed which produces the STEGO_IMG as in Figure 7.

Figure 7. STEGO_IMG [723654 bytes]

At the receiver side, both STEGO_IMG and COVER_IMG are passed as argument to RECOVERY module as discussed in section III.E which produces the extracted INFO_FILE at the receiver side as shown in Figure 8.

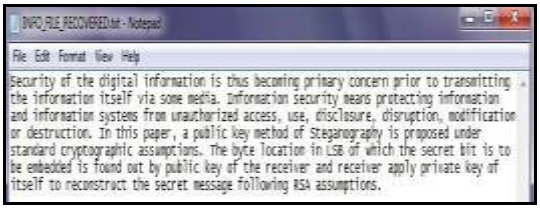

Figure 8. Extracted INFO_FILE [606 bytes]

5. CONCLUSION

This paper proposes an approach to implement public key algorithm in Steganography. Here, in this paper, Public key algorithm is used based on RSA assumptions with traditional LSB modification scheme for Steganography to randomize the position of the secret bit in cover image. So, it disperses the bits of INFO_FILE throughout the COVER_IMG unsystematically. Thus it makes almost impossible for an attacker to guess the secret information. In this paper, the basic algorithm is implemented using bitmap image as COVER_IMG and text file or any image file as INFO_FILE, but this algorithm can perfectly work with any kind of image as COVER_IMG.

6. REFERENCES

[1] Dobsicek, M., Extended steganographic system. In: 8th Intl. Student Conf. On Electrical Engineering, FEE CTU 2004, Poster 04.

[2] Yusuk Lim, Changsheng Xu and David Dagan Feng, "Web based Image Authentication Using Invisible Fragile Watermark", 2001, Pan-Sydney Area Workshop on Visual Information Processing (VIP2001), Sydney, Australia, Page(s): 31 - 34

[3] Min Wu, Member, IEEE, and Bede Liu, Fellow, IEEE, "Data Hiding in Binary Image for Authentication and Annotation", IEEE Trans. Image Processing, volume 6, Issue 4, Aug. 2004 Page(s): 528 - 538

[4] Rehab H. Alwan, Fadhil J. Kadhim, and Ahmad T. Al-Taani, "Data Embedding Based on Better Use of Bits in Image Pixels", International Journal of Signal Processing Vol 2, No. 2, 2005, Page(s): 104 – 107

[5] Nameer N. EL-Emam "Hiding a large amount of data with high security using steganography algorithm", Journal of Computer Science. April 2007, Page(s): 223 – 232

[6] S.K.Bandyopadhyay, Debnath Bhattacharyya, Swarnendu Mukherjee, Debashis Ganguly, Poulumi Das, "A Secure Scheme for Image Transformation", August 2008, IEEE SNPD, Page(s): 490 – 493

[7] G. Sahoo, R. K. Tiwari, "Designing an Embedded Algorithm for Data Hiding using Steganographic Technique by File Hybridization", January 2008, IJCSNS, Vol. 8, No. 1,Page(s): 228 – 233.