Padding and Zonal Substitution Scheme for Image Watermarking

Samir K Bandyopadhyay, Debnath Bhattacharyya[1], Debashis Ganguly[1], Swarnendu Mukherjee[1] and Poulami Das[1]

University of Calcutta, Senate House, 87 /1 College Street, Kolkata – 700073
[1]Computer Science and Engineering Department, Heritage Institute of Technology, Anandapur, Kolkata – 700107
E-mail : skb1@vsnl.com ; DebashisGanguly@gmail.com ; dippoulami@yahoo.com

ABSTRACT

The growth of high speed computer networks and that of the Internet, in particular, has hiked the ease of Information Communication as well as the fear of getting it snooped at the same time. So, Information Security is becoming an inseparable part of Computing and Communication. In realization of Information Security, Steganography plays an important role. In this paper, a heuristic approach for Information hiding in the form of images using Steganography is proposed keeping in mind two considerations - Size and Invisible Watermarking Scheme. Here, the Information as image is hidden behind another picture after zone wise replacement of pixels, where no essence of Information will be retrained within the final Morphed Image.

Keywords: *Security, Steganography, Invisible Watermarking, Morphed Image and communication.*

1. INTRODUCTION

Steganography is the art and science of writing hidden messages in such a way that no one apart from the sender and intended recipient even realizes there is a hidden message. Generally, in steganography, the actual information is not maintained in its original format and thereby it is converted into an alternative equivalent multimedia file like image, video or audio which in turn is being hidden within another object. This apparent message (known as cover text in usual terms) is sent through the network to the recipient, where the actual message is seperated from it.

The thought behind the algorithm, proposed in this paper, strikes out from the classical problem of designing a mosaic pattern of tiles in a rectangular floor which indeed is to be cost effective, i.e., to fill up a rectangular surface, one requires a minimum number of square tiles of maximum side length equals to the G.C.D of the specified height and breadth of that surface. So, in this paper the information taken in the form of KEY_PIC is extracted as zone of pixels in tile fashion and taking it as a key and after analyzing its contents, the same sized pixel zone of COVER_PIC is modified. Thus, this algorithm fascilitates the selection of pictures no matter of their size. So, no constraints of size is available as the fulfillment condition for the heuristic, where as, no matter how secured and better algorithms and approaches had been designed, most of them lack of the flexibility in selection of size of the files to be steganographed.

Another important feature, conceived in the encryption process of the images is Multilayered Security. Here, firstly the information picture, i.e., KEY_PIC is self-steganographed using the algorithm

mentioned in section-III.D, and after that this encrypted file is again hidden behind the COVER_PIC, which indeed acts as carrier of the information, using the same Steganographic algorithm. Thus if the ultimate object to be transferred gets snooped from the information channel, then also from it no way the information can be retrieved.

2. RELATED WORKS

The word "Steganography" is of Greek origin and means "covered or hidden writing". Its ancient origins can be traced back to 440 BC.

The majority of today's steganographic systems uses images as cover media because people often transmit digital pictures over email and other Internet communication.

Least significant bit (LSB) insertion is a common and simple approach to embed information in a cover file. In this method the LSB of a pixel is replaced with an M's bit. If we choose a 24-bit image as cover, we can store 3 bits in each pixel. To the human eye, the resulting stego image will look identical to the cover image [1].

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 [2].

Westfeld proposed F5 [3], an algorithm that does not overwrite LSB and preserves the stego image's statistical properties. In this algorithm, instead of replacing the least-significant bit of a DCT coefficient with message data, F5 decrements its absolute value in a process called matrix encoding.

3. OUR WORK

Before going into the details of the algorithm proposed here for invisible watermark [4] of the information behind the cover text, it is better to mention about the selection of the images and information kind which are to be steganographed. Generally, in usual ways of steganography secret information prone to be eavesdropped in the information channel e.g. signatures, secret conversation text, formulae or many others are chosen and converted into relevant, equivalent multimedia file format. Here in this paper, the algorithm basically implemented over normal bitmap image file, but it should be clarified that the same scheme can be operated over other file formats also. This selected and converted image file is here referred as KEY_PIC and the image behind which it is to be hidden is termed as COVER_PIC. The selection of neither the COVER_PIC nor the KEY_PIC is constrained by any size limit.

After having the pictures in hand following two conditions depicted in the algorithm, we have to pad the pictures with required white spaces, i.e., extra pixels. Thereafter, calculating the required G_C, G_K and m, the algorithm calculates the value returned by function SPN (G_C).

After proper padding and required analysis, according to the algorithm specified in section-III.C, COVER_PIC is altered zone by zone of pixels in squared tile fashion of size G_K, keeping zone of pixels of KEY_PIC as a key and extra m pixels along row and column wise is kept unchanged to signify a specified boundary of separation.

It is better to be confessed that the method for encryption can be personalized, i.e., can be selected according to the user ends. But, the authors specifically suggests this specialized scheme, proposed in this paper, as because here the information are no longer being merged or masked with another and instead of that keeping the KEY_PIC as a key the information in the carrier, i.e., the COVER_PIC is altered to obtain FINAL_PIC. Thus no essence of the actual information is retrained in the FINAL_PIC, whereas in usual methods of the mostly done bitwise merging; the information belongs in encrypted way directly merged into final object obtained.

3.1 PZS²IW_MAIN (KEY_PIC, COVER_PIC)

This is the main function in our algorithm. This function will be used in the sender side and will call

other modules of our algorithm like Padding and Encryption.

Arguments: This function will take KEY_PIC and COVER_PIC as argument and finally it will output the encrypted stego-image.

1. Obtain the width, length of the pixel matrix of the COVER_PIC (say R_C and C_C) and the KEY_PIC (say R_K and C_K).
2. Now check the GCD of R_C and C_C. If it is 1 then call ARRANGE (R_C, C_C) module of our algorithm to obtain the desired sizes for the COVER_PIC.
3. Next call the PAD (COVER_PIC, R_C , C_C) module to pad the COVER_PIC to make it of proper size.
4. Now check the GCD of R_K and C_K. If it is 1 then call ARRANGE (R_K, C_K) module of our algorithm to obtain the desired sizes for the KEY_PIC.
5. Next call the PAD (KEY_PIC, R_K, C_K) module to pad the KEY_PIC to make it of proper size.
6. Now compare the size of the COVER_PIC and KEY_PIC. If the size of the pixel matrix of the COVER_PIC is less than or equal to the same of the KEY_PIC then obtain the new row size and column size for the COVER_PIC.
7. After the above step, again call PAD (COVER_PIC, R_C , C_C) module to obtain the padded COVER_PIC.
8. Now, compare GCD (R_C, C_C) with GCD (R_K, C_K), if GCD (R_C, C_C) > = GCD (R_K, C_K) then go to next step, else arrange R_C, C_C and R_K, C_K in such a way that GCD (R_C, C_C) > = GCD (R_K, C_K) and the size of the pixel matrix of the COVER_PIC is greater than the same of the KEY_PIC to obtain new values and accordingly call module PAD (KEY_PIC AND/OR COVER_PIC, R_K, C_K).
9. Obtain the GCD of R_C and C_C as G_C, as well as, GCD of R_K, C_K, as G_K. Also obtain the sum of prime numbers in between 1 and G_K, as SPN using a function **δ** with argument G_K.
10. Now call PZS²IW_ENC (KEY_PIC, G_K, COVER_PIC, G_C, SPN) module to obtain the encrypted COVER_PIC as FINAL_PIC.
11. Send FINAL_PIC and COVER_PIC through the network.

3.2 ARRANGE (R, C)

This function is used in the algorithm to arrange the size of a picture in such a way that GCD (length, width)! = 0.

Arguments: This function will take length and width of any picture, as arguments and finally it will output the modified length or width or the both of the desired picture.

1. First, keep the column (C) as constant and increment the row (R) until the GCD of R and C becomes greater than one. Obtain then difference of the area due to modification of row size.
2. Next, keep the row (R) as constant and increment the column (C) until the GCD of R and C becomes greater than one. Obtain then difference of the area due to modification of column size.
3. Now either take the modified column size or the row size depending on the condition that the newly introduced area due to modification becomes less.
4. Return the modified data for row and column size.

3.3 PAD (PICTURE, R, C)

This function is used in the algorithm to pad an image to obtain an image of the desired size from the input image.

Arguments: This function will take the image, which has to be pad along with the desired length, and width of the resultant picture as arguments and finally it will output the padded picture.

1. Obtain the width, length of the pixel matrix of the PICTURE (say R_P and C_P).
2. Fill pixels with white color until R_P! = R and C_P! = C.
3. Return the PICTURE.

3.4 PZS²IW_ENC (PICTURE_1, G, PICTURE_2, G', SPN)

This function is used in the algorithm to encrypt an image with the help of either same image (self encryption) or another image to obtain an encrypted image of the desired size.

Arguments: This function will take the source image (PICTURE_1) which has to be encrypted, the associated image (PICTURE_2) with which encryption will be performed along with the calculated GCD of the source as well as of the associated image and the sum of primes from 1 to G as arguments and finally it will output the encrypted picture.

1. Obtain the difference between G' and G (say M) and the depth of the images (say DP).
2. Read the pixels from starting of the PICTURE_1.
3. If the value of the read pixel is 0 then replace the corresponding pixel of the PICTURE_2 with $2^{(8*DP)} - 1$ and read the next pixel.
4. If the value of the read pixel is $2^{(8*DP)} - 1$ then replace the corresponding pixel of the PICTURE_2 with 0 and read the next pixel and go to step 3.
5. If the value of the read pixel is greater than 0 and less than $2^{(8*DP)} - 1$ then do the following operations. Pixel'' = Pixel' + SPN where Pixel' is the corresponding pixel of the PICTURE_2.
6. Now if the calculated Pixel'' is greater than $2^{(8*DP)} - 1$ then calculate: Pixel''' = Pixel'' – $[2^{(8*DP)} - 1]$ and replace the corresponding pixel of PICTURE_2 with Pixel''' and read the next pixel and go to step 3.
7. Now if the condition of the step 6 is false then replace the corresponding pixel of PICTURE_2 with Pixel'' and read the next pixel and go to step 3.
8. Repeat the steps from 3 to 7 until zone of pixels having area G*G is not covered.
9. Now after the above the steps, insert a vertical and horizontal gap of pixels in PICTURE_2 having size M and then read the next pixel and go to step 3.
10. Repeat the above steps from 3 to 9 until the end of PICTURE_1 is reached.
11. Return PICTURE_2.

3.5 PZS²IW_ DEC (PICTURE_1, G, PICTURE_2, G', SPN)

This function is used in the algorithm to decrypt an image with the help the original image to obtain a hidden image.

Arguments: This function will take the source image (PICTURE_1) which has to be decrypted, the original image (PICTURE_2) with which decryption will be performed along with the calculated GCD of the encrypted (G) as well as of the original image (G') and the sum of primes from 1 to G as arguments and finally it will output the encrypted picture.

1. Obtain the difference between G' and G (say M) and the depth of the images (say DP). Take a normal white picture (say PICTURE_3) having size equal to either PICTURE_1 or PICTURE_2.
2. Read the pixels from starting of the PICTURE_1 (say Pixel), PICTURE_2 (say Pixel') and PICTURE_3 and PICTURE_3 (say Pixel'') one by one and compare Pixel and Pixel'.
3. If the value of Pixel is not equal to the value of Pixel' then go to the next step and else go on reading of pixels from both the pictures until the no. of times of reading overcomes M*M times and after that stops reading, go to step 8.
4. If the value of Pixel' is 0 then replace the corresponding pixel of PICTURE_3 with $2^{(8*DP)} - 1$ and read the next pixels and go to step 3.
5. If the value of Pixel' is $2^{(8*DP)} - 1$ then replace the corresponding pixel of PICTURE_3 with 0 and read the next pixels and go to step 3.
6. Now if the value of Pixel' is greater than SPN then calculate: Pixel1 = Pixel' - SPN and replace the corresponding Pixel of PICTURE_3 with Pixel1 and read the next pixels and go to step 3.

7. If the condition of the above step is false then calculate: Pixel1: = SPN – Pixel'; Pixel2: = $[2^{(8*DP)} – 1]$ – Pixel1; and then replace the corresponding pixel of PICTURE_3 with Pixel2 and read the next pixels and go to step 3.
8. Return PICTURE_3.

4. RESULT AND DISCUSSION

The stated Algorithm has got five distinct divisions, a. main function which calls next three sections; b. Arrange: to calculate desired row and column of pictures; c. Pad: to pad the images as per row-column given by previous sections; d. Encryption; e. Decryption.

4.1 Complexity analysis of the stated algorithm

For Arrange (Section III.B): Keeping row constant, column is incremented which takes C' – C = O(n). Keeping column constant, row is incremented which takes R' – R = O(n) . So, overall we get the total time complexity is O(2*n).

For Padding (Section III.C): To pad an image row wise, then we $O(n^2)$ is incurred. Again to pad an image column wise, complexity of $O(n^2)$ has incurred. So, overall time complexity becomes $O(2*n^2)$.

For Encryption and Decryption (Section III.D and Section III.E): For each row wise and column wise scan is being done. So, each one requires time of O(rowsize*columnsize) ≡ $O(n^2)$.

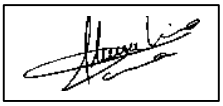
In this algorithm, operation is being done over a zone of pixels retrieved recursively from the actual pixel matrix. So, no need of remembering whole pixel matrix is required. Thus amount of space required to run this algorithm comes under O(log n) and thereby it becomes an inplace algorithm.

4.2 Test Results

Experimental results are shown in Figure-1 which, we have got after the execution of the algorithms PZS^2IW_MAIN, ARRANGE and PAD, consecutively, on Cover Image / Key Image.

The resultant output Image shown in Figure-2 after the execution of Encryption algorithm; in our case it is PZS^2IW_ENC Algorithm.

Cover Image [423 x 180]



Key Image [141 x 60]

Fig. 1:

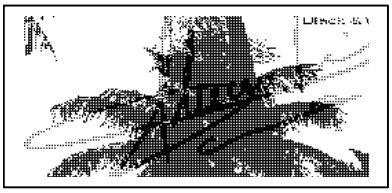


Fig. 2: Encrypted Image [423 x 180]

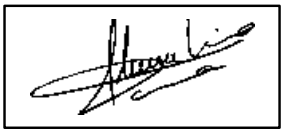


Fig. 3: Resultant Key Image [141 x 60]

The resultant output Image shown in Figure-3 after the execution of Encryption algorithm; in our case it is PZS^2IW_DEC Algorithm.

5. CONCLUSION

In this paper the major emphasis is given on the size and computational complexity, which incurs at the time of doing any operation in Steganographic approach. Here, due to selection of zone of pixels and operation on same has eliminated the constraint on size of the files chosen for Invisible Watermark. The COVER_PIC and KEY_PIC both are integral multiple of the zone of zone of pixels on which operation is being done. So due to implementation of zone wise mapping of pixels between COVER_PIC and KEY_PIC, keeping the lopping counters same, i.e., time complexity remaining

constant space complexity gets reduced as no need of remembering whole of pixel matrix of the pictures is present.

In this paper, the authors implemented the basic algorithm through bitmap pictures, but there is no such constraint over selection of file format, i.e., the same procedure can be realized through any of available multimedia file formats. It is thereby expected that any kind of future endeavor in this field will definitely route it a path to design a secure system using the proposed algorithm for both Internet and Mobile Communication Technology.

REFERENCES

1. Johnson, N. F. and Jajodia, S. (1998). Exploring steganography: Seeing the unseen. Computer, 31(2):26–34.
2. Niels Provos, Peter Honeyman, Hide and Seek: Introduction to Steganography (2003).
3. Westfeld, A. (2001). F5-a steganographic algorithm: High capacity despite better steganalysis. In Proc. 4th Int'l Workshop Information Hiding, pages 289–302.
4. Debashis Ganguly. Swarnendu Mukherjee, Mohit Mundhra (2006). Digital Watermarking: A New Approach. In Proc. 41st National conference, CSI'06, paper no. : 11.