Incorporation of Digital Watermarking through Embedding one Image within another Image

Sabyasachi Samanta
Haldia Institute of Technology,
Haldia, WB, INDIA,
E-mail id:
sabyasachi.smnt@gmail.com

Saurabh Dutta
Dr. B. C. Roy Engineering
College,
Durgapur, WB, INDIA
saurabh.dutta@bcrec.org

Gautam Sanyal
National Institute of
Technology, Durgapur,
WB, INDIA,
nitgsanyal@gmail.com

***Abstract*- In this paper, we have proposed an efficient watermarking algorithm for copyright protection of digital images. This algorithm is able to embed or hide digital image such as company's logo or trademarks into original image. We have altered all of the R, G and B values of the image which is taken as signature data, to an array of data bits. Then the data bits are encrypted to unlike pixel positions on the host image depending on key value and image size. As a result, we get a watermarked image. After that we have formed three different image shares using any two components of R, G and B of entire watermarked image. The key is also divided into three different logical blocks by digits. By combining any two blocks of key we have formed key shares and have assigned to image shares. Out of those three shares, only addition of any two is able to make the full image or key. At the decryption end, through appropriate arrangement of shares of key and image, make possible to retrieve hidden data bits from watermarked image and reform into its original content.**

Key words: Pixel, signature data, host image, visual cryptography, invisible digital watermarking

1. INTRODUCTION

Digital watermarking technique implying "copyright protection" or "content protection" is to protect host digital signal by embedding the data property like the company logo, copyright information, authentication or any indentify information into signal need protecting. In visual cryptographic technique, an image is broken into *n* shares, so that someone with *k* shares could decrypt the image, while any *k-1* shares revealed no information about the original image. A pixel with 32 bit color depth consists of α value, R (Red), G (Green) and B (Blue) value. α value is the value of opacity [8] [9] [11] [12] [14].

In this paper, we have proposed a technique to embed an image into another image. The image which has to embed into host image is referred as signature data. First the image size of the image is scanned and the width and height is converted into 8-bit binary equivalent. After that the data bits are stored into an encrypted array correspondingly. Then the values of R, G and B of each and every pixel are scanned by row and column wise. The entire values are converted into 8-bit binary equivalent and stored into the earlier encrypted array one by one as they are scanned. Now taking the key value, we have broken it into consecutive three-digit two figures by modulus division starting from most significant digit to least significant digit. In our proposed technique, the embedding process stared from this pixel value of the host image. Here the initial pixel position differs from process to process and truly depends on the value of secret key. Also depending on the size of the both image (the host image and the image which to be embed i.e. signature data) the data bits from encrypted array are embedded to the entire image. Finally, we get a watermarked image. Then to make three different shares using any two components of R, G and B of each and every pixel about the entire image is scanned. Taking any two values and the third value as 0, for the entire pixels formulate three different shares. From the viewpoint of key, we have used a key with 6-digits. Also the key is divided into three subsequent blocks of neighboring digits as image. As before, we have created key shares using the blocks or digits of key. For each and every shares of image different key shares are assigned. Out of those three communicable shares when only minimum two communicable shares of image and key, came together then we will be able to get back the original image. In our work, we have targeted any one bit of last four significant bit of each R, G and B of any selected pixel positions starting from nonlinear pixel positions. The replacement of all the data bits have done in selected pixels about the entire image using the private key cryptography technique [3] [5] [7] [8] [10] [13].

Example: An image (10 x 10) will form an array with 2416 ((2*8) + ((10*10)*24) bits of stream (first 16-bits for width and height). An image with 800 X 600 dimension has 2, 40,000 pixels. In our work, only we have altered any one bit of last two significant bits. If any bit generated from image become same to the targeted bit of image then there will be no change i.e. it will produce the same.

Section 2 represents the scheme followed in encryption technique. Section 3 represents an implementation of the technique. Section 4 gives you an idea about the experimental results. Section 5 is an analytical discussion on the technique. Section 6 draws a conclusion.

II. THE SCHEME

This section represents a description of the actual scheme used during "Incorporation of Digital Watermarking through Embedding one Image within another Image" technique. Section 2.1 describes the encryption technique using five algorithms 2.1.1, 2.1.2, 2.1.3, 2.1.4 & 2.1.5 while section 2.2 describes the decryption technique using algorithm 2.2.1 [1] [2] [3] [4].

2.1 Encryption of data bits about the image

2.1.1 Create an array of encrypted data from signature data

Step I: Take the image or signature data which to be embedding into the host image. Compute the width (w_{simg}) and height (h_{simg}).

Step II: Convert the values into 8-bit binary equivalent.

Step III: Store the data bits from the width (w_{simg}) and height (h_{simg}) and pixel (p_{simg}[][]) values into an encrypted array *earr[bit]* as LSB (Least Significant Bit) to earr[1] and MSB (Most Significant Bit) to earr[8] respectively.

Step IV: Scan R, G and B values of every pixels and convert into 8-bit binary equivalent.

Step V: Lay up the binary values to earr[] as LSB to earr[1+(i*8)] and MSB to earr[8+i*8] of α, R, G and B respectively.

Step VI: Repeat *Step IV* to *Step V* for the entire pixels (p_{simg}[i][j]) for i=1 to w_{simg} and j=1 to h_{simg}.

Step VII: Stop.

2.1.2 Formation of subset of key taking key input from keyboard

Step I: Take a 6-digit key ($K_{[0]}$) input (decimal number from 0 to 9).

Step II: To make shares split the key ($K_{[0]}$) to singular decimal digits by modulo division.

Step III: Repeat *Step II* for i=1 to 6 and store Least Significant Digit (LSD) to an array element keydgt[1] and Most Significant Digit (MSD) to keydgt[6] respectively.

Step IV: Taking the digits (i.e. keydgt[i]) and multiplying with power of the position, generate subset of keys.

a) Taking keydgt[6] as MSD and keydgt[3] as LSD form the key $K_{[1]}$.

b) Taking keydgt[4] as MSD and keydgt[1] as LSD form the key $K_{[2]}$.

c) Taking keydgt[2] (as MSD) to keydgt[1] and from keydgt[6] to keydgt[5](as LSD) form the key $K_{[3]}$.

Step V: Stop.

2.1.3 Selection of the pixel positions using the key

Step I: Take the value of *bit* from array e*arr[bit]* to calculate total number of pixels(p) is required (as three following data bit replaced in R, G and B of every pixel) about the image. So, p= (ceil (bit /3)).

*Step II***:** Take a 6-digit key input from key board.

Step III: Store the value as numeric and store it to an array *arrxy[p]*.

Step IV: The key have broken that into two numbers of consecutive three digits by modulus division.

Step V: Take most three most significant digits to *arrx[p],* next three least digits to array *arry[p]*.

Step VI: Read the width (w_{himg}) and height (h_{himg}) of the host image*.* Divide the total number of pixels of embedded image to the total number of host image. Find out the pixel difference (p_{diff}) i.e. the selected pixel position of host image.

$$p_{diff}=ceil((w_{himg}* h_{himg})/p)$$

Step VII: Take the value and select the pixel positions by adding with previous value to host image.

Step VIII: Repeat *Step V* to *Step VII* up to end of the file.

Step IX: Stop.

2.1.3 Replacement of array elements with R, G & B values of pixels

Step I: Take the value of width (w_{himg}) and height (h_{himg}) of host image.

Step II: Set x=*arrx[p]* and *y=arry[p]*.

Step III: To select the pixel position into image, compare the value of x and y with the value of w and h (where addressable pixel position is (0, 0) to (w-1, h-1)).

a) If (x >(w-1)) or (y >(h-1)) then

Set P (x, y) = P (0+(x %(w-1)), (0 +(y %(h-1)))

Otherwise Set P (x, y) = (x, y).

Step IV: To select the bit position (b) of selected pixel we have just multiplied row value to column value (z=x+y).

i) If (z%4=0) then b=LSB

ii) If (z%4=1) then b=2nd LSB

iii) If (z%4=2) then b=3rd LSB

Otherwise b=4th LSB of each R, G and B of a pixel.

Step V: To replace the array elements with the selected bit position of selected pixel and to reform as a pixel

a) After reading the values of R, G & B convert each to its equivalent 8-bit binary values.

b) Replace subsequent element of earr[bit] by following *Step III to Step V*.

c) Taking values of R, G & B switch it to the pixel value and place it to its position of the image (taking α value as before).

Step VI: For replacing the array element to pixels using the above mentioned process starting from the 0^{th} element up to the end of the array.

A) If bit%3 = 0

Go to *Step VIII.*

B) If bit%3 = 1

for 0^{th} element to $(bit-1)^{th}$ element of the array repeat *Step VI (A).*For $(bit)^{th}$ element to R, value for G and B will be remain same. And go to S*tep VII.*

C) If bit%3=2

for 0^{th} element to $(bit-2)^{th}$ element of the array repeat *Step VI (A).*For $(bit-1)^{th}$ element to R, $(bit)^{th}$ to G and B will be remain same. And go to S*tep VII.*

Step VII: Repeat *Step II to Step VI* for i=1 to p.

Step VIII: Stop**.**

2.1.5 Creation of logical region about the image

Step I: Calculate the width (w_{wimg}) and height (h_{wimg}) of the watermarked image.

Step II: Read the R, G and B value of each and every pixel of entire image. Store the values in a file as an array element.

Step III: Taking any two component of R, G and B from file create three different image shares as

For $ImgSh_{[1]}$ ={R=0, G= as file and B= as file }

For $ImgSh_{[2]}$={ R=as file , G= 0 and B=as file }

For $ImgSh_{[3]}$={ R=as file , G= as file and B=0}.

Step IV: Stop.

2.2 Decryption of the data bits from the image

2.2.1 Reformation of signature image from the watermarked image

Step I: Take any two key input as {($K_{[1]}$, $K_{[2]}$), ($K_{[2]}$, $K_{[3]}$), ($K_{[3]}$, $K_{[1]}$)}. Taking any one common part of two key shares form the key ($K_{[0]}$) as it was build *Step II to Step IV of algorithm 2.1.2*.

Step II: Take any two shares of image as {($ImgSh_{[1],}$ $ImgSh_{[2]}$), ($ImgSh_{[2],}$ $ImgSh_{[3]}$), ($ImgSh_{[3],}$ $ImgSh_{[1]}$)}. Read the R, G and B component values from any two shares and store it to file. Taking any one common value of R, G, and B from the files create the original watermarked image.

Step III: To get the pixel and bit positions in R, G and B of selected pixels go through *Step I* to *Step VIII* of *Algorithm 2.1.3* and *Step I* to *Step VII Algorithm 2.1.4.*

Step IV: Retrieving the encrypted bits from the selected bit positions of selected pixels store it to decrypted array from darrlen[1] to darrlen[bit] respectively.

Step V: To get the width (w_{simg}) and height (h_{simg}) repeat *Step II* to *Step IV* for i= 1 to 6 times (as every pixel contain three data bits).

Step VI: Taking data bits of darrlen [1] as LSB and darrlen [8] as MSB calculate the width (w_{simg}) and from darrlen [9] to darrlen [16] as before calculate height (h_{simg}).

Step VII: Retrieving the encrypted bits from the selected pixel and bit positions, store it to decrypted character array from darr[17] to darr[bit] respectively (where *bit* is the array length from characters and length).

Step VIII: Taking data values from the decrypted array darr[], LSB as darr[8*i+1] and MSB as darr[8*(i+1))] respectively.

Step IX: Convert it into decimal equivalent to reform the pixel element (p_s) of signature data by repeating *Step VIII* for i=1 to p_v=(where p_v=4*w_{simg}* h_{simg}) and store it into an array $p_s[p_v]$.

Step X: Take the data values from the array (p_s[]) and form a pixel by substituting the values as R, G and B. And repeat for i=1 to 4.

Step XI: Repeat *Step X* for j= 1 to w_{simg} and k= 1 to h_{simg} to form a complete image.

Step XII: Stop.

III. AN IMPLEMENTATION

Let the image to encrypt is in Fig. 4.1(a).

The size of image =width x height

= 10 x 10 (w_{simg} x h_{simg}).

In the Table 3.1 characters with their encrypted binary equivalent is defined.

TABLE 3.1: PIXEL VALUES WITH BINARY EQUIVALENT

Pixel Address	Pixel Elements	Pixel values in Integer
(1,1)	{R,G&B}	{228,132,228}
:	:	:
:	:	:
(10,10)	{R,G&B}	{162,070,080}

Initially the bits from width, height and then from signature image (R, G and B) is being store to array earr[bit] respectively.

The size of host image= 600 X 800(w_{himg} x h_{himg}).

Number of effected pixel required for character (p) = [ceil (2416/3)] =806.

In the Table 3.2 the image shares with RGB components and corresponding assigned key shares are given (as described in Algorithm 2.1.2 and Algorithm 2.1.5).

TABLE 3.2: POSITIONS OF ARRAY ELEMENTS ABOUT THE IMAGE

Image Share	Image Components	Key Digits
Image	RGB as in image	K
Share1	R=0,G&B as in image	K[1]
Share2	G=0,R&B as in image	K[2]
Share3	B=0,R&G as in image	K[3]

In the Table 3.3, how the array elements are replaced with R, G and B values in selected nonlinear pixels and bit position about the image is described (as described in Algorithm 2.1.3 and Algorithm 2.1.4).

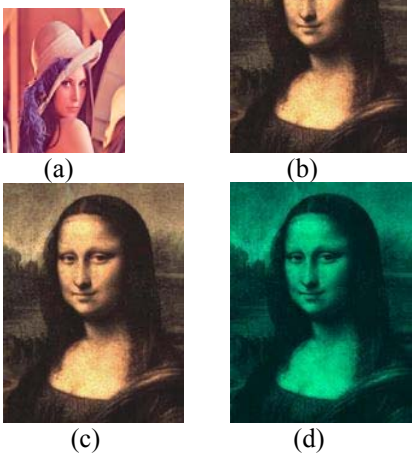
TABLE 3.3: REPLACEMENT OF BITS ABOUT IMAGE

Key,i	Value of pixel P(x,y)	Bit position b= Z%4	Array data to replace
637589,1	P(077,589)	1st LSB	earr[1] earr[2] earr[3]
:	:	:	:
637589,1072	P(259,192)	2nd LSB	earr[3216] As before As before

In this way, we can create an indistinguishable watermarked image embedding the entire message. Afterward, we are able to transmit the encrypted watermarked image through any communication channel. Applying the decryption technique as described in Algorithm 2.2 also we will be able to get back the encrypted image from that watermarked image at the decryption end.

IV. EXPERIMENTAL RESULT

An example result is shown in Figure 4.1. Fig. 4.1(a) is the image which to be embed i.e. signature data. Fig. 4.1(b) is the host image. It is one of the popular images of MONALISA.



(a) (b)

(c) (d)

(e) (f)

Figure 4.1: (a) the image to be embedding (b) the host image, (c) the watermarked image, (d) (e) and (f) image shares with absent of R, G and B components respectively.

V. ANALYSIS

Here, initially we have produced a watermarked image and then the image shares using any two components of three(R, G and B). Binary values generated form the signature data replaced nonlinear pixel positions of the host image. Here the α value is not taken at the time of embedding. Anybody may take the α value. If the α value of the entire image become same then there is no requirement to add the binary equivalent of α value each and every time to the encrypted array. On a single occasion we may add the α value to the array for entire image by using any algorithm or compression technique. So forth the effected number of pixels will also be less. As we have started the initial pixel position depending on key so it differs from process to process and truly depends on the value of secret key. Here we have targeted only the R, G and B values of host image. If we also target the α value of targeted pixels then also the number of targeted pixels will be less. Also alphanumeric characters may include for key, not only using numerical digits. As bits are replaced any one bit of least four LSB position of each R, G and B, the change of color of the targeted pixels will be less and so forth it becomes invisible to human eye. The number of targeted pixels proportionally varies with the size of signature image. If the host image size becomes large and size of the signature data becomes less then it will be quite harder to differentiate the encrypted image from the original image [2] [6] [9].

VI. CONCLUSION

This paper represents a robust hybrid watermarking technique for embedding image watermark into carrier image. Not only taking the text, we also embed any image into another image. If the signature image is of small in size and the carrier image be few times larger than the previous we may produce invisible watermarked image. Here we have used private key cryptographic technique to place the data bits (from both image and size of image) in unlike pixel and bit positions starting from nonlinear pixel position depending on key about the entire image.

After that we have generated image shares from watermarked image and key shares which are assigned to the different image shares. In the proposed method, the embedding scheme takes the spatial information into consideration. This increases the invisibility of watermark into watermark image. Moreover, it produces the similar image to see in naked eye at the time of watermarking using this method. If the key become unknown to anybody who wants to attack the information, we think, it will be quite impossible to him or her to find out the information from the watermarked image.

REFERENCES

[1] Sabyasachi Samanta, Saurabh Dutta, Goutam Sanyal, "An Enhancement of Security of Image using Permutation of RGB-Components", "3rd International Conference on Conference on Electronics Computer Technology (ICECT 2011) ", 8-10 April, pp. v2-404-v2-408

[2] Sabyasachi Samanta, Saurabh Dutta, Goutam Sanyal, "An Enhancement of Security on Image Applying Asymmetric Key Algorithm", International Journal of Computer Applications (0975 – 8887), Volume 25– No.5, July 2011, pp. 19-23

[3] Souvik Bhattacharyya, Gautam Sanyal "A Data Hiding Model with High Security Features Combining Finite State Machines and PMM method", International Journal of Electrical and Computer Engineering 5:2 2010, pp. 78-85

[4] Souvik Bhattacharyya, Gautam Sanyal, "Data Hiding in Images in Discrete Wavelet Domain Using PMM",World Academy of Science, Engineering & Technology 68 2010,pp. 597-605

[5] P. S. Revenkar, Anisa Anjum, W .Z.Gandhare "Secure Iris Authentication Using Visual Cryptography", International Journal of Computer Science and Information Security, Vol. 7, No.3, 2010, ISSN 1947-5500, pp.-217-221

[6] Jianmin Xie, Qin Qin, "Research on the Wavelet-based Adaptive Digital Watermarking Algorithm", 2009 Third International Symposium on Intelligent Information Technology Application Workshops, pp 383-386

[7] Thi Hoang Ngan Le, Kim Hung Nguyen, Hoai Bac Le "Literature Survey on Image Watermarking Tools, Watermark Attacks, and Benchmarking Tools", 2010 Second International Conferences on Advances in Multimedia,pp.67-73.

[8] M. Kameswara Rao, Sushma Yalamanchili, "Copyright Protection of Gray Scale Images by Watermarking Technique Using (N, N) Secret Sharing Scheme", Journal of Emerging Technologies in Web Intelligence, Vol. 2, No. 2, May 2010, pp. 101-105

[9] V. Padmanabha, Reddy, Dr. S. Varadarajan, "An Effective Wavelet-Based Watermarking Scheme Using Human Visual System for Protecting Copyrights of Digital Images", International Journal of Computer and Electrical Engineering, Vol. 2, No. 1, February, 2010, pp. 32-40

[10] G. Rosline Nesa Kumari , B. Vijaya Kumar, L. umalatha, Dr. V. V. Krishna, "Secure and Robust Digital Watermarking on Grey Level Images", International Journal of Advanced Science and Technology, Vol. 11, October, 2009, pp. 1-8

[11] Rosziati Ibrahim, Teoh Suk Kuan, "Steganography Algorithm to Hide Secret Message inside an Image", Computer Technology and Application 2, February 25, 2011, pp. 102-108

[12] Bin Li, Junhui He, Jiwu Huang, Yun Qing Shi, "A Survey on Image Steganography and Steganalysis", Journal of Information Hiding and Multimedia Signal Processing, ISSN 2073-4212, Volume 2, Number 2, April 2011, pp. 142-172

[13] Sara Natanj, Seyed Reza Taghizadeh, "Current Steganography Approaches: A survey", International Journal of Advance Research in Computer Science and Software Engineering ISSN: 2277 128X, Vol. 1, Issue 1, Dec. 2011, pp. 1-8

[14] Atul Kahate, "Cryptography and Network Security", 2nd Edition, THM, New Delhi

AUTHORS PROFILE

Sabyasachi Samanta is working as Assistant Professor at Dept. of IT,Haldia Institute of Technology Haldia, WB, India. He has received M.Tech Degree in IT and currently pursuing Ph.D at National Institute of Technology, Durgapur, WB, India. His main research interest includes watermarking, steganography and cryptography.

Saurabh Dutta is a professor in Dr. B. C. Roy Engineering College. He holds a Ph. D. Degree in Coputer Science. His research domain is information security and cryptology.

Gautam Sanyal is a member of the IEEE. He has received his B.E and M.Tech degree from National Institute of Technology (NIT), Durgapur, India. He has received Ph.D. (Engg.) from Jadavpur University, Kolkata, India, in the area of Robot Vision. He possesses an experience of more than 25 years in the field of teaching and research. He has published nearly 68 papers in International and National Journals / Conferences. Three Ph.Ds (Engg) have already been awarded under his guidance. At present he is guiding six Ph.Ds scholars in the field of steganography, Cellular Network, High Performance Computing and Computer Vision. He has guided over 10 PG and 100 UG thesis. His research interests include Natural Language Processing, Stochastic modeling of network traffic, High Performance Computing, Computer Vision. He is presently working as a Professor in the department of Computer Science and Engineering and also holding the post of Dean (Students' Welfare) at National Institute of Technology, Durgapur, India.