International Journal of Computer and Communication Technology

Volume 3 | Issue 1 Article 13

January 2012

Implementation of Invisible Digital Watermarking on Image using Permutation of Key and Image Shares

Sabyasachi Samanta
Haldia Institute of Technology Haldia, WB, INDIA, sabyasachi.smnt@gmail.com

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

Implementation of Invisible Digital Watermarking on Image using Permutation of Key and Image Shares

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
E-mail id: saurabh.dutta@bcrec.org

Abstract-- **Data bits from text are embedded to some suitable nonlinear pixel and bit positions about the entire image through the key. After that, we have divided that image into three logical regions vertically. Also, the key is divided into three different blocks by digits. Taking adjacent two of three logical regions we have formed three shares for both image and key. Each share of key is assigned to each and every shares of image. Out of those three shares only addition of any two is able to make the full image. Also through appropriate arrangement of shares of keys we can retrieve that hidden data bits from watermarked image and reform into its original content.**

Key words: *pixel, invisible digital watermarking, visual cryptography, symmetric key, nonlinear function.*

1. INTRODUCTION

Digital Watermarking describes the way or technology by which anybody can hide information, for example a number or text, in digital media, such as images, video or audio. In visual cryptographic technique, an image is broken into *n* shares so that only someone with all *n* shares could decrypt the image, while any *n-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. If α is 00000000, the image will be fully transparent. Each of three(R, G & B) 8-bit blocks can range from 00000000 to 11111111(0 to 255) [1][2][6].

$$\underbrace{11111111}_{\alpha}\quad \underbrace{11011100}_{R}\quad \underbrace{10101011}_{G}\quad \underbrace{11110000}_{B}$$

Figure 1.1 Bit representation of a 32-bit RGB pixel

In this paper, we have proposed a technique to embed a message about the entire image. The text taken from the keyboard or special characters encoded into its ASCII-8 (American Standard Code for Information Interchange) binary equivalent. The length of the text is also converted into its 8 bit binary equivalent. Let in a banking system, three persons hold three different keys. Out of those three keys only a proper combination of two keys can open the system. Here also we have applied the same concept to implement the cryptographic technique. The image is considered as a cylindrical drum horizontally (or vertically). Here, the image is divided into 3 logical regions vertically. Out of those three taking two neighboring regions in clockwise direction, we have assembled three shares. From the standpoint of key, we have used a key with 6 digits. In addition, the key is divided into three subsequent blocks of neighboring digits as image. As before, we have created three sub keys using the blocks of key. For each and every shares of image, different private keys are assigned. Out of those three communicable shares, when only two communicable shares came together then only we will be able to get back the original message. Also out of three sub keys, when only two sub keys will be known, the decryption of original message will be possible. Using a 6 digit maximum number only 51 pixels is truly addressable. As we have targeted any one of four least significant of each R, G and B, we can only implant maximum 18 characters using the key. The replacement of all the bits have been done in nonlinear pixel and bit positions, in any one of last four significant bit of R, G and B at selected pixels about an image using nonlinear function and the private key cryptography technique, taking the α value as 255 or as in the original image[3] [5] [7].

Example: A text with 9-characters will form an array with 80 (8+9*8) bits of stream (first 8-bits for length). An image with 800 X 600 dimension has 2, 40,000 pixels. In our work, only we have altered any one bit of last four significant bit of each R, G & B. If any bit generated from text become same to the targeted bit of image then there will be no change i.e. it will produce the same to original image[2] [5].

Section 2 represents the scheme followed in the encryption technique**.** Section 3 represents an implementation of the technique. Section 4 is an analytical discussion on the technique. Section 5 draws a conclusion.

2. THE SCHEME

This section represents a description of the actual scheme used during “Implementation of Invisible Digital Watermarking on Image using Permutation of Key and Image Shares” 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 [2] [3] [4].

2.1 Encryption of data bits about the image

2.1.1 Create an array using binary values for length and characters

Step I: Take input from keyboard or special characters.
Step II: Calculate the string length (chlen) from input.
Step III: Convert the length (chlen) into its 8 bit binary equivalent. Store that data bits to earr[bit] as LSB (Least Significant Bit) to earr[1] and MSB (Most Significant Bit) to earr[8] respectively.
Step IV: Convert each character of text into its ASCII-8 binary equivalent.
Step V: Store the binary values encoded from characters to earr [ ] as LSB to earr[1+(i*8)] and MSB to earr[8+i*8].
Step VI: Repeat *Step III* to *Step IV* for i=0 to (N-1).
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 except more consecutive 0’s and characters).
Step II: 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 $\boldsymbol{K_{[1]}}$.
b) Taking keydgt[4] as MSD and keydgt[1] as LSD form the key $\boldsymbol{K_{[1]}}$.
c) Taking keydgt[2] (as MSD) to keydgt[1] and from keydgt[6] to keydgt[5](as LSD) form the key $\boldsymbol{K_{[3]}}$.

Step IV: Stop.

2.1.3 Selection of the nonlinear pixel positions from 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 & B of every pixel)in any image. So, p= (ceil (bit /3)).
Step II Take the key (K) and calculate the value of function

$F(x, y) = K^p$ [i.e. pow (k, p)].

Step III: Store the exponential long double values into file one by one.
Step IV: Repeat *Step III* to *Step IV* for i= (1 to p) and go to *Step V*.
Step V: Read the values as character up to “e” of the every line of the file and store it to another file with out taking the point [.].
Step VI: Modify the value as numeric and store it to an array *arrxyz[p]*.
Step VII: Take most three significant digit to *arrx[p],* next three digits to array *arry [p]* and last significant digit to *arrz[p]*.
Step VIII: Repeat *Step V* to *Step VII* up to end of the file.
Step IX: Stop.

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

Step I: Calculate the width (w) and height (h) of the 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 i.e. with which bit the array data will be replaced. Set *z =arrz[p]*.

i) If (z%4=0) then b=LSB
ii) If (z%4=1) then b=2^{nd} LSB
iii) If (z%4=2) then b=3^{rd} LSB
Otherwise b=4^{th} LSB of each R, G & B of a pixel.

Step V: Verify the pixel or bit positions which previously have used or not about the image.

a) If ((P(x, y)= (P(x, y)) || P (x, y)= P (x++, y++)) && (b=b++])then
Set P ((x, y), b) =P (0, h*)* and b as *Step IV*.
Repeat *Step V (a) for j=1 to p;*
Repeat *Step V (a) for k=j to p.*
Go to *Step VI.*

Step VI: 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 VII: 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 VII (A).*For $(bit)^{th}$ element to R, value for G and B will be remain same. And go to S*tep VIII.*

C) If bit%3=2

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

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

Step IX: Stop**.**

2.1.5 Creation of logical region about the image

Step I: Calculate the width (W) and height (H) of the image.

Step II: Break the image into three different regions and set the minimum and maximum pixel address as $ImgR_{[1]}$, $ImgR_{[2]}$ and $ImgR_{[3]}$.

Set the starting and ending pixel position as *$ImgRS_{[]}$* and *$ImgRE_{[]}$* respectively for every region.

For the entire image $ImgR_{[0]}$

$ImgRS_{[0]}$[w, h]=(0,0) and $ImgRE_{[0]}$=(w, h).

For $ImgR_{[1]}$

$ImgRS_{[1]}$[w, h] = (0, 0) and $ImgRE_{[1]}$ [w, h] = (floor(w/3), floor(h)).

For $ImgR_{[2]}$

$ImgRS_{[2]}$ [w, h] = (floor(w/3+1),0) and $ImgRE_{[2]}$= [w, h] =(floor(2*w/3), floor h).

For $ImgR_{[3]}$

$ImgRS_{[3]}$ [w, h] = (floor(2*w/3+1),0) and $ImgRE_{[3]}$ [w, h] = (w, h).

Step III: Create three different shares using the image regions $ImgR_{[\,]}$.

Set the starting and ending pixel position for image share ($ImgSh_{[]}$)as

For $ImgSh_{[1]}$={$ImgRS_{[1]}$,$ImgRS_{[2]}$}
For $ImgSh_{[2]}$={ $ImgRS_{[2]}$, $ImgRS_{[3]}$}
For $ImgSh_{[3]}$={ $ImgRS_{[3]}$, $ImgRS_{[1]}$}.

Step IV: Stop.

2.2 Decryption of the data bits from the image

2.2.1 Regain of replaced bits from the watermarked image and formation of original content

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).

Step II: Take any two shares of image as {($ImgSh_{[1]}$, $ImgSh_{[2]}$), ($ImgSh_{[2]}$, $ImgSh_{[3]}$), ($ImgSh_{[3]}$, $ImgSh_{[1]}$)}. To get the full original image combine any two shares of image removing the common areas.

Step III: To get the pixel and bit position in R, G & B of selected pixels go through *Step I* to *Step VIII* of *Algorithm 2.1.3* and *Step I* to *Step VIII* of *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 length repea*t Step II* to *Step IV* for i= 1 to 3 times (as every pixel contain three data bits) taking the key ($K_{[0]}$) and $ImgR_{[0]}$.

Step VI: Taking data bits of darrlen [1] as LSB and darrlen [8] as MSB calculate the length (chlen) of message.

Step VII: Repeat *Step VIII* for i=1 to 4.

Step VIII: Retrieving the encrypted bits from the selected bit position of selected pixels and of selected region, store it to decrypted character array from darr[1] to darr[bit] respectively (where *bit* is the array length from characters).

Step IX: Taking data values from the decrypted array darr[ ], LSB as darr[8*i+1] and MSB as darr[8*(i+1))] respectively, convert to its equivalent ASCII-8 character. And store the character to an array msg[len].

Step X: Finally place the characters one by one from the array msg[len] and assemble the original message**.**

Step XI: Stop.

2. AN IMPLEMENTATION

Let the message to be encrypt is NONLINEAR.
So the length of the message

=09(Decimal equivalent)
=00001001(8 Bit Binary equivalent)
In the Table 3.1 characters with their ASCII-8 bit binary equivalent is defined.

Table3.1: Characters with binary equivalent

Character	Decimal Equivqlent	8-Bit Binary Equivalent
N	078	01001110
O	079	01001111
N	078	01001110
L	076	01001100
I	073	01001001
N	078	01001110
E	069	01000101
A	065	01000001
R	082	01010010

First store the bits for length to the array earr[bit] and then store the bits from text respectively as,

bits for length	*bits for character*	*bits for character*
earr[1]=1	earr[9]=0	earr[73]=0
: :	: :	: :
r[8]=0	earr[16]=0	earr[80]=0

Figure 3.2: Data bits in encrypted array

Let key (K) =637589.
We the key set as
$K_{[1]}=\{K_{[a]}, K_{[b]}\}=6375;$
$K_{[2]}=\{K_{[b]}, K_{[c]}\}=7589;$
$K_{[3]}=\{K_{[c]}, K_{[a]}\}=8963;$
The image size= 800 X 600(w x h).
Number of effected pixel required for character (p) = [ceil (80/3)] =27.
In the Figure-3.3 and Table-3.3, the image regions and corresponding assigned key shares are described (as described in Algorithm 2.1.2 & 2.1.3).

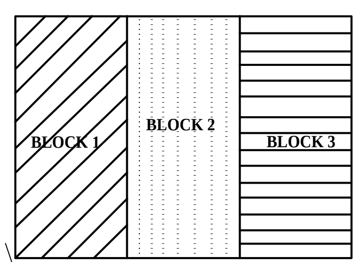

Image with Blocks

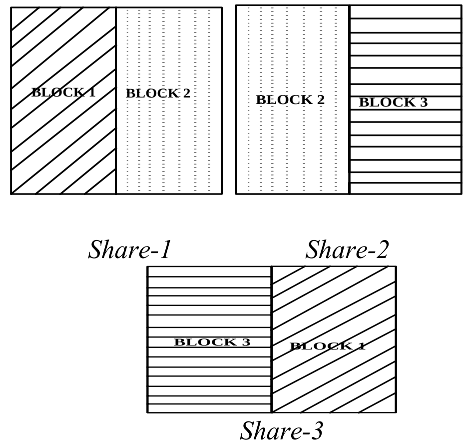

Figure 3.3: Image with shares

Table 3.3: Positions of array elements about the image

Image Region (ImgR)	ImgRS[ ]	ImgRE[ ]	ImgShare	Key digits
Image ImgR[1] ImgR[2] ImgR[3]	(1,1) (1,1) (267,1) (533,1)	(800,600) (266,600) (532,600) (800,600)	 {(1,1)to(532,600) } {266,600)to(800,600} {532,600)to(800,600) &(1,1)to(266,600)}	K **K[1]** K[2] K[3]

In the Table-3.4, how the array elements are replaced with R, G & B values in selected nonlinear pixels and bit position about the image is described (as described in Algorithm 2.1.4 & 2.1.5).

Table 3.4: Replacement of bits about image

Key(K),i	Value	Value of pixel P(x,y)	Bit position b= Z%4	Array data to replace
637589,1	6.375890e+05	P(037,589)	1st LSB	earr[1] earr[2] earr[3]
:	:	:	:	:
637589,5	1.053669e+29	P(105,366)	2nd LSB	earr[13] earr[14] earr[15]
:	:	:	:	:
637589,27	5.279612e+156	P(527,161)	3 rdLSB	earr[79] earr[80] B as same

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 2.2.1 also we will be able to get back the encrypted message from that watermarked image at the decryption end.

4. ANALYSIS

Here we have divided the image into shares vertically. Any one make shares horizontally also. We have not used any compression and/or encryption technique before the creation of array (earr[]). Any body can employ the compression and/or encryption technique(s) at the time of creation of array (earr[ ]). In that case, the length of array might be less and the strength of encryption possibly will be higher than the present. In addition the number of affected pixel will also be fewer than now. Using 6-digit key we are able to replace the data bits of maximum 21 characters. If any body wants to use variable length key, it's also possible (as *message_lenght ∞ (1 / number_of_keydigit)).* Also alphanumeric characters may include only without using numerical digits. Furthermore any body may generate another subset of keys from the key (K) as he or she wants and may divide into more shares vertically or horizontally. Binary values generated form the textual information replaced in different nonlinear places in the image. As bits are placed any one bit in lower four bits of each R, G & B, the change of color of the modified pixels are invisible to human eye. If size of the text is less then number of pixels affected from the text will be less. At that time it will be harder to differentiate the encrypted image from the original image. If the image size is large and number of pixels is less then it will also be harder to differentiate the encrypted image from the original image [6] [7] [9].

5. CONCLUSION

We have used the private key and set of key using the digits of key and nonlinear function technique (depending on key and bit length) to select both the pixel positions and bit position (of each R, G & B pixel) where the data bits from length and message will be hidden inside the entire image and logical regions of the image respectively. In our present work, only appropriate combination of two shares for both the image and key can produce the full image and original key. 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, Shyamalendu Kandar, Saurabh Dutta "Implementing Invisible Digital Watermarking on Image" in 'The Bulletin of Engineering and Science' ISSN: 09747176 September, 2008 VOL.3 No.2 pp. 79-82.

[2] Sabyasachi Samanta, Saurabh Dutta "Implementation of Invisible Digital Watermarking on Image Nonlinearly of Arithmetically Compressed Data" in 'IJCSNS International Journal of Computer Science and Network Security, Journal ISSN: 1738-7906 VOL.10 No.4, April 2010 pp.261-266.

[3] Sabyasachi Samanta, Saurabh Dutta "Implementation of Invisible Digital Watermarking on Image Nonlinearly Encrypted with Galois Field (GF-256)" in "2010 International Conference on Informatics, Cybernetics, and Computer Applications (ICICCA2010)" July 19-21,2010 pp.26-30.

[4] Sabyasachi Samanta, Saurabh Dutta "Implementation of Invisible Digital Watermarking on Image Nonlinearly " accepted by "International Conference on Computer Applications, 2010 (ICCA2010)" December 24-27, 2010.

[5] Pranam Paul, Saurabh Dutta, Anup K. Bhattacharjee, "An Approach to ensure Security through Bit-level Encryption with Possible Lossless Compression" IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.2, February 2008, pp. 291-299.

[6] Moni Noar, Adi Shamir "Visual cryptography" Department of Applied Math and Computer Science Wiazmann Institute, Rehovot.

[7] Mahmoud A. Hassan, and Mohammed A. Khalili "Self Watermarking based on Visual Cryptography" World Academy of Science, Engineering and Technology 8,, 2005.

[8] Azzam Sleit, Adel Abusitta "A Visual Cryptography Based Watermark Technology for Individual and Group Images" Systemics, Cybernetics and Informatics Volume 5, Number 2 pp.-24-32.

[9] Ashok Banerjee, Ananda Mohan Ghosh, "Multimedia Technology" TMH, New Delhi.