An Enhancement of Security on Image Applying Asymmetric Key Algorithm

Sabyasachi Samanta
Haldia Institute of Technology
Haldia, WB, India

Saurabh Dutta
Dr. B. C. Roy Engineering
College, Durgapur, WB, India

Gautam Sanyal
National Institute of Technology,
Durgapur, WB, India

ABSTRACT
Every characters of textual message are encrypted through the encryption key of RSA algorithm and then to an array of data bits. After that the data bits are embedded to some suitable nonlinear pixel and bit positions about the entire image through system modulus of RSA algorithm. 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. At the decryption end, three key shares are assigned for each and every share. The system modulus and private key are calculated from the prime factors of key shares. The embedded data bits are being retrieved from watermarked image by the system modulus and decrypted to the corresponding characters by the key, finally to its original content.

Keywords
Pixel, invisible digital watermarking, RSA Algorithm, 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. Public key cryptography is used to protect digital data going through an insecure channel from one place to another. RSA algorithm is extensively used in the popular implementations of public key infrastructures. 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 .First 8 bits of the 32 bits are reserved for this opacity (transparency of the image) value. If α is 00000000 the image is fully transparent. Each of three(R, G & B) 8-bit blocks can range from 00000000 to 11111111(0 to 255) [1] [2] [8] [9].

In this paper, we have proposed a technique to embed a message about the entire image. The text is taken from the keyboard or special characters. Then the corresponding ASCII-8 (American Standard Code for Information Interchange) value of characters is encrypted through the public key (e) of RSA algorithm and to its 8-bit binary equivalent. Also the length of the text is converted into its 8-bit binary equivalent. The data bits from length and encrypted characters are respectively stored into an encrypted array (earr[]). Then we have embedded the data bits from the array to some nonlinear pixel and bit positions using the system modulus (N) of RSA algorithm and 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 and stored as integer values to file. Taking any two and the third as 0, for the entire pixels formulate three different shares. From the viewpoint of key, we have formed three key shares by multiplying any two private key elements (but by not repeating two once more). For each and every shares of image different key shares are assigned. Out of those three communicable shares when only minimum two communicable image and key shares come together then we will be able to get back the original message. At the time of decryption the system modulus and private key are calculated from the prime factors of key shares. Using any two image shares we may form the original watermarked image and from it the embedded data bits are being retrieved by the system modulus. From that data bits we get the length and encoded values. Using the private key (d) those values are decrypted to corresponding characters and finally to its original content. In our work, we have targeted any one bit of last four significant bit of each R, G and B of any selected nonlinear pixel position about the entire image using the private key cryptography technique [3] [5] [7] [8].

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 560 X 864 dimension has 4,83,840 pixels. In our work, only we have altered any one bit of last four significant bits. 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.

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.

2. THE SCHEME
This section represents a description of the actual scheme used during "An Enhancement of Security on Image Applying Asymmetric Key Algorithm" 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 & 2.2.2 [4] [6] [7].

2.1 Encryption of data bits about the image
2.1.1 Formation of key using RSA algorithm
Step I: Select two random prime numbers p and q (a less significant amount of 256).

Step II: Compute their system modulus N=p*q and ø(N)=(p-1)*(q-1).
Step III: Select the encryption key (e), where 1<e<ø(N) and gcd(e, ø(N))=1.
Step IV: Find the decryption key (d), where e*d=1 mod ø(N) where 0≤d≤N.
Step V: Publish the public encryption key: KU= {e, N}.
Step VI: Hush up the private decryption key: KR= {d, p, q}. Taking any two, out of that three form the key shares {$K_{[1]}$, $K_{[2]}$, $K_{[3]}$}.
Step VII: Stop.

2.1.2 Create an array of encrypted data from text

Step I: Take input from keyboard or special characters (which ASCII value must be less than N). Calculate the string length (chlen) and to its 8-bit binary equivalent.
Step II: Using the encryption key (e) encrypt (C= M^e mod N) the corresponding ASCII values and into its 8-bit binary equivalent. Accumulate the data bits to earr[bit] as LSB (Least Significant Bit) to earr[1] and MSB (Most Significant Bit) to earr[8] respectively.
Step III: Store the binary values of length and characters to earr [ ] as LSB to earr[1+(i*8)] and MSB to earr[8+i*8].
Step IV: Repeat Step III to Step V for i=0 to (N-1).
Step V: Stop.

2.1.3 Select the nonlinear pixel positions using key

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

F(x, y) = N^p [i.e. pow (N, 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 next step.
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= 1st 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 & 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 0th element up to the end of the array.

A) If bit%3 = 0
Go to Step VIII.
B) If bit%3 = 1

for 0th 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 Step VIII.

C) If bit%3=2

for 0th 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 Step 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: Take the width (W) and height (H) of the 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 Regain of replaced bits from the watermarked image

Step I: Take input any two key shares out of three and calculate the prime factors. Taking first the value of p and q, calculate the system modulus (N) as it was in encryption end.
Step II: Also take any two image shares out of three. 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 position in R, G & B of selected pixels go through Step I to Step VIII of both Algorithm 2.1.3 and 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 repeat Step II to Step IV for i= 1 to 3 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 length (chlen) of message.
Step VII: Retrieving the encrypted bits from the selected pixel and bit positions, store it to decrypted character array from darr[10] to darr[bit] respectively (where bit is the array length from characters and length).
Step VIII: Repeat Step VIII for i=4 to chlen.
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 decimal number (C). Store the numbers to an array dmsg[chlen].
Step X: Stop.

2.2.2 Generation of original content using decryption algorithm of RSA algorithm

Step I: Use private key = {d, p, q}. Then compute message (M) = C^d Mod N.
Step II: Taking the decimal value of message (M) calculate the corresponding characters from ASCII-8 table.
Step III: Finally place the characters one by one from the array msg[len] and assemble the original message**.**
Step IV: Stop.

3. 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 encrypted binary equivalent is defined.

Table 3.1: characters with binary equivalent

Character	Decimal Equivalent	Decimal Equivalent After Encryption	8-Bit Binary Equivalent
N	078	056	00111000
O	079	139	10001011
N	078	056	00111000
L	076	032	00100000
I	073	061	00111101
N	078	056	00111000
E	069	086	01010110
A	065	142	10001110
R	082	082	01011011

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]=1
: :	: :	: :
earr[8]=0	earr[16]=0	earr[80]=0

Figure 3.2: Data bits in encrypted array

Let two prime number p=11 and q=17.
So, the system modulus N=p*q=11*17=187.
Ø(n)=(p-1)*(q-1)=16*10=160.
From here we get the public key= {7, 11, 17} and the private key= {23, 11, 17}.
Using the key elements we get the key shares as
$K_{[1]}$={d*p}= 253; $K_{[2]}$={p*q}=187;
$K_{[3]}$={q*d}=391;
The image size= 560 X 864 (w x h).
Number of effected pixel required for character (p) = [ceil (80/3)] =27.

In the Table 3.3, the image shares with RGB components and corresponding assigned key shares are given (as described in Algorithm 2.1.5).

Table 3.3: positions of array elements about the image

Imge Share	Image Component	Key digits
Image	R,G&B 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-3.4, 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 2.1.3 & 2.1.4).

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
187,1	1.870000e+05	P(187,000)	1st LSB	earr[1] earr[2] earr[3]
:	:	:	:	:
187,5	2.286669e+11	P(228,666)	2nd LSB	earr[13] earr[14] earr[15]
:	:	:	:	:
187,27	1.787956e+52	P(178,795)	3rd LSB	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 also we will be able to get back the encrypted message from that watermarked image at the decryption end.

4. EXPERIMENTAL RESULT

An example with result is shown in Figure 4.1. Fig. 4.1(a) is the first original image. It is one of the popular images of MONALISA. Fig. 4.1(b) is the watermarked image. Fig. 4.1(c) is the image with entire absent of R component (R, G &B of each pixel) generated form watermarked image. Fig. 4.1(d) and 4.1(e) are the image with entire absent of G and B component respectively.

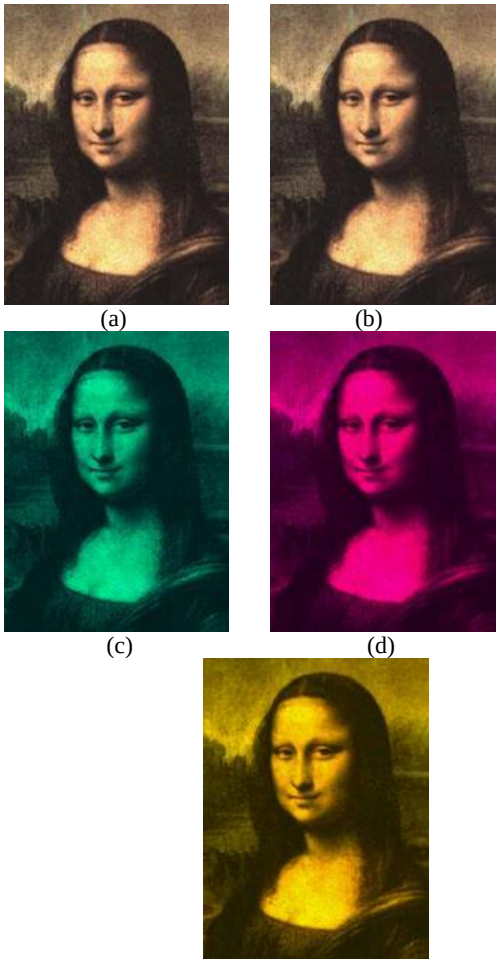

Figure 4.1(a) is the first original image, (b) the watermarked image, (c) (d) and (e) image shares with absent of R, G and B components respectively.

5. ANALYSIS

Here, initially we have produced a watermarked image and then the image shares using any two components of three(R, G and B). We have not used any compression and/or encryption technique before the creation of array (earr[]). Any body may employ any compression and/or encryption technique(s) at the time of watermarking. Then the number of bits to embed into image will be fewer and the strength of encryption will be higher than present. In addition the number of affected pixel will also be fewer than now. Also anybody may employ any other symmetric or asymmetric key cryptographic technique for this purpose. Anybody may choose two random large prime numbers. In that case the value of N will be larger. So forth the number of targeted pixels will be fewer (as *message_lenght ∞ (1 / number_of_keydigit)).* Binary values generated form the textual information replaced nonlinear pixel positions of image. As bits are placed any one bit in lower four bits 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 to size of text. If the image size becomes large and size of text becomes less then it will be quite harder to differentiate the encrypted image from the original image [4] [6] [9].

6. CONCLUSION

Here we have used asymmetric key cryptographic technique to encrypt and implant the data bits (from both text and size of text) in nonlinear pixel and bit positions about the entire image. After that we generated image shares from watermarked image and key shares from the prime elements of private key. And also by the system modulus the encrypted data is embedded into entire image. Moreover, it produces the similar image to see in naked eye at the time of watermarking using this method. At the time of decryption only a proper combination of image shares make possible to form the original watermarked image. Only from the key shares we may produce the system modulus. By using it we may extract the embedded data bits from watermarked image and by the private key we may decrypt the original message from embedded data bits. More over it produces the almost nearly similar image at the time to produce the watermarked image 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.

7. REFERENCES

[1] 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.

[2] 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.

[3] Sabyasachi Samanta, Saurabh Dutta, "Implementation of Invisible Digital Watermarking on Image using Permutation of Keys and Image Shares" Special issue of IJCCT, ISSN:0975-7449, Vol. 2, Issue 2, 3, 4; 2010, "International Conference [ICCT-2010]", December 3 – 5, pp. 125-129.

[4] Sabyasachi Samanta, Saurabh Dutta, "Implementation of Invisible Digital Watermarking on Image using Nonlinear Function", "International Conference on Computer Applications, 2010 (ICCA2010)", December 24-27, 2010, pp.189-195

[5] Sabyasachi Samanta, Saurabh Dutta, "Digital Watermarking through Embedding of Encrypted and Arithmetically Compressed Data into Image using Variable-Length Key",

"The Second International Conference on Network & Communications Security ", January 2-4, 2011, pp. 523-534

[6] 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 ", 8-10 April, 2011, pp. v2-404-v2-408

[7] Sabyasachi Samanta, Saurabh Dutta, Goutam Sanyal, "Digital Watermarking through Embedding of Encrypted and Arithmetically Compressed Data into Image using Variable-Length Key", "International Journal of Network Security & Its Applications (IJNSA)" Vol. 3, No. 2, 2011, pp.30-41

[8] J. M. BlackLedge, M. L. Hallot, "Convert Encryption and Document Authentication using Texture Coding", "i-managers Journal of Software Engineering", Vol. 3, No. 1, July- September 2008

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