

Advanced Steganography Algorithm Using Randomized Intermediate QR Host Embedded With Any Encrypted Secret Message: ASA_QR Algorithm

Somdip Dey

Department of Computer Science, St. Xavier's College [Autonomous], Kolkata, India
Email: somdipdey@acm.org

Kalyan Mondal

Department of Computer Science, St. Xavier's College [Autonomous], Kolkata, India
Email: myfirstmail.k@gmail.com

Joyshree Nath

A.K. Chaudhuri School of Information Technology, Calcutta University, Kolkata, India
Email: joyshreenath@gmail.com

Asoke Nath

Department of Computer Science, St. Xavier's College [Autonomous], Kolkata, India
Email: asokejoy1@gmail.com

Abstract— Due to tremendous growth in communication technology, now it is a real problem / challenge to send some confidential data / information through communication network. For this reason, Nath et al. developed several information security systems, combining cryptography and steganography together, and the present method, ASA_QR, is also one of them. In the present paper, the authors present a new steganography algorithm to hide any small encrypted secret message inside QR Code™, which is then randomized and then, finally embed that randomized QR Code inside some common image. Quick Response Codes (or QR Codes) are a type of two-dimensional matrix barcodes used for encoding information. It has become very popular recently for its high storage capacity. The present method is ASA_QR is a combination of strong encryption algorithm and data hiding in two stages to make the entire process extremely hard to break. Here, the secret message is encrypted first and hide it in a QR Code™ and then again that QR Code™ is embed in a cover file (picture file) in random manner, using the standard method of steganography. In this way the data, which is secured, is almost impossible to be retrieved without knowing the cryptography key, steganography password and the exact unhide method. For encrypting data The authors used a method developed by Nath et al i.e. TTJSA, which is based on generalized modified Vernam Cipher, MSA and NJJSA method; and from the cryptanalysis it is seen that TTJSA is free from any standard cryptographic attacks, like differential attack, plain-text attack or any brute force attack. After encrypting the data using TTJSA, the authors have used standard steganographic method To hide data

inside some host file. The present method may be used for sharing secret key, password, digital signature etc.

Index Terms— Steganography, Encryption, QR Code™, Security, Communication technology

I. INTRODUCTION

To communicate some crucial data from one computer to another now the time has come to combine both cryptography and steganography together to make the data hiding system unbreakable. The proposed data-hiding algorithm may be used in Banking sectors in Defense sector, Educational sector in e-Business etc. One can send secret key or password of a S/W over internet. There are many situations where only cryptography method or simply steganography method is not enough to make the confidential data secured. So it is necessary to build a perfect secure system where both cryptographic method and steganography method may be clubbed together to make the system unbreakable. The present system can not be broken by only applying cryptanalysis or steganalysis. In this paper, the authors present such a security system, which is a combination of both methods in an effective way. We have used TTJSA encryption technique to encrypt the data. Then we store the data in a QR Code™ [14][15]. After that, we use randomization technique to encrypt the QR Code and using steganography method we hide the encrypted QR Code in the cover file (picture file).

The main encryption technique used in this method is TTJSA [19], which is based on generalized modified

Vernam Cipher, MSA [1] and NJJSA. TTJSA is basically a symmetric key cryptographic technique. After applying TTJSA encryption technique the data are stored in a QR Code. Quick Response codes are a type of two-dimensional barcodes consisting of black patterns on a white background invented by the Denso Wave company in 1994 primarily for the purposes of the automotive company. In industry it is used for object hyperlinking by encoding Uniform Resource Locators and Identifiers and displayed in public places like automobiles, tickets, documentations, advertisements, etc. QR Codes have fast rate of coding and decoding method, for this reason QR Codes can be used to hide encrypted messages, so that they can be retrieved very fast when needed.

The QR Code, formed as a result, will then be randomized using byte manipulation multiple times and in random manner. In this way, the QR Code is again encrypted and that encrypted QR Code is hidden in a cover file (picture file), which is at least 10 times larger than the encrypted QR Code. We hide the encrypted QR Code in the cover file using LSB or LSB+1 or LSB+2 or LSB+3 replacement techniques, which is again random and dependent on the symmetric key entered for encryption [3][4].

The most popular method for steganography is the Least Significant Bit (LSB) encoding. Using any digital image, LSB replaces the least significant bits of each byte by the hidden message bits. Depending on the image format the resulting changes made by the least-significant bits are visually detectable or not. For example, the GIF format is susceptible to visual attacks while JPEG being in the frequency domain, is less prone to such attacks. Other modified steganography methods include LSB+1 replacement or LSB+2 replacement or LSB+3 replacement techniques, where +1 or +2 or +3 signifies the bit number measured from the LSB, for replacement.

In this method to secure a data we use the following algorithm:

- 1) Encrypt small secret message using TTJSA method.
- 2) Formation of QR Code of the encrypted data.
- 3) Encrypting the QR Code using Randomization.
- 4) Hide Encrypted QR Code in the cover file using steganography.

II. ENCRYPTION ALGORITHM

A. Encrypt Data Using TTJSA Method

The detail description of TTJSA method is discussed in detail by Nath et al[19]. TTJSA is a symmetric key algorithm which is a combination of 3 distinct cryptography methods namely (i) Generalized modified vernam cipher method with feedback, (ii) NJJSA method which is essentially bit level encryption method and (iii) MSA algorithm which is actually modified generalized Playfair method. Nath et al developed NJJSA[19] and MSA [1] method. The modified

generalized vernam cipher method developed by Nath et al[19]. Here we will describe briefly the above 3 encryption algorithms:

1) Algorithm of TTJSA (Encryption)

```

Step 1: Start
Step 2: Initialize the matrix mat[16][16] with numbers 0
to 255 in row major wise.
Step 3: call keygen() to calculate randomization number
(=times), encryption number (=secure)
Step 4: call randomization() function to randomize the
contents of mat[16][16].
Step 5: times2=times
Step 6: copy file f1 into file2
Step 7: k=1
Step 8: if k>secure go to Step 15
Step 9: p=k%6
Step 10: if p=0 then
    call vernamenc(file2,outf1)
    times=times2
    call njjsaa(outf1,outf2)
    call msa_encryption(outf2,file1)
else if p=1 then
    call vernamenc(file2,outf1)
    times=times2
    call msa_encryption(outf1,file1)
    call file_rev(file1,outf1)
    call njjsaa(outf1,file2)
    call msa_encryption(file2,outf1)
    call vernamenc(outf1,file1)
    times=times2
else if p=2 then
    call msa_encryption(file2,outf1)
    call vernamenc(outf1,outf2)
    set times=times2
    call njjsaa(outf2,file1)
else if p=3 then
    call msa_encryption(file2,outf1)
    call njjsaa(outf1,outf2)
    call vernamenc(outf2,file1)
    times=times2
else if p=4 then
    call njjsaa(file2,outf1)
    call vernamenc(outf1,outf2)
    times=times2
    call msa_encryption(outf2,file1)
else if p=5 then
    call njjsaa(file2,outf1)
    call msa_encryption(outf1,outf2)
    call vernamenc(outf2,file1)
    times=times2
Step 11: call function file_rev(file1,outf1)
Step 12: copy file outf1 into file2
Step 13: k=k+1
Step 14: goto Step 8
Step 15: End

```

2) Algorithm of vernamenc(f1,f2)

```

Step 1: Start vernamenc() function

```

Step 2: The matrix $mat[16][16]$ is initialized with numbers 0-255 in row major wise order
 Step 3: call function `randomization()` to randomize the contents of $mat[16][16]$.
 Step 4: Copy the elements of random matrix $mat[16][16]$ into $key[256]$ (row major wise)
 Step 5: $pass=1$, $times3=1$, $ch1=0$
 Step 6: Read a block from the input file $f1$ where number of characters in the block 256 characters
 Step 7: If block size < 256 then goto Step 15
 Step 8: copy all the characters of the block into an array $str[256]$
 Step 9: call function encryption where $str[]$ is passed as parameter along with the size of the current block
 Step 10: if $pass=1$ then
 $times=(times+times3*11)\%64$
 $pass=pass+1$
 else if $pass=2$ then
 $times=(times+times3*3)\%64$
 $pass=pass+1$
 else if $pass=3$ then
 $times=(times+times3*7)\%64$
 $pass=pass+1$
 else if $pass=4$ then
 $times=(times+times3*13)\%64$
 $pass=pass+1$
 else if $pass=5$ then
 $times=(times+times3*times3)\%64$
 $pass=pass+1$
 else if $pass=6$ then
 $times=(times+times3*times3*times3)\%64$
 $pass=1$
 Step 11: call function `randomization()` with current value of $times$
 Step 12: copy the elements of $mat[16][16]$ into $key[256]$
 Step 13: read the next block
 Step 14: goto Step 7
 Step 15: copy the last block (residual characters, if any) into $str[]$
 Step 16: call function encryption() using $str[]$ and the no. of residual characters
 Step 17: Return

3) Algorithm of function encryption($str[],n$)

Step 1: Start encryption() function
 Step2: $ch1=0$
 Step 3: calculate $ch=(str[0]+key[0]+ch1)\%256$
 Step 4: write ch into output file
 Step 5: $ch1=ch$
 Step 6: $i=1$
 Step 7: if $i=n$ then goto Step 13
 Step 8: $ch=(str[i]+key[i]+ch1)\%256$
 Step 9: write ch into the output file
 Step 10: $ch1=ch$
 Step 11: $i=i+1$
 Step 12: goto Step 7
 Step 13: Return

4) Algorithm for Decryption

Step 1: Start

Step 2: initialize $mat[16][16]$ with 0-255 in row major wise
 Step 3: call function `keygen()` to generate $times$ and $secure$
 Step 4: call function `randomization()`
 Step 5: set $times2=times$
 Step 6: call `file_rev(f1,outf1)`
 Step 7: set $k=secure$
 Step 8: if $k<1$ go to Step 15
 Step 9: call function `file_rev(outf1,file2)`
 Step 10: set $p=k\%6$
 Step 11: if $p=0$ then
 call `msa_decryption(file2,outf1)`
 call `njjsaa(outf1,outf2)`
 call `vernamdec(outf2,file2)`
 $times=times2$
 else if $p=1$ then
 call function `vernamdec(file2,outf1)`
 set $times=times2$
 call function `msa_decryption(outf1,outf2)`
 call function `njjsaa(outf2,file2)`
 call function `file_rev(file2,outf2)`
 call function `msa_decryption(outf2,outf1)`
 call function `vernamdec(outf1,file2)`
 $times=times2$
 else if $p=2$ then
 call `njjsaa(file2,outf1)`
 call `vernamdec(outf1,outf2)`
 $times=times2$
 call `msa_decryption(outf2,file2)`
 else if $p=3$ then
 call `vernamdec(file2,outf1)`
 $times=times2$
 call `njjsaa(outf1,outf2)`
 call `msa_decryption(outf2,file2)`
 else if $p=4$ then
 call `msa_decryption(file2,outf1)`
 call `vernamdec(outf1,outf2)`
 $times=times2$
 call `njjsaa(outf2,file2)`
 else if $p=5$ then
 call `vernamdec(file2,outf1)`
 $times=times2$
 call `msa_decryption(outf1,outf2)`
 call `njjsaa(outf2,file2)`
 Step 12: copy the content of file2 to outf1
 Step 13: set $k=k-1$
 Step 14: Goto Step 8
 Step 15: End

5) Algorithm of function vernamdec($f1,f2$)

The algorithm of `vernamdec()` function is same as `vernamenc()` function. Here the only difference is that `decryption()` function is called instead of `encryption()` function.

6) Algorithm of decryption($str[],n$)

Step 1: Start
 Step 2: $ch1=0$
 Step 3: $ch=(256+str[0]-key[0]-ch1)\%256$

Step 4: write ch into the output file
 Step 5: i=i+1
 Step 6: if i then goto Step 12
 Step 7: ch=(256+str[i]-key[i]-str[i-1]) %256
 Step 8: write ch into the output file
 Step 9: i=i+1
 Step 10: goto Step 6
 Step 11: ch1=str[n-1]
 Step 12: Return

7) Algorithm of function file_rev(f1,f2)

Step 1: Start
 Step 2: open the file f1 in input mode
 Step 3: open the file f2 in output mode
 Step 4: calculate n=sizeof(file f1)
 Step 5: move file pointer to n
 Step 6: read one byte
 Step 7: write the byte on f2
 Step 8: n=n-1
 Step 9: if n>=1 then goto step-6
 Step 10: close file f1, f2
 Step 11: Return

8) NJJSAA Algorithm

Nath et al. [2] proposed a method which is basically a bit manipulation method to encrypt or to decrypt any file.

The encryption number (=secure) and randomization number (=times) is calculated according to the method mentioned in MSA algorithm [1].

Step 1: Read 32 bytes at a time from the input file.
 Step 2: Convert 32 bytes into 256 bits and store in some 1- dimensional array.
 Step 3: Choose the first bit from the bit stream and also the corresponding number(n) from the key matrix. Interchange the 1st bit and the n-th bit of the bit stream.
 Step 4: Repeat step-3 for 2nd bit, 3rd bit...256-th bit of the bit stream
 Step 5: Perform right shift by one bit.
 Step 6: Perform bit(1) XOR bit(2), bit(3) XOR bit(4),...,bit(255) XOR bit(256)
 Step 7: Repeat Step 5 with 2 bit right, 3 bit right,...,n bit right shift followed by Step 6 after each completion of right bit shift.

9) MSA (Meheboob, Saima, Asoke) Encryption and Decryption Algorithm

Nath et al. (1) proposed a symmetric key method where they have used a random key generator for generating the initial key and that key is used for encrypting the given source file. MSA method is basically a substitution method where we take 2 characters from any input file and then search the corresponding characters from the random key matrix and store the encrypted data in another file. MSA method provides us multiple encryptions and multiple decryptions. The key matrix (16x16) is formed from all characters (ASCII code 0 to 255) in a random order.

The randomization of key matrix is done using the following function calls:

Step-1: call Function cycling()
 Step-2: call Function upshift()
 Step-3: call Function downshift()
 Step-4: call Function leftshift()
 Step-5: call Function rightshift()

N.B: Cycling, upshift, downshift, leftshift, rightshift are matrix operations performed (applied) on the matrix, formed from the key.

B. Formation of QR Code of the encrypted data:

In this method the formation of QR Code from the encrypted message is discussed. The first step in creating a QR code is to create a string of data bits. This string includes the characters of the original message (encrypted message in this case) that you are encoding, as well as some information bits that will tell a QR decoder what type of QR Code it is. After generating the aforementioned string of bits, we use it to generate the error correction code words from the QR Code. QR Codes use Reed-Solomon Error Correction technique [13]. In coding theory, Reed-Solomon codes (RS codes) are non-binary cyclic error correction codes invented by scientists Irving S. Reed and Gustave Solomon. After the generation of bit-string and error correction code words, the resultant data is used to generate eight different QR Codes, each of which uses a different mask pattern. A mask pattern controls and changes the pixels to light or dark ones, according to a particular formula. The eight mask pattern formulas are defined in the QR Code specification, which is referred to as the time of mask generation needed for the QR Code generation. Each of the eight QR codes is then given a penalty score that is based on rules defined in the QR specification. The purpose of this step is to make sure that the QR code doesn't contain patterns that might be difficult for a QR decoder to read, like large blocks of same-colored pixels, for example. After determining the best mask pattern, the QR Code, which uses that mask pattern is generated and shown as an output.

If the size of the encrypted message becomes more than 1,900 characters then the characters appearing after 1,900 characters are used separately to generate another QR Code and the above mentioned process is repeated until and unless the total encrypted message is converted to QR Code(s).

The method is discussed in details below:

Now, the Encrypted file, which is created in the process of TTJSA is now treated as the input file and the string is extracted from the file to generate the QR Code.

Step 1: call function file_read(output_file)
 Step 2: call function generateQRCode(str[])
 Step 3: call function delete_file(output_file)

C. Encrypting the QR Code using Randomization :

In this step, we randomize the content of the QR Code generated in such a way that the file can not be read or

data can not be retrieved without knowing the randomization technique used on it. After execution of this step, the content of the QR Code is totally randomized and the steps are executed in a random fashion, which will depend on the symmetric key entered for encryption technique. But, before applying randomization technique, the whole file (QR Code) is broken up into different blocks, each of size 256 (16 x 16). We form a square matrix of dimension '16 x 16' holding each byte of the QR Code file i.e., each matrix can hold 256 bytes of the QR Code file. In this way we break the whole QR Code file into several blocks and then apply randomization technique on each block. To execute this step, we follow the following algorithm in random manner:

- Step-1: Function cycling()
- Step-2: Function upshift()
- Step-3: Function rightshift()
- Step-4: Function downshift()
- Step-5: Function leftshift()
- Step-6: Function column_Randomization()
- Step-7: Function row_Randomization()
- Step-8: Function left_diagonal_Randomization()
- Step-9: Function_right_diagonal_Randomization()

The above randomization process we apply for n1 times, which will depend on the symmetric key entered for encryption technique, and in each time we change the sequence of operations to make the system more random. Once the randomization is complete we write one complete block in the output key file. Thus, after randomization is applied on all the blocks, the blocks are written on the output file. If the size of the last block is not equal to 256 i.e., less than 256 bytes, then randomization technique is not applied on the last block and this block is called the 'residue block' and it is written down as it is on the output file. Thus, the encrypted QR Code is formed, which will again be hidden in a cover file using steganography method.

D. Hide Encrypted QR Code in the cover file using steganography:

Almost all digital file formats can be used for steganography, but the formats that are more suitable are those with high degree of redundancy. Redundancy can be defined as the bits of an object that provide accuracy far greater than necessary for the object's use and display. The redundant bits of an object are those bits that can be altered without the alteration being detected easily. In our present work we will try to embed QR Code's .PNG file, which was generated in the previous step of the method, in some image file such as .JPEG or .PNG or .BMP file. We will be doing mainly steganography in image domain:

Image Domain -

(i) Least Significant Bit (LSB)

Insertion method: Least significant bit(LSB) insertion is a common, simple approach to embedding information in a cover image. The least significant bit or in other

words 8-th bit of some or all the bytes inside an image is changed to a bit of the secret message. Let us consider a cover image contains the following bit patterns:

Byte-1 Byte-2 Byte-3 Byte-4
 00101101 00011100 11011100 10100110

Byte-5 Byte-6 Byte-7 Byte-8
 11000100 00001100 11010010 10101101

Suppose we want to embed a number 200 in the above bit pattern. Now the binary representation of 200 is 11001000. To embed this information we need at least 8 bytes in cover file. We have taken 8 bytes in the cover file.

Now we modify the LSB of each byte of the cover file by each of the bit of embed text 11001000, and then, we want to show what happens to cover file text after we embed 11001000 in the LSB of all 8 bytes:

TABLE I
 CHANGING LSB

Before Replacement	After Replacement	Bit inserted	Change in Cover file
00101101	00101101	1	No change in bit pattern
00011100	00011101	1	Change in bit pattern(i)
11011100	11011100	0	No change in bit pattern
10100110	10100110	0	No change in bit pattern
11000100	11000101	1	Change in bit pattern(ii)
00001100	00001100	0	No change in bit pattern
11010010	11010010	0	No change in bit pattern
10101101	10101100	0	Change in bit pattern(iii)

So here we can see that out of 8 bytes only 3 bytes get changed only at the LSB position. Since we are changing the LSB hence we are either changing the corresponding character in forward direction or in backward direction by only one unit and depending on the situation there may not be any change also as we have seen in the above example. As our eye is not very sensitive so therefore after embedding a secret message in a cover file our eye may not be able to find the difference between the original message and the message after inserting some secret text or message on to it.

(ii) Change of bit at LSB+1 bit:

In this method instead of inserting bit in LSB we also insert a bit in LSB+1 position. It means to embed 1 byte secret message we need only 4-bytes. We choose the first 4 bytes of the cover file and the same secret message 11001000. We are now showing how the secret message is been inserted in the cover file:

TABLE II
CHANGING LSB+1

Before Replacement	After Replacement	Bit inserted	Change in Cover file
00101101	00101111	11	Change in bit pattern(i)
00011100	00011100	00	No change in bit pattern
11011100	11011110	10	Change in bit pattern(ii)
10100110	10100100	00	Change in bit pattern(iii)

Here we can see that first, third and fourth bytes have been modified but 2nd byte remains same. It is obvious that if we replace 2 consecutive bits then change of byte in cover file will be prominent and hence this method may not be appropriate to embed secret message in any .JPEG file but this may work perfectly in any .BMP file. Human eye will not be able to detect the difference between the original cover image and the cover image after inserting secret message in 2 consecutive bits in every byte of the cover file.

(iii) Change of bit at LSB+3bit:

In this method we modify only the LSB+3 bit in the cover file. This is almost same as changing the bit in LSB position. It is already proved that to insert bits of the secret message in the LSB of cover file is the most safe method as it changes the minimum in the cover file. Here we try to find whether we get slightly better or worse result if we modify the 4-th bit from the R.H.S. We made long experiment on different .JPEG file and also on .BMP file and we finally we conclude that for .BMP file this works perfectly OK but for .JPEG file it is not better than method-1. Now we will be showing how the bits are changing in the cover file:

TABLE III
CHANGING LSB+3

Before Replacement	After Replacement	Bit inserted	Change in Cover file
00101101	00101101	1	No change in bit pattern
00011100	00011101	1	No change in bit pattern
11011100	11010100	0	Change in bit pattern(i)
10100110	10100110	0	No change in bit pattern
11000100	11001101	1	Change in bit pattern(ii)
00001100	00000100	0	Change in bit pattern(iii)
11010010	11010010	0	No change in bit pattern
10101101	10100101	0	Change in bit pattern(iv)

In this particular case we can see that 4 bytes have been modified and the rest 4 bytes remain same. But again this may change for different pattern and different cover file.

In all 3 aforementioned methods we have used some common scheme:

We did not insert any secret message from the beginning of any cover file so that the header portion of any file should remain intact. The last 1000byte we use to store the password and size of the secret file. To embed secret message we have to first skip 1000bytes from the last byte of the cover file. After that according to size of the secret message (say n bytes) we skip $8*n$ bytes or $4*n$ depending on scheme we have used. After that we start to insert the bits of the secret file into the cover file. Under no circumstances the size of the cover should not be less the $10* \text{sizeof}(\text{secret message})$ then our method will fail.

For extracting embedded file from the cover file we have to perform the following: To enter the password which we have entered at time of embedding a file in the cover file. If password is correct then, the program will read the file size from the cover file. Once we get the file size we follow simply the reverse process of embedding a file in the cover file. We read LSB of each byte and accumulate 8 bits to form a character and we immediately write that character on to a file.

DECRYPTION METHOD:

All the steps of the method used in this paper can be reversed and the decryption can be done easily by following the reverse engineering of the algorithms used in this method.

III. RESULTS AND DISCUSSIONS

We chose different test cases and used them to test the combined cryptographic method in this paper, and following are few given examples:

Original Text	Encrypted text	QR Code	Original Cover File	Embedded Cover File
(i) 128 Number of ASCII Value (1)	<pre>∞%∂ Ä··Z- Ë™òuíøLVfS- 7Ω³·;,,, H*ÆδW™TÄ†JÏ <Γ”Äç ÄyÄ≠ À Á@pç 5o 3BÓ°IZm™Qkâ4 4sVZX”áfi□ò·,£ C+sgÆ'.ì ~úgkÆμ @δSÏx”XRU/□a ©†â[dúOv□Ï□î”</pre>			
(ii) abcdefghijklmn opqrstuvwxyz01 23456789abcdef ghijklmnopqrstu vwxyz01234567 89abcdefghijklmnop qrstuvwxyz0123 456789abcdefghijkl mnopqrstuvwxyz 0123456789 abcdefghijklmnop qrstuvwxyz01234 56789abcdefghijkl mnopqrstuvwxyz0 123456789	<pre>?□,ë...R√ "±oq-#Ä-€>©} \$+~:ÖejçÜhè‡Æ' D ĬÐÇó¥f1l- ;~...l\ Pè— ä í-S-°úΣ ô¿â«cÈÉ/□Δ”≥O ,□iÖiæ“+Cæ\$□ æ:7b i ÆE √>tÁyã□C],á'√ □ææ□√ Σ ≤u@-âX □,ilf· Y áøð □¥WΠĭ é Ê π †(J 8□◇Èĭê”t≈SbV} ◇9◇•\$ΩçM_Î* ØO ÿ ’æ é %P`dđLDðØ`ã w≠_`p<çÇ:</pre>			

Note: We used .BMP files for the steganography part of this combined cryptographic method with highest preference, because the distortion in the embedded cover file is visually detectable in all other popular picture file formats like .JPG or .PNG, except in .BMP file format; and for this reason, if other file formats (except .BMP) are used then, 'steganalysis' can be done easily by 'Visual LSB Detection method'.

IV. CRYPTANALYSIS

One of the classical cryptanalysis method used is by detecting the frequency of characters in the encrypted text (message). So to test the effectiveness of the combined cryptographic method presented in this paper, spectral analysis of the frequency of characters are closely observed.

Using this cryptographic method we ran many analysis and tested different strings as input and used various methods of cryptanalysis. To show the usefulness and integrity of this cryptographic module, we used spectral analysis of the frequency of characters.

Original Text (Message)	Encrypted Message Using TTJSA Encryption Module
128 Numbers of ASCII Value (1)	<pre>∞%∂ Ä··Z- Ë™òuíøLVfS-7Ω³·;,,, H*ÆδW™TÄ†JÏ<Γ”Äç ÄyÄ≠ À Á@pç □5o3BÓ°TZ m™Qkâ44sVZX”áfi□ò·, £C+sgÆ'.ì ~úgkÆμ@δSÏx”XR U/□a©†â[dúOv□Ï□î”</pre>

Thus, Fig 1.1 shows the spectral analysis of the frequency of characters of the original text and Fig 1.2 shows the spectral analysis of the frequency of characters of the encrypted text.

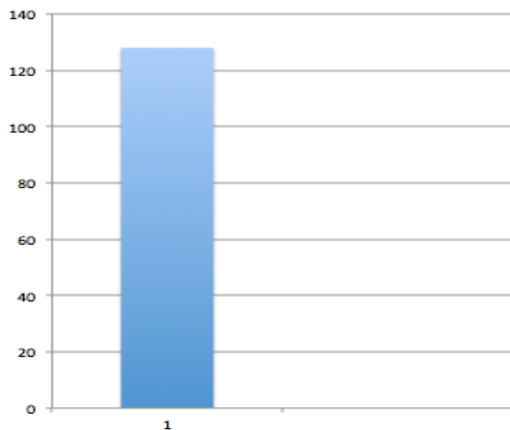


Fig 1.1: Spectral Analysis of the Original text

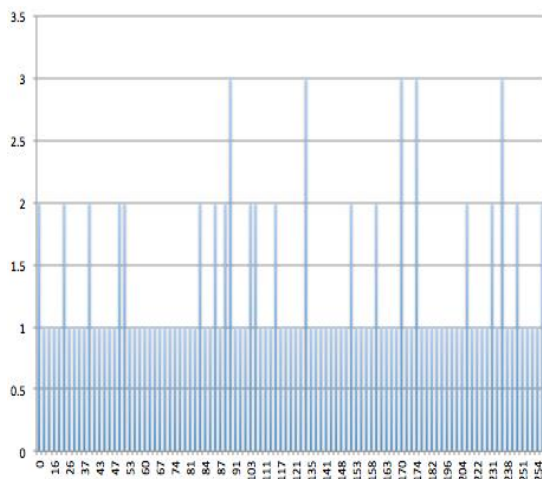


Fig 1.2: Spectral Analysis of the Encrypted Text

Thus, from the spectral analysis we can see that, for a repeated character also, the encryption technique covers a wide range of ASCII value and the frequency of the occurrence of the ASCII values are different and there is no detectable pattern. For this reason, the encryption technique is used in this method is very strong and can not be broken by normal cryptanalysis methods like differential attack, spectral analysis, brute force method, etc.

V. STEGANALYSIS

Steganalysis is the art of detecting messages hidden by stegosystems [16]. Steganalysis can be performed by examining the statistics of the cover image or by visual inspection.

There are different types of attacks against such systems [17][18]. In one such attack, the *Known cover attack*, the original cover object and the stego-object are available for analysis. The idea in this attack is to compare the original media with the stego-media and note the differences. These differences may lead to the

emergence of patterns that would constitute a signature of a known steganographic technique. A different approach to steganalysis is to model images using a feature vector as in blind steganalysis and capture the relationship between the change in the feature vector to the change rate using regression. Yet another approach is based on the Maximum Likelihood principle. The concept of steganographic security, in the statistical sense, has been formalized by Cachin by using an information-theoretic model for steganography. In this model the action of detecting hidden messages is equivalent to the task of hypothesis testing. In a perfectly secure stegosystem the eavesdropper has no information at all about the presence of a hidden message.

In our method, as discussed earlier, the change can be detected easily by Visual LSB Detection method and to deal with that problem, we prefer to use .BMP files as our cover file to hide the data. Use of .BMP cover files make it very hard to detect the change in usual visual detection method.

VI. CONCLUSION AND FUTURE SCOPE

The word steganography is well known to data communication and network for hiding data inside some known image and then send through internet. In digital steganography the process of data hiding has been modified drastically in the last few years. Nath et al already developed several methods for hiding secret message inside some common image, audio or video. Our proposed method is much better than any existing steganography methods as here we hide the secret message in two steps. In step-1 we embed encrypted secret message inside QR Code™. Then, the embedded QR Code is randomized using the randomization method used in MSA algorithm. Finally, the randomized embedded code is inserted inside some common image. To get back the original secret message, we have to first decode the randomized embedded QR Code from the embedded image. Then, we have to apply reverse randomization method to get back original embedded QR Code. Finally, we extract the secret message from that second host file and then apply the decryption method to get back original secret message. Because of this double encryption and double embedding method, no one can extract the original secret message without knowing the exact method. The present method may be used to this proposed method, SKJA, aims at imparting more security to the confidential data. The present method, SKJA, may be further improved by compressing the QR Code, by hiding it in text or document files, which are abundant on the internet and will not give rise to any suspicion by the attacker. This work can be further improvised upon in the future in a number of ways.

ACKNOWLEDGEMENT

The authors, SD, KM, JS and AN are grateful to the entire Department of Computer Science of St. Xavier's College [Autonomous], Kolkata, for giving them the

opportunity and facilities to work on this research paper, and for bearing with them throughout the course of this work. One of the authors, AN, sincerely expresses his gratitude to Fr. Dr. Felix Raj and Fr. Jimmy Keepuram for giving constant inspiration to carry out the research work. AN is grateful to the University Grants Commission for granting financial assistance through Minor Research Project. SD, KM, JS and AN are also thankful to all the students of 3rd year, Computer Science Hons. (2011-2012 batch) for their encouragement and support to finish this work.

REFERENCES

- [1] Symmetric Key Cryptography using Random Key generator : Asoke Nath, Saima Ghosh, Meheboob Alam Mallik: "Proceedings of International conference on security and management(SAM'10" held at Las Vegas, USA Jul 12-15, 2010), P-Vol-2, 239-244(2010).
- [2] An Integrated Symmetric key Cryptography Algorithm using Generalised modified Vernam Cipher method and DJSA method: DJMNA symmetric key algorithm : Debanjan Das, Joyshree Nath, Megholova Mukherjee, Neha Choudhary, Asoke Nath: Communicated for publication in IEEE International conference WICT 2011 to be held at Mumbai Dec 11-14, 2011.
- [3] Data Hiding and Retrieval : Asoke Nath, Sankar Das, Amlan Chakraborty, published in IEEE "Proceedings of International Conference on Computational Intelligence and Communication Networks(CICN 2010)" held from 26-28 NOV' 2010 at Bhopal.
- [4] Advanced Steganographic approach for hiding encrypted secret message in LSB, LSB+1, LSB+2, LSB+3 bits in non standard cover files : Joyshree Nath, Sankar Das, Shalabh Agarwal and Asoke Nath, International Journal of Computer Applications, Vol-14, No. 7, Page-31-35, Feb (2011).
- [5] Advanced Steganography Algorithm using encrypted secret message : Joyshree Nath and Asoke Nath, International Journal of Computer Science and Applications, Vol-2, No. 3, Page- 19-24, Mar (2010).
- [6] A Challenge in hiding encrypted message in LSB and LSB+1 bit positions in any cover files: executable files, Microsoft Office files and database files, image files, audio and video files : Joyshree Nath, Sankar Das, Shalabh Agarwal and Asoke Nath : JGRCS, Vol-2, No. 4, Page-180-185, Apr (2011).
- [7] New Data Hiding Algorithm in MATLAB using Encrypted secret message: Agniswar Dutta, Abhirup Kumar Sen, Sankar Das, Shalabh Agarwal and Asoke Nath : Proceedings of IEEE CSNT-2011 held at SMVDU (Jammu), 03-06 Jun, 2011, Page 262-267.
- [8] New Steganography algorithm using encrypted secret message : Joyshree Nath, Meheboob Alam Mallik, Saima Ghosh and Asoke Nath : Proceedings of Worldcomp 2011 held at Las Vegas (USA), 18-21 Jul, 2011.
- [9] Steganography In Digital Media: Principles, Algorithms and Applications by Jessica Fridrich : Cambridge University Press.
- [10] Cryptography and Network Security, William Stallings, Prentice Hall of India.
- [11] Cryptography & Network Security, Behrouz A. Forouzan, Tata McGraw Hill Book Company.
- [12] Cryptography and Information Security, V. K. Pachghare, Prentice Hall of India
- [13] Reed and G. Solomon. Polynomial codes over certain finite fields. Journal of the Society for Industrial and Applied Mathematics, 8(2):300-304, 1960.
- [14] "QR Code, Wikipedia", http://en.wikipedia.org/wiki/QR_code [Online] [Retrieved 2012-02-09]
- [15] "ZXING- QR Code Library ", <http://code.google.com/p/zxing/> [Online] [Retrieved 2012-02-09]
- [16] N. Johnson and S. Jajodia. Steganalysis: The investigation of hidden information. *Proc. Of the 1998 IEEE Information Technology Conference*, 1998.
- [17] B. Dunbar. A detailed look at steganographic techniques and their use in an open- systems environment. *Sans InfoSec Reading Room*, 2002.
- [18] E. T. Lin and E. J. Delp. A review of data hiding in digital images. Proceedings of the Image Processing, Image Quality, Image Capture Systems Conference, 1999.
- [19] Symmetric key cryptosystem using combined cryptographic algorithms - Generalized modified Vernam Cipher method, MSA method and NJSSAA method: TTJSA algorithm Proceedings of "Information and Communication Technologies (WICT), 2011 " held at Mumbai, 11th – 14th Dec, 2011, Pages:1175-1180.
- [20] Somdip Dey, "SD-EQR: A New Technique To Use QR CodesTM in Cryptography", Proceedings of "1st International Conference on Emerging Trends in Computer and Information Technology (ICETCIT 2012)", Coimbatore, India, pp. 11-21.

Somdip Dey is currently a final year student of B.Sc (Hons) in Computer Science at St. Xavier's College [Autonomous], Kolkata, India. His interests of research are on Cryptography, Information and Network Security, Computer Architecture and HPC (High Performance Computing), and have few publications on the aforementioned topics. He is also a Microsoft Student Partner (MSP) representing his college from India.

Kalyan Mondal is currently a final-year undergraduate student at the Department of Computer Science at St. Xavier's College (Autonomous), Kolkata. His research interests include web engineering, web development using PHP and J2EE, cloud computing, graphics designing, network security, ethical hacking, etc. He has been working in the fields of web development for past four years.

Joyshree Nath is currently a final-year student in M.Tech(IT) at A.K.Chaudhuri School of IT, Calcutta University. Her research area is Cryptography, Steganography, e-learning. She has attended several International conferences in India and in abroad.

Asoke Nath is Assistant Professor in Department of Computer Science, St. Xavier's College(Autonomous), Kolkata. His main research area include Cryptography, Steganography, Green Computing, e-learning and Distance Education. He has attended many International Conferences in India and USA.