ISSN: 0067-2904
GIF: 0.851

X.K.N: A Proposed Method for Data Encryption Using XOR and NOT Logical Gates with LFSR Generated Keys

Qusay Mohamed Jaafar*
Department of Computer Communication Engineering, Alrafidain University College, Baghdad, Iraq

Abstract

In this paper, a method for data encryption was proposed using two secret keys, where the first one is a matrix of **XOR's** and **NOT's** gates (XN key), whereas the second key is a binary matrix (KEYB) key. XN and KEYB are (m*n) matrices where m is equal to n. Furthermore this paper proposed a strategy to generate secret keys (KEYBs) using the concept of the LFSR method (Linear Feedback Shift Registers) depending on a secret start point (third secret key s-key). The proposed method will be named as X.K.N. (X.K.N) is a type of symmetric encryption and it will deal with the data as a set of blocks in its preprocessing and then encrypt the binary data in a case of stream cipher.

Keywords: Computer security, Encryption, LFSR, Symmetric, block and stream cipher, Exclusive OR, NOT.

X.K.N طريقة مقترحة لتشفير البيانات باستخدام البوابات المنطقية XOR و NOT مع مولد المفاتيح LFSR

قصي محمد جعفر*
قسم هندسة اتصالات الحاسوب، كلية الرافدين الجامعة، بغداد، العراق

الخلاصة

في هذا البحث تم اقتراح طريقة لتشفير البيانات باستخدام مفتاحين سريين، حيث ان المفتاح الاول سوف يكون كمصفوفة من بوابات XOR وبوابات NOT وسوف يتم تسميته (المفتاح XN)، اما المفتاح الثاني سيكون كمصفوفة ثنائية وسوف يتم تسميته (المفتاح KEYB).XN و KEYB هي مصفوفتين بحجم (m*n) حيث ان m يساوي n. ايضاً في هذه الطريقة المقترحة تم اقتراح استراتيجية لتوليد عدد من المفاتيح السرية (KEYBs) مستخدما مفهوم طريقة (LFSR) (مسجل ازاحة التغذية المسترجعة الخطية) اعتماداً على نقطة بداية سرية (مفتاح سري ثالث s-key). هذه الطريقة المقترحة سوف يتم تسميتها X.K.N وتعتبر كنوع من التشفير المتناظر والتي سوف تتعامل مع البيانات كمجموعة من الكتل في مراحل المعالجة الاولية ومن ثم تشفير البيانات الثنائية في حالة من التشفير التدفقي.

1- Introduction

Everyone in this world need to protect his/her information from others, normally when we talk about information protection this will lead to the encryption. Before beginning with the proposed method, lets firstly give a simple definition to encryption. Encryption is a tactic to prevent unauthorized persons to understand what they are getting from insecure channels. Encryption needs strategy (algorithm) and key to work and both must be strong enough to obtain the protected information.

*Email: qmjsalman@yahoo.com

To understand what X.K.N refers to: X from XOR gate, K from key, and N from NOT gate. The proposed method is considered as symmetric encryption, in which the same keys will be used in encryption and decryption phases. Also the proposed method depends totally on the XOR and NOT gates to give an encrypted data. There are two main secret keys one of them as a block of binary digits named as KEYB key, while the other will be a block of two symbols X and N named as XN key. Besides, there is another key used for keys generation named as s-key, so that the proposed method will need from the encrypter and the decrypter to enter three secret keys.

The *objective* of this research is to propose a new algorithm for data encryption using two main matrices; the first one is XOR and NOT operations, the second one is 0's and 1's to encrypt the data, each one of these matrices will represent a secret key, where the second key will be generated using LFSR.

2- Theoretical Background

This section explains the theoretical background about the two main concepts: XOR, and LFSR.

2.1 Vernam (XOR)

"The Vernam cipher generates a cipher text bit stream $C = E_K(M)$" [1]

$$\mathbf{c_i = (m_i + k_i) \bmod 2, \quad i = 1, 2. \ldots .}$$

"The Vernam cipher is efficiently implemented in microelectronics by taking the "exclusive-or" of each plaintext/key pair" [1]

$$c_i = m_i \oplus k_i$$

Let the plaintext is 10110 and the key is 10011 and by applying XOR bitwise we will get the cipher text 00101.

Many researchers use XOR to perform the encryption process. XOR is applied on each element of the data matrix of size 8*8 with rotation process [2].

2.2 Linear feedback shift register (LFSR)

"An elegant way of realizing long pseudorandom sequences is to use linear feedback shift registers (LFSRs). LFSRs are easily implemented in hardware and many, but certainly not all, stream ciphers make use of LFSRs. A prominent example is the A5/1 cipher, which is standardized for voice encryption in GSM, combinations of LFSRs, such as A5/1 or the cipher Trivium, can make secure stream ciphers". [3].

For example a *simple* LFSR [3] linear feedback shift register of degree 3 with initial values s_2, s_1, s_0, the output bit of the LFSR is computed as:

$$\mathbf{S(i+3) \equiv S(i+1)+S(i) \bmod 2} \qquad \mathbf{(1)}$$

"The maximum sequence length generated by an LFSR of degree m is $2^m - 1$ "[3].

Where, $S_{(i+3)} \equiv S_{(i+1)}+S_{(i)} \bmod 2 \implies S_{(i+3)} = S_{(i+1)} \oplus S_{(i)}$

There are many attempts to generate the secret keys and [4] one of them. There are two conventional LFSR: standard and modular [5]. This research (X.K.N) used the principle of the modular LFSR as a method to generate the KEYBs.

3- The proposed X.K.N method structure

The proposed method used XOR as a level of its strategy in the encryption/decryption and in the key generation. Also the proposed method takes the two concepts of the encryption (stream cipher and block cipher), the stream cipher in its low level encryption, and block cipher as results of encrypted blocks that will be grouped to reconstruct the cipher/plain text. The proposed method for the keys generation that depends totally on the LFSR it is stream cipher also.

The idea of X.K.N begins after some contemplation in XOR gate and how it is possible to use more than one gate of XOR to produce a good cipher method. There are many methods used more than one XOR in its strategy, but in this research a matrix of (XOR and NOT) gates is used as a key called XN with another matrix of 0s and 1s as another key called (KEYB). These two matrices (XN and KEYB) will be applied on the DATA matrix to produce an encrypted DATA matrix, from this point it is good to deal with DATA matrix, XN (XOR and NOT) matrix, and KEYB matrix as blocks of bits or contents. The bits refer to the DATA matrix values and KEYB matrix values, while contents refer to the elements of the XN matrix (X,N). So XN will be represented with two elements X for XOR gate and N for NOT gate (one bit complement). Figure-1a-c shows the three matrices of DATA, XN, and KEYB (key as binary).

0	0	1	1	.	.
1	1	0	1	.	.
0	1	1	0	.	.
0	0	1	1	.	.
.	.	.	.	.	.
.	.	.	.	.	.

X	N	X	N	.	.
N	N	X	X	.	.
X	N	N	N	.	.
X	X	N	X	.	.
.	.	.	.	.	.
.	.	.	.	.	.

1	1	0	1	.	.
0	1	1	0	.	.
0	1	0	0	.	.
1	1	0	1	.	.
.	.	.	.	.	.
.	.	.	.	.	.

Figure 1- a-DATA matrix of (m*n) (One block) b- XN matrix of (m*n) c- KEYB matrix of (m*n)

It is important to mention that the dimensions size of the XN and KEYB must be the same, and applying the KEYBs on the DATA will depend totally on the XN, so that there are two secret keys (XN and KEYB), KEYB will be generated after each iteration of encryption to get q keys of KEYB type with one XN key. Equation (2) explains how the DATA will be encrypted, while equation (3) explains how the Enc will be decrypted.

$$Enc(i,j)= E(XN(i,j) [DATA(i,j),KEYB(i,j)]) \quad (2)$$

$$DATA(i,j)=D(XN(i,j) [Enc(i,j), KEYB(i,j)]) \quad (3)$$

Where Enc(i,j) refers to the result of the encrypted block, E denotes the mechanism of the X.K.N in encryption stage (the proposed method), DATA (i,j) refers to the data block before encryption and after decryption, D denotes the mechanism of the X.K.N in decryption stage, KEYB (i,j) denotes the binary secret keys of size m*n, XN(i,j) denotes the XN matrix of size m*n with X and N elements.

X.K.N uses the concept of the LFSR method (linear feedback shift register), to generate (q) number of keys of size m*n as shown in section (3-2) with some modification on Eq(1).

X.K.N behaves as a block and stream cipher in encryption phase and decryption phase but as a stream cipher when it generates (q) keys from the initial secret KEYB using LFSR.

3.1 The Mechanism of the proposed method (X.K.N):

There is no relation between LFSR and this proposed method in its essences, but LFSR is used to generate the secret keys (KEYBs) from initial secret key ($KEYB_1$). Figure-2 shows the block diagram of the proposed method.

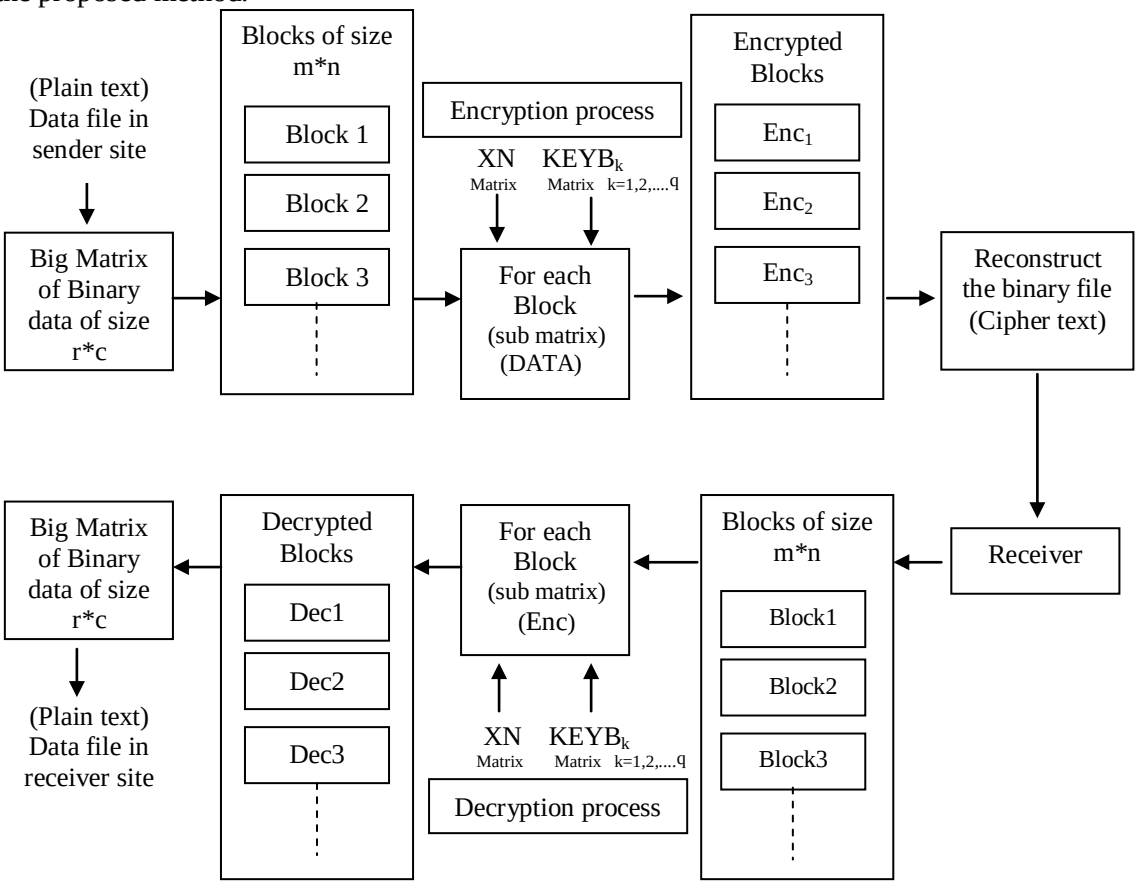


Figure 2- Block diagram of the proposed method

As shown in Figure-2 there are two secret keys one of them generated by itself by using LFSR (according to initial secret key) to produce new matrices of keys ($KEYB_k$), whereas the other is a matrix of two elements, each one represents one case, X for exclusive OR, N for NOT. Now lets show some points of the proposed method (X.K.N):

Firstly, it must to deal with the data file as a binary matrix of size r*c (r row,c column), and then divide this matrix to form DATAs (blocks of binary data) of size m*n where m and n are the dimensions of the $DATA_z$, $KEYB_k$ and XN matrices whereas m equal to n. For example the string (HELLOSIR) will be a matrix of 8*8 bits and the XN key and KEYB key are matrices of 4*4 as shown in Fig.(1), so, there are 4 blocks of DATA each of size (4*4) and for each block the same XN matrix but different KEYB (in this case the first four KEYBs are enough), in general for long string the KEYB will generate until the KEYB will be repeated ($KEYB_1$, $KEYB_2$,…..$KEYB_q$, q is the total number of KEYB generated by modified LFSR) ($DATA_1$ will be encrypted depending on XN and $KEYB_1$, $DATA_2$ will be encrypted depending on XN and $KEYB_2$, and so on for all blocks of the original data file).

z=1,2... total number of DATA blocks (e), k=1,2 … total number of KEYBs generated by LFSR (q).

by applying Eq(2): for q=4 (total number of KEYBs)

$Enc_z(i,j)= E(XN(i,j) [DATA_z(i,j),KEYB_k(i,j)])$

$Enc_1(i,j)= E(XN(i,j) [DATA_1(i,j),KEYB_1(i,j)])$

$Enc_2(i,j)= E(XN(i,j) [DATA_2(i,j),KEYB_2(i,j)])$

$Enc_3(i,j)= E(XN(i,j) [DATA_3(i,j),KEYB_3(i,j)])$

$Enc_4(i,j)= E(XN(i,j) [DATA_4(i,j),KEYB_4(i,j)])$

Also apply Eq(3) to decrypt the encrypted data with the same procedure.

The number of KEYBs that will be selected to perform the encryption process depends on the total number of data blocks, and the block size of the data (DATA) depends totally on the size of the XN and KEYB.

X.K.N Algorithm:

Bellow the steps to show the algorithm of the encryption process:

1- Input the matrix of XN key
2- Input the KEYBs matrices (resulted from Keys generation process)
3- Detect the block size (m*n), m is the number of rows in key matrix; n is the number of columns in key matrix.
4- Compute the total number of DATA blocks (e)
5- Detect the total number of the KEYBs (q)
6- k=0, z=1
7- Loop

```
    If k=q then
       k=1
    Else
       k=k+1
    End if
    For i=1 to m
    For j=1 to n
       If XN[i,j]=X then
          Enc_z[i,j]=KEYB_k[i,j]  XOR  DATA_z[i,j]
       Else   (if XN[i,j]=N)
    Enc_z[i,j]=NOT (DATA_z[i,j])
       End if
    Next j
    Next i
    z=z+1
  Until (z=e)   (stop the encryption process)
```

8- Group all Enc blocks in one matrix (binary file)
9- End

In **decryption** process the same algorithm will be used but by using **Eq(3)**, with the same secret keys.

3.2 Keys generation using LFSR

Keys generation needs two keys to begin (initial KEYB and S-key). S-key with rang between (1) and (length-1) (length=m*n of the KEYB), m denotes the number of rows in KEYB, n denotes the number of columns in KEYB, s-key denotes the start point key (from this point the XOR feed back must start) in KEYB.

The proposed method supposed there are two secret keys (XN and KEYB) by default, but to increase the complexity LFSR was used.

To explain how the KEYBs will be generated, let the initial secret key ($KEYB_1$) as follows: Figure-3 shows an example of using the principle of the LFSR to generate the keys of type KEYB.

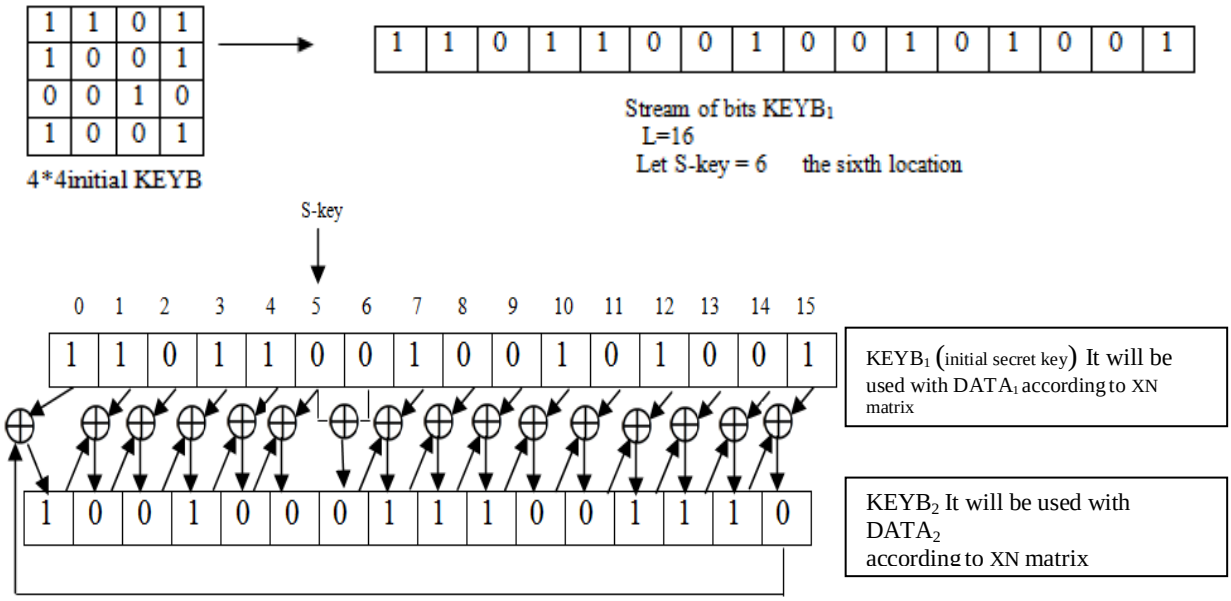


Figure 3- The mechanism of the secret key generation using LFSR principle to produce q number of KEYBs

Apply the same procedure in Figure-3 on $KEYB_2$ to generate $KEYB_3$ and so on for all KEYBs. Figure-4 shows some of the KEYBs, each one will be reconstructed as a matrix of 4*4 (in this example) and with XN matrix to encrypt the DATA matrix. Note that to produce KEYBs, the same procedure in encryption and decryption is used, also the length of the secret key (KEYB) is L bits, with the same size to XN secret key and DATA block.

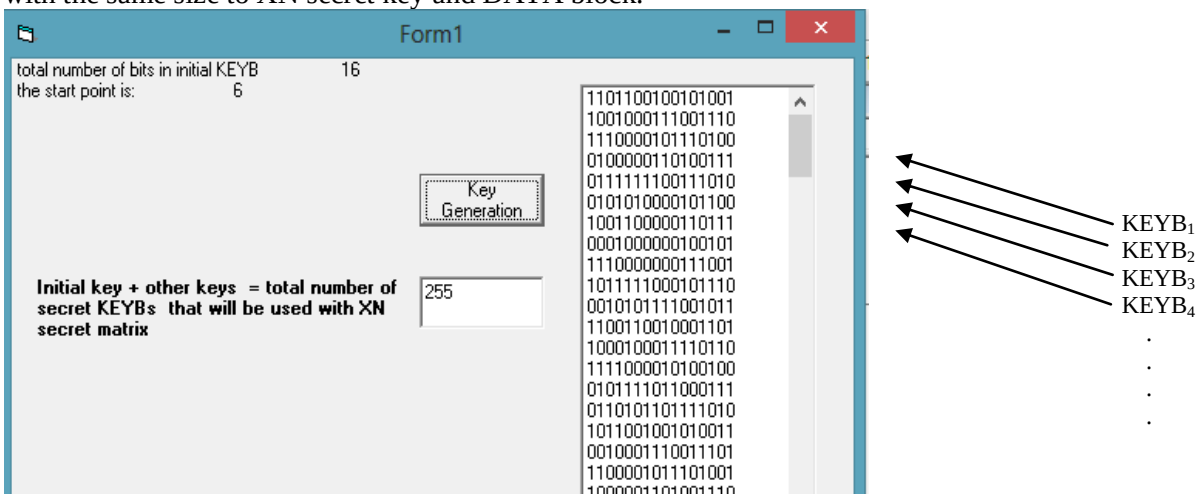


Figure 4- Secret Keys of type (KEYBs) generated by

According to Figures-3,-4, for the initial secret key 1101100100101001 there are 255 KEYBs each with length of 16 bits, where the start point (s-key) was 6 (first location in the array start from 0 to n-1). For the same initial secret key, but with start point s-key= 9, it will get the same 255 KEYBs but its sequence will differ ($KEYB_1$, $KEYB_2$,..... ,$KEYB_{255}$), s-key also must be secret as possible to complicate the prediction of the $KEYB_{k+1}$.

The procedure of KEYBs generation will stop until the last KEYB is the same as ($KEYB_1$) (the initial key).

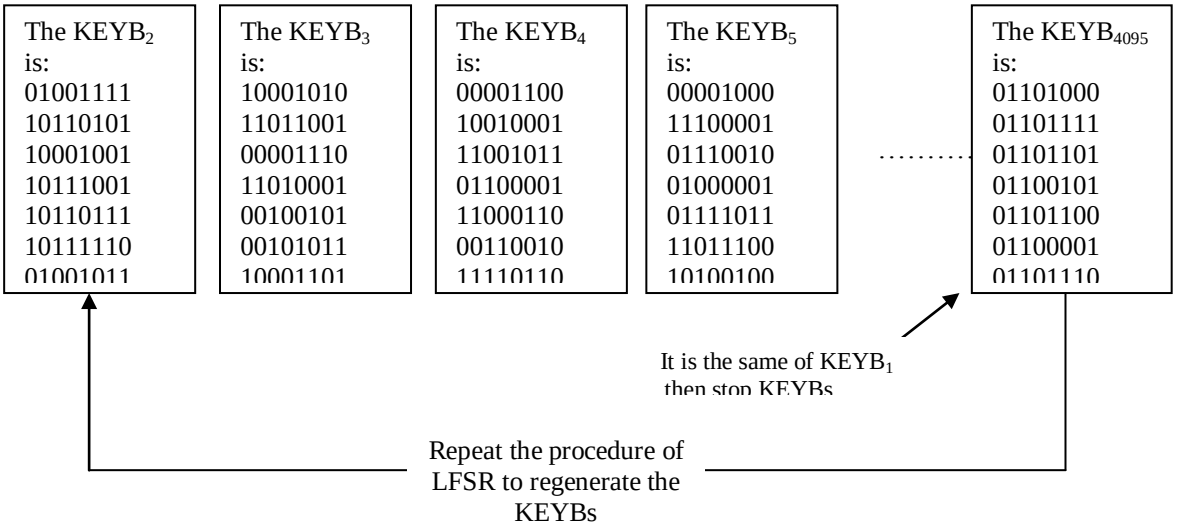
Another example, suppose the initial secret key is the word (homeland)
Ascii for h= 104, o=111, m=109, e= 101, l=108 a=97, n=110, d=100

0	1	1	0	1	0	0	0
0	1	1	0	1	1	1	1
0	1	1	0	1	1	0	1
0	1	1	0	0	1	0	1
0	1	1	0	1	1	0	0
0	1	1	0	0	0	0	1
0	1	1	0	1	1	1	0
0	1	1	0	0	1	0	0

Block of 8*8 (initial secret key ($KEYB_1$)

→ stream of bits
01101000011011110110110101100101011011000110000101101110011001000

For this stream of bits there are 4095 possible KEYBs, each one with XN matrix will apply on each $DATA_z$ matrix of size 8*8 (DATA block size), also XN must be a matrix of 8*8, L=64 bits (key size)
If s-key (start point= 17) then



When we select another s-key, $KEYB_{k+1}$ will differ, for example, for the same $KEYB_1$ in example above, let the s-key is 38, $KEYB_2$ as a stream of bits will be as follows:
1011000001001010010010011011100110110110010000011011010001000111, also there are 4095 possible KEYBs no one like the other.

According to our program and by using an initial KEYB of length 128 bits, there are 16383 KEYBs will be generated.

Keys generation algorithm:

Bellow the steps to show the algorithm of the key generation using LFSR

1- Input the matrix of the initial secret key ($KEYB_1[i,j]$) i=1 to m, j=1 to n
2- Convert the matrix to stream of bits ($KEYB_1[0..L-1]$), L=m*n
3- Detect the start point (s-key) as a secret point between 1 and (L-1), L is the length of the bits stream, s-key≠L, k=1, k will increase by one in each iteration.
4- IF s-key=1 then

Repeat
$KEYB_{k+1}[1] = KEYB_k[0]$ XOR $KEYB_k[1]$
For i=1 to L-1
$KEYB_{k+1}[i + 1] = KEYB_{k+1}[i]$ XOR $KEYB_k[i + 1]$
Next
$KEYB_{k+1}[0] = KEYB_{k+1}[L-1]$ XOR $KEYB_k[0]$
Save $KEYB_{k+1}$ in a matrix of m*n (KEYB_M), k=k+1
Until ($KEYB_{k+1} = KEYB_1$)
End IF

5- IF s-key≠1 then
Repeat
$KEYB_{k+1}[s\text{-}key + 1] = KEYB_k[s\text{-}key]$ XOR $KEYB_k[s\text{-}key + 1]$
For i = s-key + 2 To (L- 1)
$KEYB_{k+1}[i] = KEYB_{k+1}[i–1]$ XOR $KEYB_k[i]$
Next
$KEYB_{k+1}[0] = KEYB_{k+1}[L – 1]$ XOR $KEYB_k[0]$
For i = 1 To s-key
$KEYB_{k+1}[i] = KEYB_{k+1}[i-1]$ XOR $KEYB_k[i]$
Next
Save $KEYB_{k+1}$ in a matrix of m*n (KEYB_M), k=k+1
Until ($KEYB_{k+1} = KEYB_1$)
End IF

Note that, the algorithm of the key generation will be used in encryption phase and decryption phase with the same sequence to produce all the KEYBs

3.3 The Encryption phase

In the sender site the plain text file must be preprocessed (binary file, block***s*** of size m*n (DATA matrices)) and then the encryption stage will begin, Figure-5 shows the applied of the two secret keys (KEYB,XN) as a blocks on the DATA blocks, also the encrypter will use pre-determine (s-key) and applying the LFSR to generate the KEYBs.

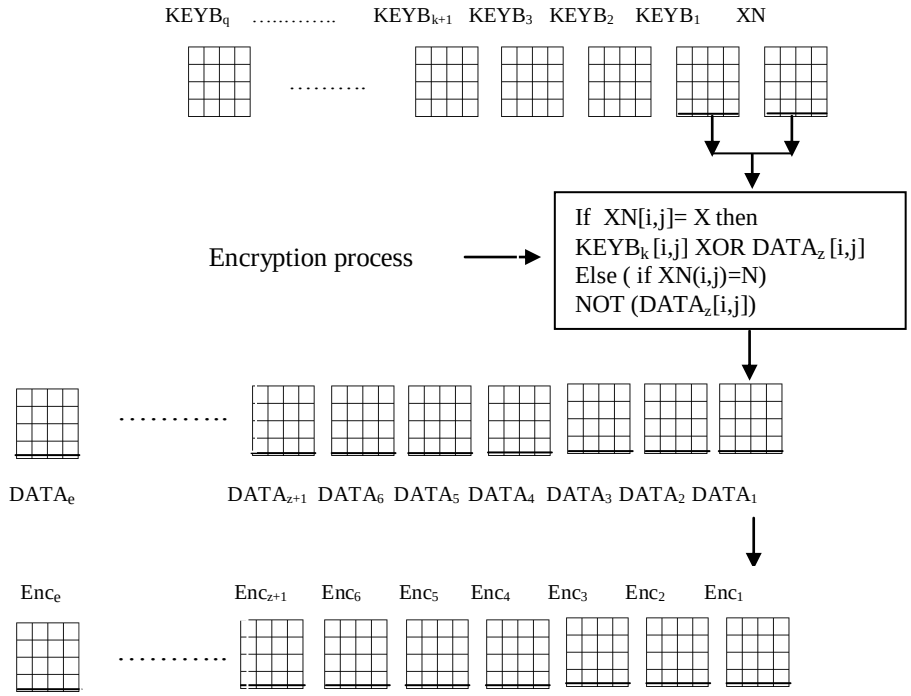


Figure 5- The Encryption stage

DATA, Enc, KEYB, and XN are blocks with the same size, Enc is the resulted blocks of the encryption stage, by gathering them to reconstruct the encrypted file according to the same sequence, and finally the encrypted file is ready for sending to the receiver.

As shown in the above section, each cell in XN is (XOR) or (NOT) gate. Figure-6 shows how the two keys (XN,KEYB) will work with the DATA matrix (of the two characters AI) to get the cipher text as binary (Enc1). Whereas the block size is 4*4 as an example, also this example will explain one iteration of Figure-5 if the string is more than two characters.

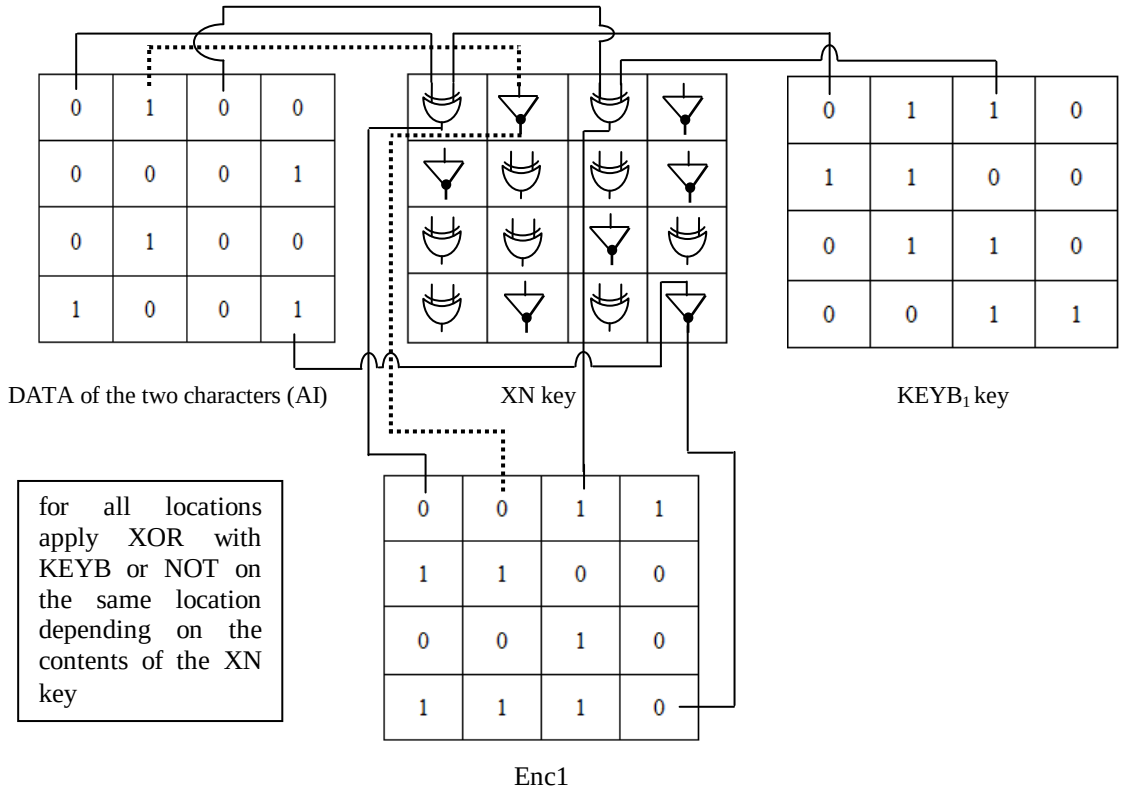
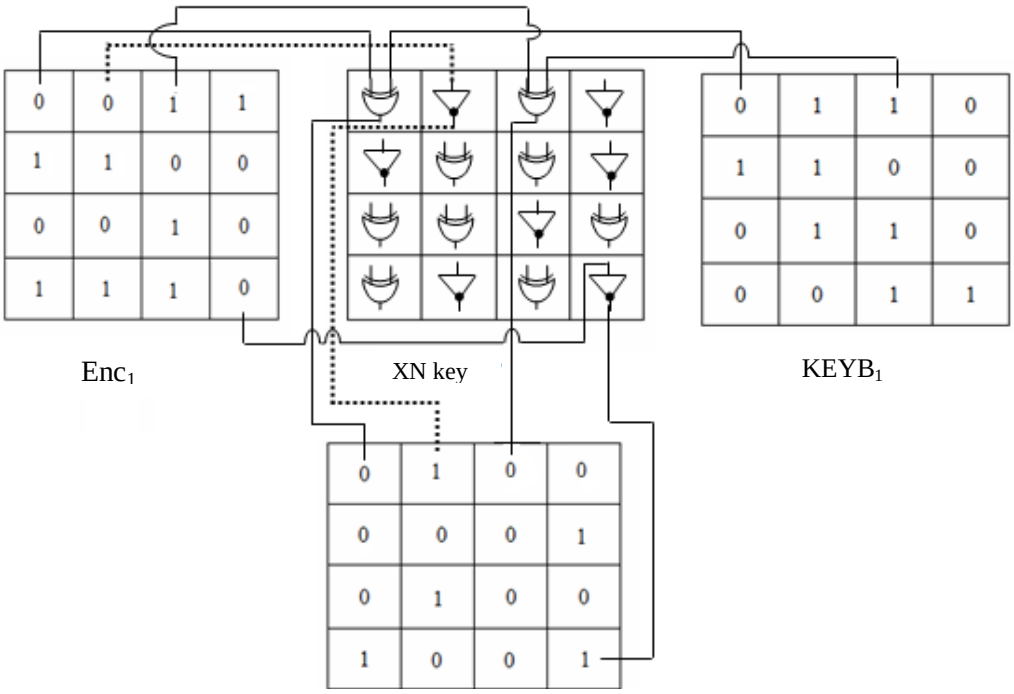


Figure 6- Example of applying the X.K.N encryption with blocks size 4*4

For long string, the next block of data ($DATA_2$), the same XN with next KEYB ($KEYB_2$) to get Enc_2 block and so on.

3.4 The Decryption phase

To begin the decryption stage, the decrypter needs to know the XN, KEYB, and the start point (s-key) to regenerate the KEYBs using LFSR. For the same example in section 3-2 the Enc1 was (0011110000101110), use the same mechanism of the encryption but in reverse, Figure-7 shows how the two keys (XN,KEYB) will work with Enc_1 matrix to get the plaintext (AI) as a binary, also the decryption stage will use $KEYB_2$ that generated by LFSR with $DATA_2$ if the data is more than two characters with proposing the blocks sizes are 4*4. The DATA blocks must be suitable with size of the secret keys (XN and KEYB matrices) to make easy of applying the procedure of encryption and decryption. If the length (matrix size) of the secret key increase then the number of the KEYBs that generated by the proposed LFSR will be increased and the probability of selecting the s-key also will be increased. To solve the problem of unfitting between the last block ($DATA_e$) and XN, KEYB matrices it is good to use the padding (extra bits of 0) to the end of the $DATA_e$ block.



DATA of the two characters (AI)

When some one try to break the encrypted data must to take two factors, cost and time [6, 7], the first factor related to the encrypted data itself, the second factor will be brute force attack (trying all possible keys).There is an average total time to try all the keys [6]. Now it must to compute the average time required to break all the keys. Before beginning the times computations, lets first to detect the following points:

1- How many bits in KEYB

2- How many cells in XN key, normally the same number of bits in KEYB.

3- What is the start point (s-key)

To understand these three points lets take the following example

KEYB= 64 bits (block of 8*8) XN= 64 cell (of XOR and NOT gates), if the KEYB is 64 bits then there are 63 possible start point (s-key), from these three keys one will conclude the following equation:

Number of all possible keys= $(2^{64*64})*63$ =

$6.5796499529028607921580420775147*10^{1234}$ possible keys

Note: s-key must not to be 64.

If each one possible key take a time of 1 micro second the total time to check all possible keys will be:

$1.269222598939595060215671697051 5*10^{1223}$ years

Therefore,

Total time for Brute force =$(2^{No.\ of\ KEYB\ bits*No.\ of\ XN\ cells})$*No. of possible start points **(4)**

Where, *No. of possible start points = number of KEYB bits -1*

Also making the selection of the XN and KEYB dimensions as a dynamic will increase the security of the proposed method.

For these reasons, the proposed method (X.K.N) will be very secure.

5- Experimental results

To understand the mechanism of the proposed method, the following data will be encrypted: "Hello the meeting will be in RUC" which represent a block of 16*16 bits, whereas, XN and KEYB are blocks of size 8*8, and the start point (s-key) is 37, sections (5.1 and 5.2) shows the encryption and decryption phases.

5.1 Encryption phase:

This section explains how the data will be encrypted depending on the information from section (5) in the above.

0100100001100101
0110110001101100
0110111100100000
0111010001101000
0110010100100000
0110110101100101
0110010101110100
0110100101101110
0110011100100000
0111011101101001
0110110001101100
0010000001100010
0110010100100000
0110100101101110
0010000001010010
0101010101000011

DATA matrix of 16*16 bits (Binary plain text)

X	X	N	N	N	X	X	N
N	N	X	X	N	N	X	X
N	N	X	N	X	X	N	X
N	X	X	N	X	X	N	X
X	X	X	N	N	X	N	N
X	X	N	X	N	X	X	N
N	N	X	X	X	N	X	N
X	N	X	N	X	N	N	X

XN key

1	0	0	0	1	1	0	1
1	0	1	0	1	0	1	1
0	0	1	1	1	0	0	0
0	1	0	1	0	1	0	1
1	0	1	1	0	1	1	1
0	1	0	0	1	1	0	1
1	1	0	0	0	1	1	0
1	0	1	1	1	0	1	1

KEYB key

From the KEYB and by applying the LFSR procedure (key generation) there are 4095 keys ($KEYB_k$), and from these 4095 keys only the first four keys will be enough for our DATA

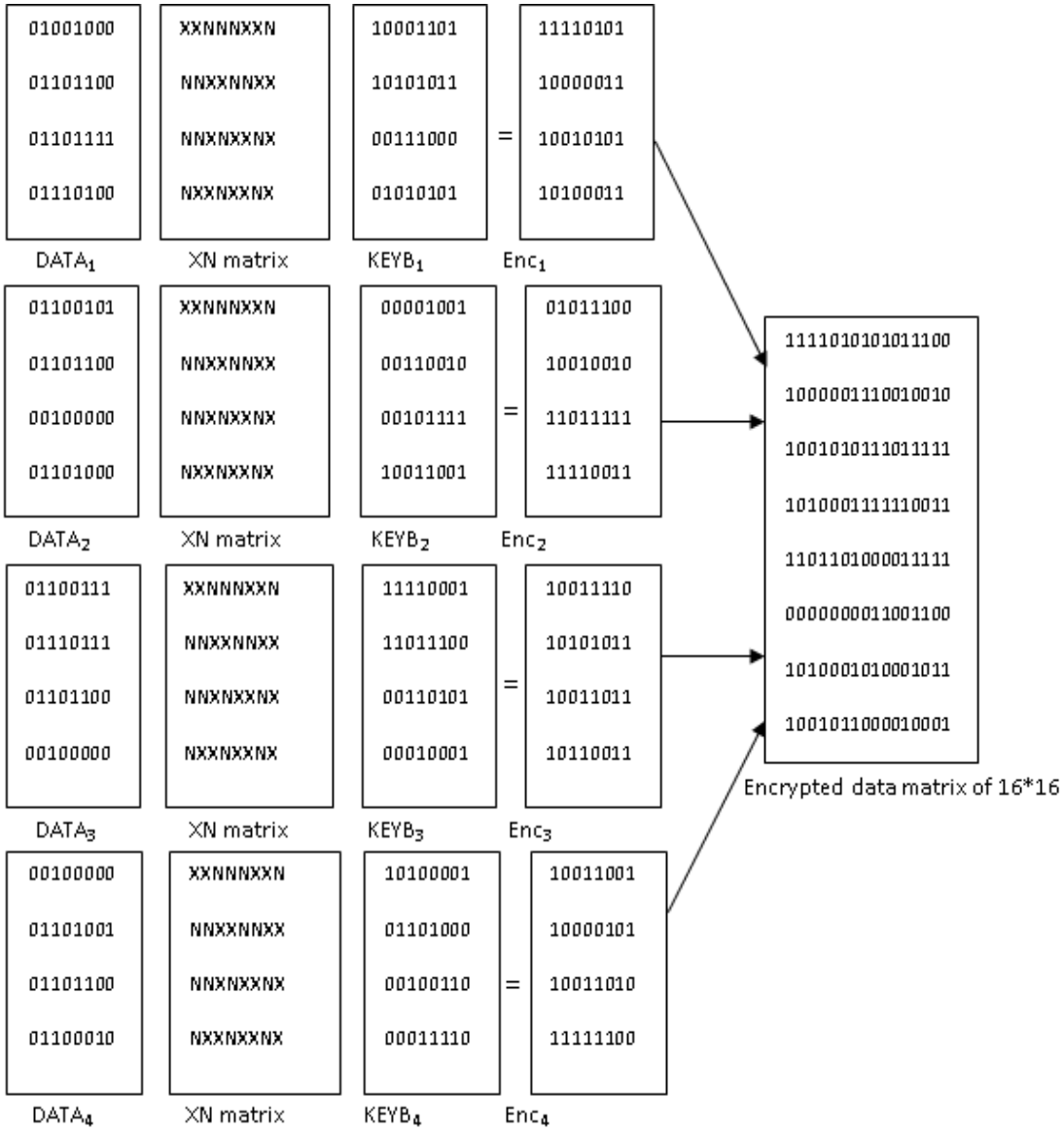
$KEYB_1$= 1000110110101011001110000101010110110111010011011100011010111011 (initial secret key)

$KEYB_2$= 0000100100110010001011111001100100100101100010010111101100101101

$KEYB_3$= 1111000111011100001101010001000111000110111100011010110111001001

$KEYB_4$= 1010000101101000001001100001111010000100101000010011011010001110

Now by applying each one of these four keys as a matrix of 8*8 with XN key on the DATA matrix (4 blocks of 8*8) to get the following encrypted data (Enc matrix)

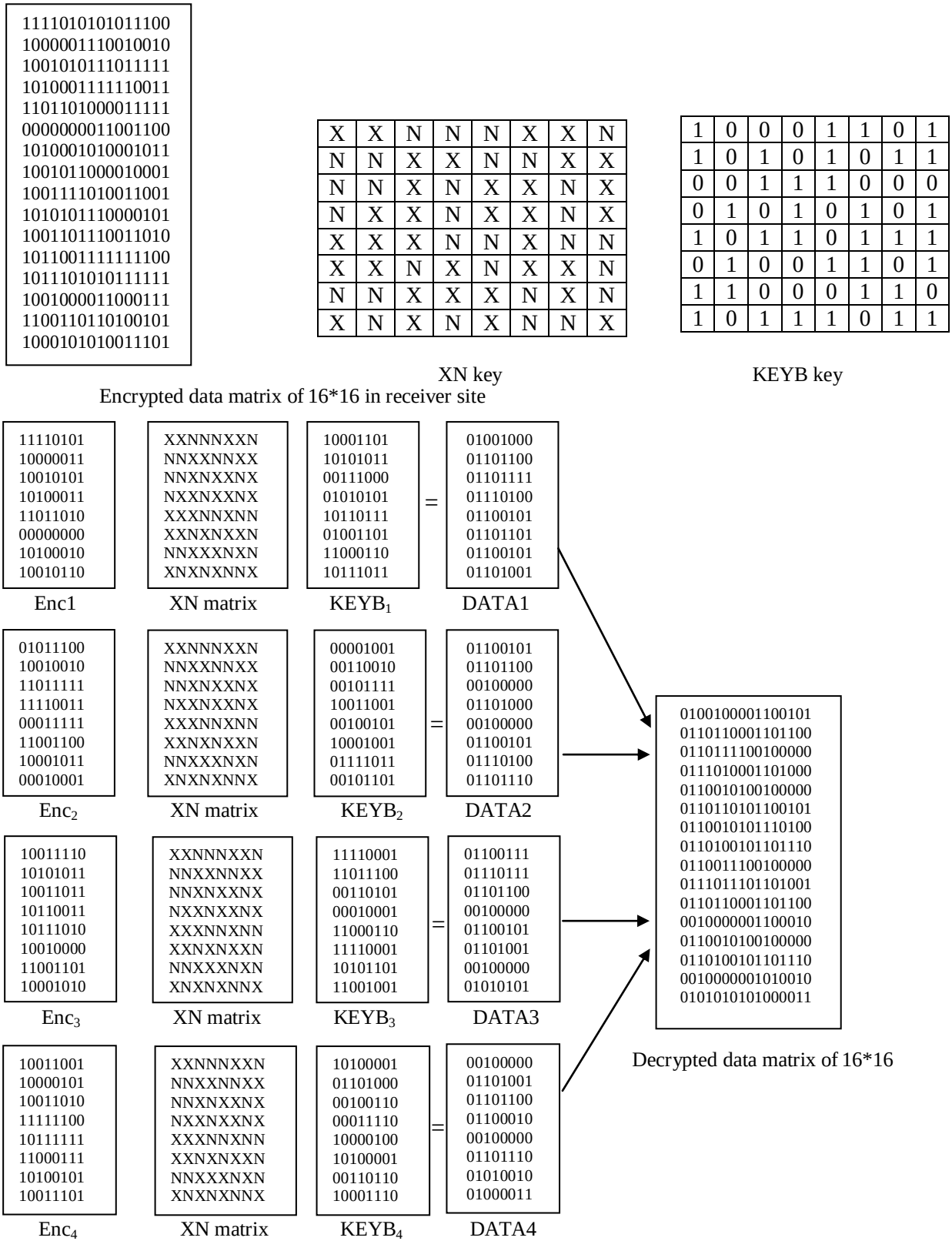


5.2 Decryption phase:

With the same mechanism of the encryption in section 5.1, this section tries to decrypt the encrypted data that resulted from the encryption phase using the same XN key and KEYB. Also apply the LFSR to generate the $KEYB_i$ that shown in encryption phase, the start point was 37 (s-key), also the decryption phase will use only the first four KEYBs.

Note 1: the procedure of the proposed strategy to generate KEYBs that derived from LFSR is the same in encryption and decryption using the same initial KEYB and start point.

Note 2: to overcome the problem of intruders or hackers the XN key, KEYB, and s-key (three secret keys) must be sent using a secure channel.



After decoding the decrypted data matrix and by using the ASCII code for each eight bits, the original plain text is ("Hello the meeting will be in RUC").

6- Conclusions

Everyone knows any transaction between any two nodes it already will be compelling to encryption strategies. And from this proposed method (X.K.N) it is possible to conclude the following conclusions:

1. Using X.K.N, there is no limitation in the size of the XN, KEYB,S-key, and data size.
2. S-key was used in X.K.N to increase the security because it increases the probabilities and variations of the KEYBs.
3. The proposed method is new strategy in data encryption.
4. LFSR (linear feedback shift register) is good method to generate *q* number of KEYBs, also the key generation algorithm adopted in this paper may be itself as a simple proposed method for keys generating using the start point.
5. The proposed method behaves as a stream cipher in some phases and as block cipher in other phases.
6. According to the proposed key generation strategy that depends totally on LFSR, there is no standard to the number of generating keys, with the same number of bits and for each group of bits string there are *q* numbers of KEYBs.
7. If the number of the data file blocks less than the number of KEYBs then the proposed method will correspond to be as a one time pad method (one KEYB for each block with the same XN).
8. The time required for encryption and decryption is the same, and the file size before and after encryption is the same.
9. There are no complex mathematical computations that may require long time to execute, X.K.N depends on conditional states to determine XOR or NOT and then applying the determine operation, and this logically will be very fast.
10. It is good to recommend to make the dimensions of the two secret keys (XN and KEYB) as dynamic matrices (m*n) (m not equal to n), and this will increase the security of the proposed method.

Acknowledgment

I would like to thank and express my gratitude to Prof. Dr. Hussein Kettan Alkafaji for his help and support throughout preparing this paper.

References:

1. Denning, D., E. **1982**. *Cryptography and Data Security*, Addison-Wesley.
2. Sravan D. Kumar, Suneetha, C.H. and Chandrasekhar. A. **2011**. A Block Cipher Using Rotation and Logical XOR Operations, *IJCSI International Journal of Computer Science*, 8(6).
3. Christof, P. and Jan, P. **2010**. *Understanding Cryptography*, Springer-Verlag Berlin Heidelberg.
4. Al-Suffar, A., M., M. **2015**. Implementing a New Serial Control As Nonlinear Function For Key Stream Pseudo-Random Number Generator, *Journal of Al-Rafidain University College*, 35.
5. Hathwalia, S., Yadav, M. **2014**. Design and Analysis of 32 Bit Linear Feedback Shift Register Using VHDL, *Journal of Engineering Research and Applications*, 4(6), (version 6).
6. Stallings, W. **2006.** *Cryptography and Network Security*, Fourth Edition, Prentice Hall.
7. Pfleeger, C., P. **1989.** *Security in Computing*, Prentice Hall.