6 X 6 Playfair Cipher using LFSR based Unique Random Number Generator

Amandeep Kaur, Harsh Kumar Verma, Ravindra Kumar Singh
Department of Computer Science and Engineering
Dr. B. R. Ambedkar National Institute of Technology, Jalandhar (India)

ABSTRACT

Playfair cipher is the well-known multiple letter encryption cipher. Here the digraphs in the plaintext are treated as single units and converted into corresponding cipher text digraphs. However because of the drawbacks inherent in the 5 X 5 Playfair cipher which adversely affects the security we proposed a 6 X 6 Playfair cipher and then coupled it with Linear Feedback Shift Register based Unique Random Number Generator [1]. 6 X 6 Playfair cipher supports all 26 alphabets (A-Z) and 10 digits (0-9) which eliminate the limitation of 5 X 5 Playfair in which "i" and "j" both character could not appear at the same time [2, 3]. LFSR not only enhances the security up to a considerable level by generating random sequences but also provides a much faster rate of encryption and decryption [1], that's why LFSR based Unique Random Number Generator is chosen for the consideration. This paper deals in with the security issues of the new proposed system. Various types of cryptography attacks have been taken under consideration for original Playfair cipher but not vulnerable for this proposed cipher.

Keywords- Playfair cipher, Random number, LFSR, Polyalphabetic cipher

1. INTRODUCTION

The relationship of Cryptography and random numbers are investigated [4, 5]. Linear Feedback Shift Register based Unique Random Number Generator is a good candidate for generating random numbers because logical circuit variations are high and its software and hardware implementation is also very easy [1]. We can easily produce different random numbers by just changing the seed or taps of LFSR.

This paper presents a new approach for encryption which uses 6 X 6 Playfair cipher with LFSR based unique random number generator. In 6 X 6 Playfair cipher, the alphabets and numerals are arranged in 6 X 6 matrix based on secret key.

Table 1: 6 X 6 Playfair matrix based on secret key "NITJ"

N	I	T	J	0	1
2	3	4	5	6	7
8	9	A	B	C	D
E	F	G	H	K	L
M	O	P	Q	R	S
U	V	W	X	Y	Z

Even though it is very difficult to break the ciphertext but it can be breakable by few hundreds of letters [6, 7]. So random numbers are taken into consideration for eliminate this limitation [8]. In our approach, 36 unique random numbers generated by LSFR based unique random number generator are arranged into 6 X 6 matrix.

Table 2: 36 unique random numbers arranged into 6 X 6 matrix

20	57	15	10	35	49
56	30	19	50	39	37
31	7	3	60	18	33
42	17	62	61	11	44
55	6	36	23	59	13
8	41	47	2	34	25

Then it is used for mapping with the 6 X 6 Playfair matrix to transmit random numbers instead of ciphertext.

Table 3: Mapping Table (Table 1 & Table 2)

N – 20	I – 57	T – 15	J – 10	0 – 35	1 – 49
2 – 56	3 – 30	4 – 19	5 – 50	6 – 39	7 - 37
8 – 31	9 – 7	A – 3	B – 60	C – 18	D – 33
E – 42	F – 17	G – 62	H – 61	K – 11	L – 44
M – 55	O – 6	P – 36	Q – 23	R – 59	S – 13
U – 8	V – 41	W – 47	X – 2	Y – 34	Z – 25

At first plaintext digraphs are converted into corresponding ciphertext digraphs based on Playfair cipher rules then these ciphertext digraphs are used to find out the corresponding random numbers by using the mapping table and then these random numbers are used in transmission for enhancing the security.

2. 6 X 6 Playfair Cipher

6 X 6 Playfair cipher is the multiple letter encryption cipher, which encrypts digraphs of plaintext into corresponding cipher text digraphs. For that purpose it requires a 6 X 6 matrix to store alphabets and numerals. These alphabets and numerals are arranged in 6 X 6 matrix based on secret key. 6 X 6 Playfair cipher has mainly 3 algorithms, Key-Matrix Generation, Encryption and Decryption. These are described below-

2.1Key-Matrix Generation

6 X 6 Playfair Cipher makes use of 6 X 6 matrix (table), which is used to store a keyword that becomes the key for encryption and decryption. The way this is entered into 6 X 6 matrix is based on some simple rules, as below.

1. Enter the secret (password) which may contain numerals and alphabets like: aman2012nitj, ravindra1987singh, cipher, 29101989 etc.

2. Find out the keyword by dropping the duplicate letters of key. Ex: amn201itj, ravind1987sgh, cipher, 29108 for above keys.
3. Arrange the keyword in 6 X 6 matrix row-wise: left to right and then top-to-bottom.
4. Fill the remaining spaces in the matrix with the rest of numerals (0-9) and alphabets (A-Z) that were not the part of our keyword.

This is illustrated with the secret "FRIENDS4EVER" then keyword will be "FRIENDS4V" and Key-Matrix will be :

Table 4: 6 X 6 Playfair Key-Matrix based on secret key "FRIENDS4EVER"

F	R	I	E	N	D
S	4	V	0	1	2
3	5	6	7	8	9
A	B	C	G	H	J
K	L	M	O	P	Q
T	U	W	X	Y	Z

2.1 Encryption

To encrypt a message, one would break the message into digraphs (groups of 2 letters). If both letters are the same or only one letter is left, add a filler letter "x" after the first letter. So that "**BALLOON"** would be treated as "BA" "LX" "LO" "ON" and "HELLOWORLD" would be treated as "HE" "LX" "LO" "WO" "RL" "DX". Then apply the following 3 rules, in order, to each diagraph (pair of letters) in the plaintext:

1. If both letter appear on the same row of Key-Matrix, replace them with the letters to their immediate right respectively (wrapping around to the left side of the row if a letter in the original pair was on the right side of the row).
2. If both letter appear on the same column of Key-Matrix, replace them with the letters immediately below respectively (wrapping around to the top side of the column if a letter in the original pair was on the bottom side of the column).
3. If both letters didn't fall on the same row or column, replace them with the letters on the same row respectively but at the column of other letter of pair.

2.2 Decryption

Apply the following 3 rules, in order, to each diagraph (pair of letters) in the ciphertext to find the digraph:

1. If both letter appear on the same row of Key-Matrix, replace them with the letters to their immediate left respectively (wrapping around to the right side of the row if a letter in the original pair was on the left side of the row).
2. If both letter appear on the same column of Key-Matrix, replace them with the letters immediately up respectively (wrapping around to the bottom of the column if a letter in the original pair was on the top of the column).
3. If both letters didn't fall on the same row or column, replace them with the letters on the same row respectively but at the column of other letter of pair.

Remove the filler letter from the digraphs (Dropping any extra "X"s that don't make sense in the final message when finished) to find out the actual text (plaintext).

3. LINEAR FEEDBACK SHIFT REGISTER

A Linear Feedback Shift Register is a shift register whose input state is a linear function of its previous state [9,10]. The only linear functions of single bits are XOR and inverse-XOR; thus it is a shift register whose input bit is driven by the exclusive-or (XOR) of some bits of the overall shift register value [11]. The L-bit initial value of LFSR is called **seed** where L is called its length, the stream values produced by the register is completely determined by previous state [12]. It can produce various random sequences by varying the **taps** [13, 14].The bit position that affects next state is called tap. This is illustrated as follows [15].

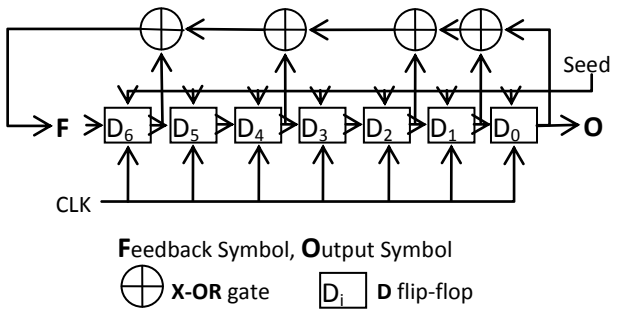

Figure 1. **LFSR for Tap = [0, 1, 2, 4, 6], length (L) = 7**

In this circuit, at each pulse, the state of the flip-flop is shifted to the next one down the line and also computes Boolean function of the state of the flip-flops [9, 16].

If Tap = {0, 2, 3, 5, 6} and Seed = 51 whose binary equivalent is 0110011. Then-

Table 5: Generated Sequence By LFSR

Feedback Symbol	State of Shift Register							Output Symbol
	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
	0	**1**	***1***	**0**	**0**	***1***	**1**	
0	0	0	*1*	1	0	*0*	1	1
0	0	0	*0*	1	1	*0*	0	1
0	0	0	*0*	0	1	*1*	0	0
1	1	0	*0*	0	0	*1*	1	0
0	0	1	*0*	0	0	*0*	1	1
0	0	0	*1*	0	0	*0*	0	1
0	0	0	*0*	1	0	*0*	0	0
1	1	0	*0*	0	1	*0*	0	0
0	0	1	*0*	0	0	*1*	0	0
1	1	0	*1*	0	0	*0*	1	0
0	0	1	*0*	1	0	*0*	0	1
0	0	0	*1*	0	1	*0*	0	0
1	1	0	*0*	1	0	*1*	0	0
0	0	1	*0*	0	1	*0*	1	0
1	1	0	*1*	0	0	*1*	0	1
1	1	1	*0*	1	0	*0*	1	0
0	0	1	*1*	0	1	*0*	0	1
0	0	0	*1*	1	0	*1*	0	0
1	1	0	*0*	1	1	*0*	1	0
0	0	1	*0*	0	1	*1*	0	1

0	0	0	*1*	0	0	*1*	1	0
1	1	0	*0*	1	0	*0*	1	1
1	1	1	*0*	0	1	*0*	0	1
1	1	1	*1*	0	0	*1*	0	0
0	0	1	*1*	1	0	*0*	1	0
1	1	0	*1*	1	1	*0*	0	1
1	1	1	*0*	1	1	*1*	0	0
0	0	1	*1*	0	1	*1*	1	0
1	1	0	*1*	1	0	*1*	1	1
1	1	1	*0*	1	1	*0*	1	1
1	1	1	*1*	0	1	*1*	0	1
1	1	1	*1*	1	0	*1*	1	0
0	0	1	*1*	1	1	*0*	1	1
0	0	0	*1*	1	1	*1*	0	1
0	0	0	*0*	1	1	*1*	1	0
1	1	0	*0*	0	1	*1*	1	1
1	1	1	*0*	0	0	*1*	1	1
1	1	1	*1*	0	0	*0*	1	1
1	1	1	*1*	1	0	*0*	0	1
1	1	1	*1*	1	1	*0*	0	0
0	0	1	*1*	1	1	*1*	0	0
1	1	0	*1*	1	1	*1*	1	0
0	0	1	*0*	1	1	*1*	1	1
0	0	0	*1*	0	1	*1*	1	1
0	0	0	*0*	1	0	*1*	1	1
0	0	0	*0*	0	1	*0*	1	1
0	0	0	*0*	0	0	*1*	0	1
0	0	0	*0*	0	0	*0*	1	0
1	1	0	*0*	0	0	*0*	0	1
1	1	1	*0*	0	0	*0*	0	0
0	0	1	*1*	0	0	*0*	0	0
1	1	0	*1*	1	0	*0*	0	0
0	0	1	*0*	1	1	*0*	0	0
1	1	0	*1*	0	1	*1*	0	0
0	0	1	*0*	1	0	*1*	1	0
1	1	0	*1*	0	1	*0*	1	1
1	1	1	*0*	1	0	*1*	0	1
1	1	1	*1*	0	1	*0*	1	0
0	0	1	*1*	1	0	*1*	0	1
0	0	0	*1*	1	1	*0*	1	0
1	1	0	*0*	1	1	*1*	0	1
1	1	1	*0*	0	1	*1*	1	0
0	**0**	**1**	***1***	**0**	**0**	***1***	**1**	1

Generated sequence will be (1 1 0 0 1 1 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 1 1 0 0 1 0 0 1 1 1 0 1 1 0 1 1 1 1 0 0 0 1 1 1 1 1 0 1 0 0 0 0 0 0 1 1 0 1 0 1 0 1) [1]

LFSR can also be used as a random number generator [17, 18]. By using the range (R) of random number it can be determined that how many bits (B) will be grouped together to represent a random number by the formula-

$$2^B >= R > 2^{B-1} \quad (1)$$

If the range of random number is 0 to 63 then number of bits will be 6 to represent the random number. Generated sequences of random numbers are repeated to generate the required count of random numbers [19].

The binary value of produced random number from above LSFR is 110011, 000010, 001010, 010110, 010011, 101101, 111000, 111110, 100000 and 011010 while 101 is left, it will be used in next repetition on the top of generated sequence. So next values will be 101110, 011000, 010001, 010010, 110010, 011101, 101111, 000111, 110100, 000011 and 010101, these numbers will be repeated again and again to generate the specified count of random numbers [20].

But the fastest moving era of computer science demands the non repeating random numbers in some applications. At those situations the existing approach can not satisfy the demand, that's why LFSR Based Unique Random Number Generators came in the focus.

4. LFSR BASED UNIQUE RANDOM NUMBER GENERATOR

The proposed algorithm ensures to generate the specified count of unique random numbers by using linear feedback shift register with some associated rules [1].

4.1 Basic Terms

There are some basic terms required in LFSR based Unique Random Number Generator.

4.1.1 Required Registers(R)

Number of required registers in LFSR based Unique Random Number Generator is depends on the number of letters exist in that number system. If there are N letters in any number system then it will need "R" registers. Where-

$$2^R >= N > 2^{R-1} \quad (2)$$

Ex: ascii codes requires 7 registers, (2^7 = 128). Because there are total 128 letters in ascii number system. A letter may be alphabet (case sensitive), numeral or a special symbol [1].

4.1.2 Relevant Values

A relevant value is assigned to all letters exist in the number system. Ex: ascii codes are that relevant numbers for ascii number system which contain 128 letters [1].

4.1.3 Seed Sequence

Seed sequence is also depends upon the keyword. Keyword's relevant values stores in first come first serve manner by removing duplicates is called **Seed Sequence** [1].

4.1.4 Tap Sequence

Tap sequence depends upon the keyword. $\text{Mod}_{\mathbf{R}}$ of keyword's relevant values stores in first come first serve manner by removing duplicates is called **Tap Sequence** [1].

4.1.5 Length (L) of Unique Random Number

Length of unique random number is totally depending on the total count of required unique random number (U), which can be calculated by the given formula [1]-

$$2^L >= U > 2^{L-1} \quad (3)$$

4.2 Algorithm

These are the following steps to produce unique random numbers-

*4.2.1 **Unique Random Number Generator** (Keyword, N, OS[N][2], U, URnum[U], R, L, i, j, k, m, x, y, z, w, flag)*

This algorithm is used for generating U count of unique random numbers by using user's Keyword. URnum is a linear array of U

length for storing U cout of unique random number where OS is a 2-D array which stores output symbols of LFSR [1].

{

1. [Initialize]
   y = 0, z = 0, w = 0, flag = 0;
2. [Find the number of **Required Registers**(**R**) of that number system.]
   2^R >=N > 2^{R-1};
3. [Find the **Length (L)** of Unique Random Number.]
   2^L >U >= 2^{L-1};
4. Find the **Seed Sequence** of that keyword.
5. Using mod_R operations find out the **Tap Sequence** of the keyword.
6. Fetch the first value from **Seed Sequence** for the **Seed** of **LFSR** and delete it from the **Seed Sequence**.
7. Use the **Tap Sequence** as the **Tap** for the LFSR.
8. Generate the Output symbols from **LFSR** (described detailed in next algorithm) until the seed gets repeated. And store it in OS**[k][0]** where **k** = 0 to number of output symbols (x) generated by LFSR while OS[k][1] will be 0 for all.
9. do
   {
   z = flag = 0;
   if (OS[y][1] = = 0)
   {
   OS[y][1] = 1;
   m = y;
   for (k = L-1 to 0)
   {
   z = z + (2^k) * (OS[y][0])
   y = ((y+1) % x);
   }
   for (j = 0 to w)
   {
   if (URnum[j] = = z)
   flag = 1;
   }
   if (flag = = 0)
   {
   URnum[w] = z;
   z = 0;
   w = w + 1;
   if (w = = U)
   goto **13;**
   }
   else
   {
   y = (m + 1) % x;
   flag = 0;
   }
   }
   else
   {
   y = (y + 1) % x;
   }
   }
   until (OS[i][1] = = 0)
10. y = 0, x = 0;
11. if (length [Tap Sequence] > 2) then remove first value of Tap Sequence and consider it as the seed. goto 8;
12. goto 6;
13. return URnum[U];

}

*4.2.2 **LFSR** (LFSRT[R], Tap[Lt], Seed[R], Lt, R, w, i, j, k)*

This algorithm is used to generate the random Output Symbols by using Seed[R] and Tap[Lt], Lt represents the length of Tap. LFSRT is a linear array of R length for storing the next Seed [1].

```
{
w = 0;
for (i = 0 to Lt)
{
j=Tap[i];
w = X-OR (w , LFSRT[(R-1) - j]);
}
OS[k][0]=LFSRT[R-1];
OS[k][1]=0;
for (i = R-2 to 0)
{
LFSRT[i+1]=LFSRT[i];
}
LFSRT[0] = w;
k = k + 1;
} until (Seed gets repeated);
```

5. Result and Analysis

6 X 6 Playfair cipher requires only 36 unique random numbers so it requires 6 registers (2^R >= N). Where R is the no of register required and N denotes the total unique random numbers needed (so 2^6 >= 36).

5.1 Tap Sequence

If keyword is "FRIENDS4V" then tap sequence will be:
F≈15 % 6 =3
R≈27 % 6= 3
I≈18 % 6 = 0
E≈14 % 6= 2
N≈23 % 6= 5
D≈13 % 6= 1
S≈28 % 6 = 4
4 ≈ 4 % 6 = 4
V ≈ 31 % 6 = 1
Tap sequence: {3, 0, 2, 5, 1, 4}

5.2 Seed Sequence

If keyword is "FRIENDS4V" then seed sequence will be:
F≈15
R≈27
I≈18
E≈14
N≈23
D≈13
S≈28
4 ≈ 4
V ≈ 31

Seed sequence: {15, 27, 18, 14, 23, 13, 28, 4, 31}

5.3 Unique Random Numbers

If keyword is "FRIENDS4V" then Tap sequence will be {3, 0, 2, 5, 1, 4} and Seed sequence will be {15, 27, 18, 14, 23, 13, 28,

4, 31}. So LFSR based Unique Random Number Generator will generate the following random numbers:

60, 30, 15, 7, 35, 49, 56, 57, 19, 50, 39, 37, 31, 10, 62, 20, 41, 33, 42, 17, 61, 3, 18, 11, 59, 6, 36, 23, 55, 13, 8, 47, 44, 26, 34, 25

There are 36 letters so there may be 36 X 36 = 1296 unique matrix that is very difficult to identify the particular structure. It can be cracked if there is enough text by known plaintext attack method. Involvement of random numbers reduces this drawback up to an acceptable limit by hiding the actual structure from the intruder. This may possible by using random numbers for transmission of user message. For this technique 36 unique random numbers are required for mapping this actual structure to those random numbers. These random numbers will be arranged in 6 X 6 matrix for the mapping purpose. For example if 6 X 6 random number matrix is-

Table 6: 36 unique random numbers generated on keyword "FRIENDS4V"

60	30	15	7	35	49
56	57	19	50	39	37
31	10	62	20	41	33
42	17	61	3	18	11
59	6	36	23	55	13
8	47	44	26	34	25

5.4 Mapping Table

If keyword is "FRIENDS4V" then Key-Matrix will be according to table 4. So the mapping table will be (Table 4 with Table 6):

Table 7: Mapping table for keyword "FRIENDS4V"

F – 60	R – 30	I – 15	E – 7	N – 35	D – 49
S – 56	4 – 57	V – 19	0 – 50	1 – 39	2 - 37
3 – 31	5 – 10	6 – 62	7 – 20	8 – 41	9 – 33
A – 42	B – 17	C – 61	G – 3	H – 18	J – 11
K – 59	L – 6	M – 36	O – 23	P – 55	Q – 13
T – 8	U – 47	W – 44	X – 26	Y – 34	Z – 25

5.5 Encryption

If keyword is "FRIENDS4V" then encryption will we as following-

Plaintext: HELLO
HE - GN
LX - OU
LO - MP
Ciphertext:
G, N = 3, 35
O, U = 23, 47
M, P = 36, 55
The transmitted ciphertext is {(3, 35), (23, 47), (36, 55)} instead of alphabetical letter.

5.6 Decryption

If keyword is "FRIENDS4V" then encryption will we as following-

Ciphertext: transmitted ciphertext is {(3, 35), (23, 47), (36, 55)}
3, 35 = G, N
23, 47 = O, U
36, 55 = M, P
Ciphertext:
GN = HE
OU = LX
MP = LO
The actual plaintext will be "HELLO" after ignoring the filling character.

6. CONCLUSION

This proposed cipher rapidly increases the security of the ciphertext by transferring random numbers on behalf of the actual ciphertext letters. The proposed algorithm make use of LFSR based Unique Random Number Generator, that can be used to generate unpredictable different random sequences by varying seed sequence and tap sequence. Inner structure of LFSR is very simple, and its quiet easy and cost effective to implement it on hardware and Software. It is considerably fast compare to other methods so it will be best for low bandwidth less memory storage small applications where only alphabets and numeric values are required to be protected.

7. REFERENCES

[1] Harsh Kumar Verma, Ravindra Kumar Singh, "Linear Feedback Shift Register based Unique Random Number Generator" *International Conference* on Electrical Engineering and Computer Science, Goa (India), April 7th 2012.

[2] William Stallings, Cryptography and Network Security Principles and Practice. Second edition, Pearson Education.

[3] Behrouz A. Forouzan, Cryptography and Network Security. Special Indian Edition, The McGraw- Hill companies, New Delhi,2007.

[4] Menezes AJ, Oorschot PCV, Vanstone SA, Handbook of applied cryptography. Boca Raton, Florida, USA: CRC Press; 1997.

[5] Johannes A.Buchmann, Introduction to Cryptography. Second Edition, Springer –Verlag NY, LLC, 2001.

[6] Dhiren R.Patel, Information Security Theory and Practice. First Edition, Prentice-Hall of India Private Limited, 2008.

[7] Keith Harrison, Bill Munro and Tim Spiller, Security through uncertainty. *P* Laboratories, February, 2007.

[8] Schnier B, Applied cryptography: protocols, algorithms and source code in C. New York: John Wiley and sons, 1996.

[9] Wayne Tomasi "Electronic Communications System Fundamentals through Advanced . 5th edition, Pearson Education, 2008.

[10] Rajski J, Tyszer J, "On the diagnostic properties of linear feedback shift registers", ISSN : 0278-0070, IEEE @ 06 August 2002

[11] Raina R, Marionos P, "Signature analysis with modified linear feedback shift registers (M-LFSRs)", Print ISBN: 0-8186-2150-8, IEEE @ 06 August 2002

[12] Simon Haykin , Communication Systems. , 4th Edition , Willey.

[13] Krishnaswamy S, Pillai H K, "On the Number of Linear Feedback Shift Registers With a Special Structure", ISSN : 0018-9448, IEEE @ 27 February 2012

[14] Murali P, Senthilkumar G, "Modified Version of Playfair Cipher Using Linear Feedback Shift Register", Print ISBN: 978-0-7695-3595-1, IEEE @ 19 June 2009

[15] "Linear Feedback Shift Registers", Available at : http://homepage.mac.com/afj/lfsr.html.

[16] "Linear feedback shift register ", Wikipedia [online], Available at : http://en.wikipedia.org/wiki/Linear_feedback_shift_ register.html.

[17] The Art of Electronics, 2ndEdition,Horowitzand Hill, 1989, pp. 665-667

[18] Dan Healy, "Understanding Linear Feedback Shift Registers – The Easy Way", Yikes [online], Available at : http://www.yikes.com/~ptolemy/lfsr_web/index.htm

[19] P. Alfke, "Efficient Shift Registers, LFSR, Counters, and Long Pseudo-Random Sequence Generators,"XAPP 052, July 7,1996 (Version 1.1)

[20] W.W. Peterson and E.J. Weldon, Jr. *Error Correcting Codes*, MIT press, Cambridge, MA 1972.