Using Geffe-Generator To Generate "One-Time-Pad" Keys

Awney M. Kaftan
College Of Computers Sciences And Mathematics , University of Tikrit , Tikrit , Iraq
(Received 17 / 2 / 2009 , Accepted 14 / 2 /2010)

1-Abstract
In this paper, we use the Geffe-generator which is a nonlinear generator after the advantage of the previous treatments of the weak points of this generator to obtain the sequence (stream) of bits (keys) as a keys of " One- Time- Pad" system such system is a strong system among all other cryptography systems and it is use in cipher and decipher systems. The previous advantage which mention above is treat the relationship between the initial values of the linear feedback shift registers and the output of systems .

2-Introduction
There are two objectives of protection data (in computers and communication systems)[3] :
1. Secrecy (privacy): prevent the unauthorized disclosure of data.
2. Authenticity (integrity): to prevent the unauthorized modification of data.
If the key generator is a weak random, then a cryptanalyst will break the system till we use a good algorithm.[5].

The design goal of the stream cipher is to generate pseudo- random keys (bits) efficiently which are indistinguishable from truly random keys[2],[3], consequently, there is some class of encryption algorithm which is called "One-Time – Pad" (O.T.P.) which is satisfy the upper conditions.

3- One-Time-Pad
Perfect cipher system is used for encryption a large set of characters which are non repeated, i.e. the system uses a random string of characters for message encryption. Every new message is encrypted using a new random string of key. If the random key stream is added to the plain text or nonrandom plain text message, the result will be random cipher text message, and if it is used for encryption, there is no computer power that could decrypt it [4], [5]. The only encryption system that is proven fully secure and can not be broken, is "One-Time – Pad" [5]. This system has a long(maximum) period, now we take this example: if the total number of keys is (N), then the time necessary to perform all keys will be:
T =(N/D) years, to reach the solution it is not necessary to analyze all message.
With the probability of 1/2 are analyzed N/2 message,
and consequently with the probability of 1/K, N/K message are analyzed [4].
If the probability of reaching the key is 1/2 and if it is required that message remain secret more than, for example, 109 years, then:
T=N/(2*D) = N/(2*1034) >=109 → 2* 1043 ≤ N or 2143.8 ≤ N
i.e. the key length must be at least (144) bits.
- If the probability of reaching the key is(1/1012) (i.e. the key is reached much faster), then:
T=N/(1012*D)= N/(1012*1034)= N/1046

4- Secrecy Requirements
Any secure system must have the bellow requirements [1] , [3]:
1. the security of system should depend only on the secrecy of keys not on the secrecy of the algorithms enciphering or deciphering.
2. It should be computationally in feasible for cryptanalyst to determine the deciphering transformation systematically from intercepted cipher text even the corresponding plain text is known. And so on for plain text with cipher text.

5- Key Generator
The serial of keys which obtain from the key generator must satisfy the following conditions [1] [2]:-
Maximum period.
Random or pseudo-random .
high linear complexity
good statistical properties.
pass the random tests which is(frequency test, serial test, run test, poker test, and the auto correlation test).
A good running key generator must produce a key system that has a long period , a high linear complexity and good statistical properties in order to make the key system unpredictable.[1].
Now , For example, we take the Geffe- Generator[6] to obtain the stream of keys (in this case the output of this generator is a sequence of bits) but this generator ,as we know, don't satisfy almost the upper conditions and when we treat the weak points of this generator we can use it to generate a good sequence of bits which satisfy all the conditions and over that we can use it as a key generator for one-time-pad system, in spite of this system is high complicity and random or pseudo-random system, but after we put some treatments for all weak points in Geffe- Generator[7] as explain in figure (1) to obtain a new generator (figure (2)) we obtain a sequence of bits (keys):
Seq.
(01010111011000111110011010010001101010111000001010101110101110010011101000010110110011111010 0111010……) and when we take subsequence from the main sequence and we apply the standard random tests we see it is pass all the tests as shown bellow:
Subsequence (01010111011000111110011010010000),
N=31, n0= 15, n1=16,

1.Frequency test:
X1=(n0 – n1)2 /N = 0.0322580 < 3.84 (table value) (**pass**.)

2.Serial test:
n00=6, n01=6, n10=8, n11=8
X2 = (4/30)(36+36+64+64)- 2/31((15)2 +(16)2) +1=0.3680<5.99(**pass**)

3.Poker test:
m=5, k=6
n0=0, n1=1, n2=2, n3=1, n4=2, n5=0, s.t. n0 is the No. of block which is contain one (0), and n1 contain one (1) and so on….

X3= (32/6) (0+1+4+1+4+0)-6 < 0.5(length of seq.) **d.f**= 2.4 **(pass).**

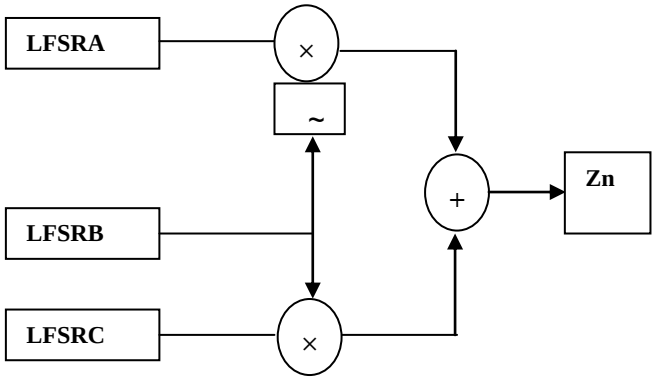


Fig.(1) (Geffe-Generator)[6]

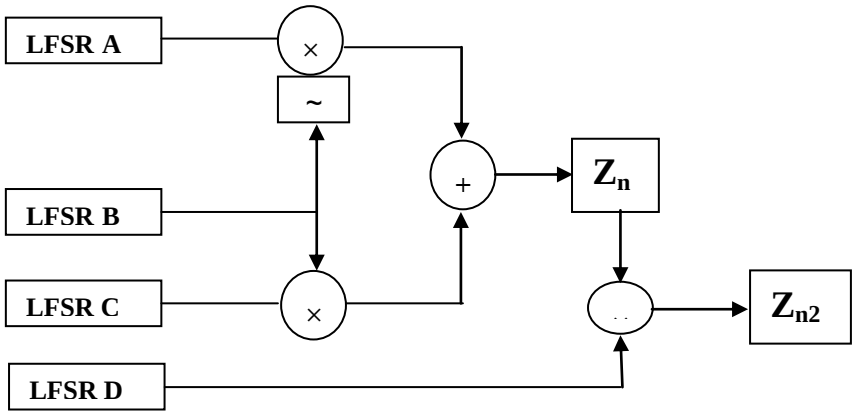
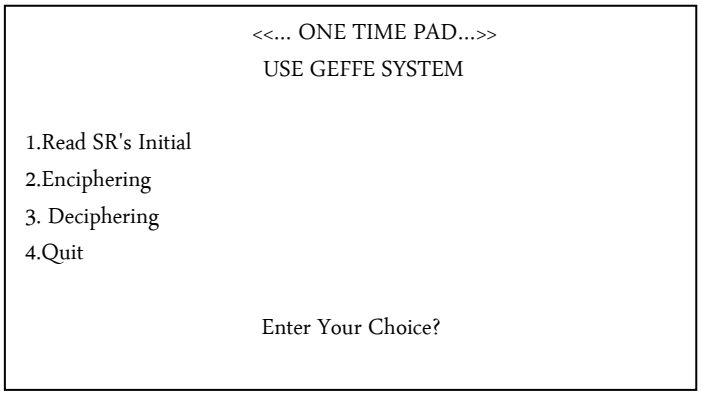


Fig.(2)(Geffe-Generator) after treatment[7]

6- building and implementation :

Now, when we run the general program which is build for this cases (see appendix) to generate sequence of keys (for cipher or decipher) we see it satisfies all the previous conditions and (as shown) it passes all random tests (manually and after tested the sequence of output the generator. The following windows appears as explain bellow:

1.Main Window:



2.Sub-program (procedures):

BIN-TO-DEC : convert binary to decimals.
DEC-TO-BIN : convert decimals to binary.
RED-INITIAL: read the initial values and feedback functions for shift register.
SR-INITIAL: filling shift register by the initial values.
DEC-ENC: implementation the cipher and decipher text.

Appendix

Program One_ Time_ Pad

```
 USES CRT.

TYPE
  Z_O = 0..1;
  ARY8 = ARRAY[1..8] OF Z_O;
  ARY3 = ARRAY[1..3] OF Z_O;

VAR
   OUTPUT_NAME, INPUT_NAME,TITLE1,
            TITLE2,B,INAME                :STRING;
    TIT                                  :ARRAY[1..2,1..2]OF STRING;
     LEN,LENST                           :ARRAY[1..3]OF BYTE;
     STAGE                               : ARRAY[1..3,1..6]OF BYTE;
     SR                                  : ARRAY[1..3,1..127]OF Z_O;
     OPER                                : CHAR;
{……}
PROCEDURE BIN_TO_DEC(BIN :ARY8; VAR  DEC:BYE);
VAR
   I   : INTEGER;
BEGIN
     DEC :=0;
      FOR I:= 1 TO 8   DO
      BEGIN
          DEC:=DEC SHL 1;
           DEC:=DEC  XOR  BIT[I];
       END;
END.

{……}
PROCEDURE  DEC_TO_BIN(DEC :BYTE; VAR  BIN:ARY8);
VAR
   I   : INTEGER;
BEGIN
     DEC :=0;
      FOR I:= 1 TO 8   DO BIT[I]:=0;
      I:=9
      REPEAT
        I:=I-1
         BIT[I]:=DEC  AND 1;
         DEC:=DEC  SHR 1;
        UNTIL DEC:=0;
END.

{……}

PROCEDURE READ_INITIAL;
VAR
   I,J                    :INTEGER;
   INI_FILE               :TEXT;
 BEGIN
   ASSIGN (INI_FILE , INAME); RESET(INI_FILE);
    FOR I:=1  TO   3   DO
    BEGIN
      READ (INI_FILE,LEN[I]);
      READ (INI_FILE,LENST[I]);
      FOR I:= 1 TO  LENST[I]   DO READ (INI_FILE, STAGE[I,J]);
       READLN(INI_FILE);
   END;
   CLOSE(INI_FILE);
 END;
 {……}
```

```
PROCEDURE SR_INITIAL;
VAR
   SR127                  :ARRAY[1..128] OF Z_O;
   I,J,K,M,BT             : INTEGER;
   BIT                    : ARY8;
BEGIN
  FOR I:=1  TO 16   DO
   BEGIN
       DEC_TO_BIN(ORD(BK[I],BIT);
       FOR J:=1 TO 8 DO    SR127[(I-1)*8 + J]:= BIT[J];
    END;
    SR127[I]:=1;
    FOR J:=1 TO 3  DO
    BEGIN
      FOR K:= 1 TO LEN[J] -1 DO
       BEGIN
          BT:=SR127[127] XOR  SR127[13];
          FOR M:= 127 DOWNTO  2 DO SR127[[M]:=SR127[M-I];
          SR127[I]:=BT;
          SR[J,K]:=BT;
         END;
           SR[J,LEN[J]]:=I;
      END;
      END;
{……}

PROCEDURE INFO;
VAR
  FIL           :FILE OF CHAR;
  CH            : CHAR;
  I,J           :INTEGER;
  F             :TEXT;
  REC           :STRING;

BEGIN
    WINDOW (1,1,80,25); TEXTBACKGROUND(BLUE);CLRSCR;
TEXTCOLOR(WHITE);
FOR I:=1 TO 25 DO
   BEGIN
     TEXTCOLOR(BLUE);
  WINDOW(41-I,41-I,19); TEXTBACKGROUND(WHITE);CLRSCR;
   WRITELN; WRITE(TIT[1,1][42-I]);WRITE(TIT[2,1][42-I]);
   WINDOW(41+I,41+I,19); TEXTBACKGROUND(WHITE);CLRSCR;
   WRITELN; WRITE(TIT[1,2][ I]);WRITE(TIT[2,2][I]);
 DELAY(3440);
 END;
WINDOW(18,9,63,18); TEXTCOLOR(BLACK);
ASSIGN(FIL,'INFO.DAT'); RESET(FIL);
WHILE NOT EOF(FIL) DO
 BEGIN
    READ(FIL,CH);
    WRITE(CH);DELAY(100);
  END;
CLOSE(FIL);
END;

{……}

PROCEDURE DEC_ENC;
VAR
FILEI,FILEO       : FILE OF CHAR;
CH,PC             : CHAR;
OUT               : ARY8;
```

```
 I,J,K,M,N            : INTEGER;
 KEY                  : BYTE;
 BT1                  : ARY3;
 BT                   : Z_O;

BEGIN
 ASSIGN(FILEI, INPUT_NAME); RESET(FILEI);
 ASSIGN(FILEO,OUT_NAME); REWRITE(FILEO);
 WINDOW(1,1,80,22); TEXTBACKGROUND(BLUE);
 TEXTCOLOR(WHITE); CLRSCR;
 WHILE NOT EOF(FILEI) DO
  BEGIN
     READ(FILEI,CH);
     FOR M:= 1 TO 8 DO
      BEGIN
        FOR  J:= 1 TO 3 DO
        BEGIN
          BT1[J]:= 0;
          FOR K:= 1 TO LENST[J] DO
            BT1:= BT1[J] XOR SR[J, STAGE[J,K]];
             FOR K:= LEN[J] DOWNTO 2 DO SR[J,K]:=SR[J,K-1];
                SR[J,I]:= BT1[J];
             END;
         OUT[M]:= (BT1 [2]*BT1[1]+ ((BT1[2] XOR 1)* BT1[3]);
       END;
      BIN_TO_DEC (OUT,KEY);
PC:= CHR(ORD(CH) XOR KEY);
WRITE( FILEO,PC); WRITE(PC);
END;
CLOSE(FILI); CLOSE (FILEO);
WINDOW(1,1,80,25);
 TEXTBACKGROUND(BLACK); TEXTCOLOR(30);
GOTOXY(33,24); WRITELN(' PRESS ENTER'); READLN;
END;
{……}

BEGIN
  TITLE1:= '<<… ONE TIME PAD…>>;
   TITEL2:=' USE GEFFE SYSTEM ';
TITLE;
INFO;
REPEAT
WIN;
WINDOW(25,8,60,21);TEXTBACKGROUND(BLACK);CLRSCR;
WINDOW(23,7,58,20);TEXTBACKGROUND(5);CLRSCR;
TEXTCOLOR( YELLOW);
WRITELN; WRITELN;
WRITELN( '        (1) Read Sr's Initial' ); WRITELN;
WRITELN( '        (2) Enciphering' ); WRITELN;
WRITELN( '        (3) Deciphering' ); WRITELN;
WRITELN( '        (4) Quit' ); WRITELN;
  WRITELN( '        Enter Your Choice?' ); WRITELN;
OPER:= READKEY;
 CASE OPER OF
'1'       : BEGIN
         WINDOW(1,1,80,25); TEXTBACKGROUND(BLUE);CLSCR;
         WINDOW(18,11,62,14); TEXTBACKGROUND(0);CLSCR;
         WINDOW(17,10,60,1); TEXTBACKGROUND(5);CLSCR;
       WRITELN; WRITEL(' ENTER INITIAL FILE NAME?');
        READLN(INAME);
        READ_INITIAL;
END;
'2'..'3'  : BEGIN
```

```
WINDOW(1,1,80,25); TEXTBACKGROUND(BLUE);CLRSCR;
 TEXTCOLOR(3);GOTOXY(34,5);
  IF OPER = '2' THEN WRITE( 'Enciphering')
   ELSE WRITE('Deciphering');
WINDOW(17,8,67,17); TEXTBACKGROUND(0);CLSCR;
WINDOW(15,7,65,16); TEXTBACKGROUND(5);CLSCR;
    TEXTCOLOR( WHITE); WRITELN;
    IF OPER ='2' THEN WRITE ('ENTER PLAIN FILE NAME?')
    ELSE WRITE ('ENTER CIPHER FILE NAME?');
    READLN(INPUT_NAME); WRITELN;
    WRITE('ENTER OUTPUT FILE NAME?');
   READLN(OUTPUT_NAME); WRITELN;
WRITELN(': 19,'123456789023456');
WRITE('ENTER BASIC KEY?');READLN(BK);
SR_INITIAL;
DEC_ENC;
END;
END;
UNTIL OPER = '4';
WINDOW(1,1,80,25);TEXTBACKGROUND(0);TEXTCOLOR(7);
CLSCR;
END.
```

References :

1- Prof. Dr. Golec, Jan. 2002 (key generator system) Rotary Institute , Beograd.

2- (Omnisec ,s Solution to the correlation problem in Bruer: threshold –generator and pless-generator) Omnisec company for cryptography Equipments . 1994 .

3- Prof. Brainier Keskenovic and Prof. Miladin Dabic, 2001, (Roots of stream Cipher) , Bill Export-Import, Beograde, Yugoslavia.

4- Hongjun Wu, April ,2004, (A New Stream Cipher HC-256) , Institute of Infocmm Research , Singapore,.

5-Martin B., and, Mette V. and others, 2001, (Rabbit: A New High – performance Stream Cipher) CRYPTICO A/S, Copenhagen, Denmark.

6. Awney M. Kaftan, and, Nazar K. Hussain,(A NEW TREATMENT OF THE ATTACK BY USING CORRELATIONSHIP), مجلة تكريت للعلوم الادارية والاقتصادية، المجلد٤، العدد (١١) لسنة 2008.

٧. عوني محمد كفطان، (استخدام علاقة الارتباط في ايجاد المحتويات الابتدائية والربط لمسجلات الازاحة الخطية) مجلة تكريت للعلوم الادارية والاقتصادية، المجلد١، العدد الاول لسنة ٢٠٠٥).

استخدام مولد GEFFE لتوليد مفتاح المرة الواحدة "ONE-TIME-PAD"

عوني محمد كفطان

كلية علوم الحاسبات والرياضيات ، جامعة تكريت ، تكريت ، العراق

(تاريخ الاستلام: ١٧ / ٢ / ٢٠٠٩ ، تاريخ القبول: ١٤ / ٢ / ٢٠١٠)

الملخص:

استخدمنا في هذا البحث المولد Geffe والذي هو مولد مفاتيح غير خطي كمولد مفاتيح شفرية عشوائية بعد الاستفادة من المعالجات السابقة لمولد Geffe واستخدام هذه المفاتيح كمفاتيح لنظام مفتاح المرة الواحدة (One-Time-Pad) والذي يعتبر من اقوى الانظمة الشفرية والتي تستخدم للتشفير او لحل الشفرة. اذ اننا استفدنا من معالجة علاقة الارتباط بين المدخلات في المسجلات الخطية الزاحفة وبين المخرجات النهائية للمنظومة والتي كانت احدى اسباب ضعف مخرجات المولد Geffe .