Solving Linear Equations Systems Using Genetic Algorithm

Faez Hassan Ali Al_Azawi
Al_Mustansiriyah University

Ahmed Shawki Jaber Al_Asady
Al_Mustansiriyah University

Abstract:

Genetic Algorithms (GA's) are a class of optimization algorithms. GA's attempts to solve problems through modeling a simplified version of genetic process. There are many problems for which a GA approach is useful.

This paper aims to solve Linear Equations System (LES) for any number of variables using the GA. The application of this paper represented by cryptanalysis application, this done by attacking stream cipher systems, choosing one Linear Feedback Shift Register (LFSR), since its considered as a basic unit of stream cipher systems, in the performance of GA. The application divided into two stages, first, constructing LES's for the LFSR, and the second, is attacking the variables of LES's which they are also the initial key values the of LFSR.

1. Introduction

Cryptanalysis is the science and study of methods of breaking ciphers. It is a system identification problem, and the goal of Cryptography is to build systems that are hard to identify [1]. To attack a cryptographic system successfully the cryptanalysis is forced to be based on subtle approaches, such as knowledge of at least part of the text encrypted, knowledge of characteristic features of the language used,..., with some luck. The Cryptosystem are the systems which use the encryption and decryption processes.

The GA is an adaptive search method that has the ability for a smart search to find the best solution and to reduce the number of trials and time required for obtaining the optimal solution. The practicality of using the GA is to solve complex problems compared with traditional search techniques.

In this work, a LES of LFSR (which is considered as a basic unit of stream cipher systems) is constructed then solve it using the genetic algorithm.

2. Modern Cryptosystems

There are essentially two different types of cryptographic systems (cryptosystems), these cryptosystems are: public key and secret key cryptosystems [2]. First let us redefined some important notations:

- **P is the Plaintext message and C is the Ciphertext message.**
- **Key space K: a set of strings (keys) over some alphabet, which includes the encryption key e_k and the decryption key d_k.**
- **The Encryption process (algorithm) E: $Ee_k(P) = C$.**
- **The Decryption process (algorithm) D: $Dd_k(C) = P$.**
- **The algorithms E and D must have the property that: $Dd_k(C)=Dd_k(Ee_k(P)) = P$.**

The public key cryptosystem also called asymmetric cryptosystems. In a public key (non-secret key) cryptosystem, the encryption key e_k and decryption key d_k are different, that is $e_k \neq d_k$. The secret Key Cryptosystem also called symmetric cryptosystems. In a conventional secret-key cryptosystem, the same key ($e_k=d_k=k \hat{I} K$), called secret key, used in both encryption and decryption; we are interest in this type of cryptosystems. The stream cipher systems one of the important types of the secret key cryptosystems [3].

3. Stream Cipher systems

In stream ciphers, the message units are bits, and the key is usual produced by a random bit generator (see fig.1). The plaintext is encrypted on a bit-by-bit basis.

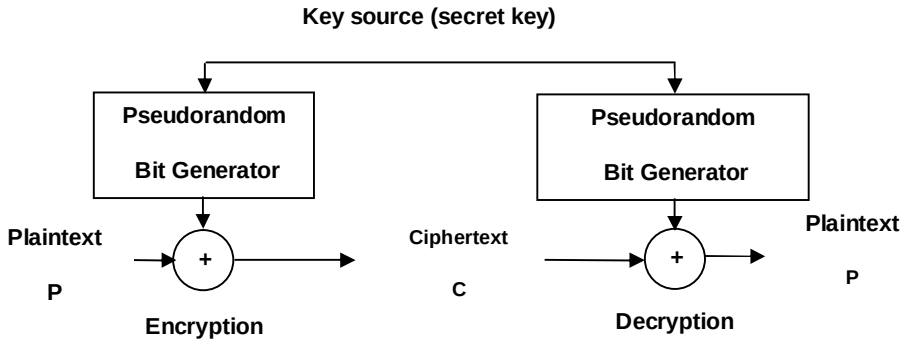

Fig.1 Stream cipher system.

The key is fed into random bit generator to create a long sequence of binary signals. This "key-stream" k is then mixed with plaintext m, usually by a bit wise XOR (Exclusive-OR modulo 2 addition) to produce the ciphertext stream, using the same random bit generator and seed.

Linear Feedback Shift Register (LFSR) systems are used widely in stream cipher systems field. A LFSR System consists of two main basic units. First, is a LFSR function and initial state values. The second one is, the Combining Function (CF), which is a boolean function. Most of all stream cipher systems are depend on these two basic units. Most practical stream-cipher designs center around LFSR. In the early days of electronics, they were very easy to build. A shift register is nothing more than an array of bit memories and the feedback sequence is just a series of XOR gates. A LFSR-based stream cipher can give a lot of security with only a few logic gates [4].

4. Genetic Algorithms (GA's)

Genetic Algorithms (GA's) are search algorithms based on the mechanics of natural selection and natural genetics.

They combine survival of the fittest among string structures with a structured yet randomized information exchange to form a search algorithm with some of the innovative flair of human search [5]. GA is an iterative procedure, which maintains a constant size population of candidate solution. During each iteration step (Generation) the structures in the current population are evaluated, and, on the basic of those evaluations, a new population of candidate solutions formed. The basic GA cycle shown in fig. 2 [6].

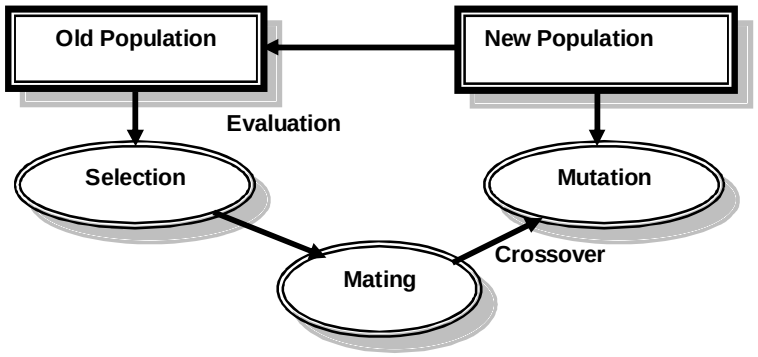

Fig. 2 The basic cycle of GA [6].

An abstract view of the GA is:

Generation=0;

Initialize G(P); {G=Generation ; P=Population}

Evaluate G(P);

While (GA has not converged or terminated)

Generation = Generation + 1;

Select G(P) from G(P-1);

Crossover G(P);

Mutate G(P);

Evaluate G(P);

End (While)

Terminate the GA [6].

5. Constructing a Linear Equations System

Let's assume that the tested LFSR is maximum LFSR (m-LFSR), then its period is $P=2^r-1$, where r is LFSR length. Let SR_r be a single LFSR with length r, let $A_0=(a_{-1},a_{-2},\ldots,a_{-r})$ be the initial value vector of SR_r, s.t. a_{-j}, $1 \pounds j \pounds r$, be the component j of the vector A_0, in another word, a_{-j} is the initial bit of stage j of SR_r, let $C_0^T=(c_1,\ldots,c_r)$ be the feedback vector, $c_j \hat{I} \{0,1\}$, if $c_j=1$ that means the stage j is connected else its not. Let $S=\{s_i\}_{i=0}^{m-1}$ be the sequence (or $S=(s_0,s_1,\ldots,s_{m-1})$ read "S vector") with length m generated from SR_r. The generation of S depending on the following equation:

$$s_i = a_i = \sum_{j=1}^{r} a_{i-j} c_j \quad i=0,1,\ldots \qquad \ldots(1)$$

Equation (1) represents the linear recurrence relation [7].

The objective is finding A_0, when r, C_0 and S are known. Let M be a $r \times r$ matrix, which is describes the initial phase of SR_r:

$M=(C_0|I_{r \times r-1})$, where $M^0=I$.

Let A_1 represents the new initial of SR_r after one shift, s.t.

$$A_1=A_0 \times M=(a_{-1},a_{-2},\ldots,a_{-r}) \begin{pmatrix} c_1 & 1 & \mathbf{L} & 0 \\ c_2 & 0 & \mathbf{L} & 0 \\ M & M & M & M \\ c_r & 0 & \mathbf{L} & 0 \end{pmatrix} = \left(\sum_{j=1}^{r} a_{-j} c_j , a_{-1},\ldots,a_{1-r}\right).$$

In general,

$$A_i=A_{i-1} \times M, \quad i=0,1,2,\ldots \qquad \ldots(2)$$

Equation (2) can be considered as a recurrence relation, so we have:

$$A_i=A_{i-1} \times M=A_{i-2} \times M^2=\ldots=A_0 \times M^i \qquad \ldots(3)$$

The matrix M^i represents the i phase of SR_r, equations (2) and (3) can be considered as a Markov Process s.t., A_0, is the initial probability distribution, A_i represents probability distribution and M be the transition matrix [8].

notice that:

$M^2=[C_1C_0|I_{r\times r-2}]$ and so on until get $M^i=[C_{i-1}\ldots C_0|I_{r\times r-i}]$, where $1\pounds i<r$.

When $C_P=C_0$ then $M^{P+1}=M$.

Now let's calculate C_i s.t.

$$C_i=M\times C_{i-1},\ i=1,2,\ldots \qquad \ldots(4)$$

Equation (1) can be rewritten as:

$$A_0\times C_i=s_i,\ i=0,1,..,r-1 \qquad \ldots(5)$$

if i=0 then $A_0\times C_0=s_0$ is the 1st equation of the LES,

if i=1 then $A_0\times C_1=s_1$ is the 2nd equation of the LES, and

if i=r-1 then $A_0\times C_{r-1}=s_{r-1}$ is the r^{th} equation of the LES.

In general:

$$A_0\times \Psi=S \qquad \ldots(6)$$

Ψ represents the matrix of all C_i vectors s.t.

$$\Psi = (C_0C_1\ldots C_{r-1}) \qquad \ldots(7)$$

The LES can be formulated as:

$$A = [\Psi^T|S^T] \qquad \ldots(8)$$

A represents the extended (augmented) matrix of the LES.

Example (1)

Let the SR_4 has $C_0^T=(0,0,1,1)$ and S=(1,0,0,1), by using equation (4), we get:

$$C_1=M\times C_0=\begin{pmatrix}0&1&0&0\\0&0&1&0\\1&0&0&1\\1&0&0&0\end{pmatrix}\begin{pmatrix}0\\0\\1\\1\end{pmatrix}=\begin{pmatrix}0\\1\\1\\0\end{pmatrix}$$, in the same way, $C_2=\begin{pmatrix}1\\1\\0\\0\end{pmatrix}$, $C_3=\begin{pmatrix}1\\0\\1\\1\end{pmatrix}$

From equation (6) we have:

$A_0\begin{pmatrix}0&0&1&1\\0&1&1&0\\1&1&0&1\\1&0&0&1\end{pmatrix}$=(1,0,0,1), this system can be written as equations:

$$a_{-3}+a_{-4}=1$$

$$a_{-2}+a_{-3}=0$$

$$a_{-1}+a_{-2}=0$$

$$a_{-1}+a_{-3}+a_{-4}=1$$

Then the LES after using formula (8) is:

$$A=\left[\begin{array}{cccc|c}0&0&1&1&1\\0&1&1&0&0\\1&1&0&0&0\\1&0&1&1&1\end{array}\right] \quad \ldots(9)$$

6. Use GA to Break Stream Cipher System

The GA will be used to solve LES's of LFSR with length r, m=r equations are needed to solve the LES.

6.1 Coding Scheme

A LES is decoded by binary representation. As an example, the equation $a_2+a_5=1$ of LFSR with length 5 decoded by the equation string (01001-1), where the absolute value (right side) of the equation is the real key of the LFSR. These equations are constant for fixed LFSR's length and connection function. As this representation indicates, the size of the equations space is 2^m-1 (ignoring the zero string). By increasing the number of bits that are used for representing one continuous variable the accuracy of the representation can be increased [9].

6.2 Initial Population

For the initialization process we can initialize the population by a random sample of combinations of 0 and 1 with m-string length represents the probable initial values LFSR's. The creation of the population must submit to what we called non-zero initial condition. The *Initial Population Algorithm* shown in fig.3.

Initial Population Algorithm

INPUT : READ NumGenes; {*Number of genes*}

READ m; {*Length of string*}

PROCESS : FOR i := 1 TO NumGenes DO

BEGIN

FOR j = 1 TO m DO

s_i := RANDOM(2);

Fig. 3 Initial Population Algorithm

6.3 Evaluation Function (Fitness Function)

This function is used to determine the "best" representation. The process of the evaluation function selection is as follows:

1. **From population, a chromosome k, k=1,…,m initial string of length m bits, so we get the string $X_k=(X_{k1},X_{k2},\ldots,X_{km})$.**
2. **The string bit X_{kj} product with corresponding equation string bit Y_{ij}, where $1\le j\le m$ s.t. the equation string is $Y_i=(Y_{i1},Y_{i2},\ldots,Y_{im})$ and calculate the observed value:**

$$O_{ki}=X_{k1}*Y_{i1}\text{Å}X_{k2}*Y_{i2}\text{Å}\ldots\text{Å}X_{km}*Y_{im}=\sum_{j=1}^{m} X_{kj} * Y_{ij} \qquad \ldots(10)$$

3. **Compare the observed value O_{ki} with key value K_i which represents the known output value of the cryptosystem, by using mean absolute error (MAE) s.t.**

$$MAE_k = \frac{1}{m}\sum_{i=1}^{m}\left|O_{ki} - K_i\right| \quad \text{...(11)}$$

4. The Fitness value is

$$Fitness_k = 1\text{-}MAE_k = 1\text{-}\frac{1}{m}\sum_{i=1}^{m}\left|O_{ki} - K_i\right| \quad \text{...(12)}$$

where

m : The length of the chromosome string or equation string.

X_{kj}: is the initial value j in String X_k.

Y_{ij}: is the equation variable j in the string Y_i.

O_{ki}: is the observed value i of string X_k calculated from equation (10).

K_i: is the key bit (actual value) i.

When the measured (observed) value O_{ki} matches the key bit K_i, for all $1 \le i \le m$, then the summation terms MAE_k in equation (11) evaluate to 0 so the fitness value is 1. The fact that a fitness value of 0 is never achieved does not affect the algorithm since high fitness values are more important than low fitness values. As a result, the search process is always moving towards fitness values closer to or equal 1. The steps of the Fitness Algorithm are shown in fig. 4:

Fitness Algorithm

INPUT : 1. READ X vector; {*Initial string with length m from population*}

2. READ Y vector; {*Equation string from data base file*}

3. READ K vector ;{*Actual key=absolute value of LES*}

PROCESS : 4. FOR i = 1 TO m DO

4.1 $O_i := \sum_{j=1}^{m} X_j * Y_j$; {*XOR sum, O_i is observed key*}

4.2 $Dif_i := |O_i - K_i|$;

Fig. 4 Fitness algorithm

6.4 Genetic Operators

I. Selection Operator

This method uses the roulette wheel selection method. The string with high fitness has a higher probability of contributing one or more offspring to the next generation.

II. Crossover Operator

Given a population of initial string, each one with a fitness value, the algorithm progresses by a high fitness value selecting two initial strings for mating. The two parents generate two children using crossover operation. The breeding process is achieved using the single (one)-point crossover select randomly one point in the range [1,2,...,n], where n is the length of the string. The first offspring is produced by deleting the crossover fragment of the first parent from the first and inserting the crossover fragment of the second parent at the crossover point of the first point. The second offspring is produced in a symmetric manner [10]. The *Crossover Algorithm* shown in Fig. 5.

```
Crossover Algorithm
INPUT   : 1. READ Parent1, Parent2;
PROCESS         : 2. REPEAT
                        2.1 x := RANDOM(m);
                    UNTIL x > 1;
                  3. Child1 := copy(parent1,1,x)+copy(parent2,x+1,m);
```

Fig.5 Crossover algorithm

III. Mutation Operator

After the new generation has been determined, the initial strings are subjected to a low rate mutation process. For this study when a bit in the initial string is selected for mutation randomly, it is swapped with the bit in the right (or left). If the fitness of the string is in the top half of the population, the character is

swapped with the character to its right. Of course, the swapping process subjects to non-zero initial condition.

Fig.6 shows the Mutation Algorithm.

Mutation Algorithm

```
INPUT   : 1. READ STR;
PROCESS       : 2. New_STR := STR;
                3. REPEAT
                        3.1 x := RANDOM(m);
                   UNTIL x > 0;
```

Fig. 6 The Mutation Process.

6.5 Genetic Parameters

The following parameters are been used: population size =10, probability of crossover P_c=0.7, probability of mutation P_m=0.05, and 100 or 200 generations.

7. Experimental Results

Two stopping criterions are be used to stop the GA cryptanalysis system, first criterion, two hundred generations are enough to reach this level of fitness. The second, when the fitness value reaches (1.0), so no need to reach the high number of generation. The algorithm was fast enough that this took less than a minute on P4 PC. Let's use 11 stages-LFSR, which has $1+x^2+x^{11}$ as characteristic primitive polynomial. The initial key value is: 10000000001. Table (1) shows the 11-stage equations of LES for single LFSR with binary representation.

Table (1) The 11-Stage Equations of LES for Single LFSR.

I	Equation	Binary representation
1	$a_2+a_{11}=1$	01000000001 1
2	$a_1+a_{10}=1$	10000000010 1
3	$a_2+a_9+a_{11}=1$	01000000101 1
4	$a_1+a_8+a_{10}=1$	10000001010 1
5	$a_2+a_7+a_9+a_{11}=1$	01000010101 1
6	$a_1+a_6+a_8+a_{10}=1$	10000101010 1
7	$a_2+a_5+a_7+a_9+a_{11}=1$	01001010101 1
8	$a_1+a_4+a_6+a_8+a_{10}=1$	10010101010 1
9	$a_2+a_3+a_5+a_7+a_9+a_{11}=1$	01101010101 1
10	$a_1+a_2+a_4+a_6+a_8+a_{10}=1$	11010101010 1
11	$a_1+a_2+a_3+a_5+a_7+a_9+a_{11}=0$	11101010101 0

For this example, only 10 initial keys were in the gene pool. The system began by generating 10 random initial key as shown:

Key 1: 00111000011 à Fitness: 0.64

Key 2: 10001011111 à Fitness: 0.55

Key 3: 11111101111 à Fitness: 0.55

Key 4: 10100111101 à Fitness: 0.64

Key 5: 00000001110 à Fitness: 0.45

Key 6: 00111110101 à Fitness: 0.36

Key 7: 01110110110 à Fitness: 0.55

Key 8: 11110100110 à Fitness: 0.45

Key 9: 11000111110 à Fitness: 0.73

Key 10: 11011101110 à Fitness: 0.55

The average fitness is 0.55. The best of these keys (key9) has a fitness value: 0.73.

After 31 generation, the pool begins to converge at a high rate of speed:

Key 1: 10001010001 à Fitness: 0.82

Key 2: 10010100001 à Fitness: 0.91

Key 3: 10010000001 à Fitness: 0.73

Key 4: 10010000001 à Fitness: 0.82

Key 5: 00101110001 à Fitness: 0.64

Key 6: 10100100010 à Fitness: 0.45

Key 7: 10100100011 à Fitness: 0.64

Key 8: 10100100001 à Fitness: 0.55

Key 9: 10110100001 à Fitness: 0.73

Key 10: 10010100101 à Fitness: 0.45

Average fitness has risen to 0.70 with the best key (key2) coming in at 0.91.

By generation 50, the gene pool looks like:

Key 1: 10000000001 à Fitness: 1.00

Key 2: 10101000001 à Fitness: 0.82

Key 3: 10010100001 à Fitness: 0.91

Key 4: 10100000001 à Fitness: 0.82

Key 5: 10101100001 à Fitness: 0.64

Key 6: 10011010001 à Fitness: 0.73

Key 7: 10000010010 à Fitness: 0.36

Key 8: 10100010001 à Fitness: 0.82

Key 9: 10000100001 à Fitness: 0.73

Key 10: 10111001001 à Fitness: 0.73

The average fitness is at 0.75. key1 turns out to have the highest fitness and on examination is the exact key. In table (2) we provide the generation number for which noted improvement in the evaluation function, together with the value of the function.

Table (2) Results of 50 Generations for Single LFSR.

Gen.	Fitness	Average	key	Best Initial Key
0	0.73	0.55	9	11000111110
17	0.82	0.60	2	10010001001
31	0.91	0.70	2	10010100001
50	1.00	0.75	1	10000000001

The best initial key after (11) generations was (and equal to the real initial key):

1 0 0 0 0 0 0 0 0 0 1

8. Design of GA Cryptanalysis System

The GA cryptanalysis system can be view as two main parts, first, is the LES constructing which described LES constructing algorithm which is shown in Fig.7.

LES Constructing Algorithm

INPUT : 1. **READ** A_0; { *initial value of LFSR* }

2**. READ** C_0^T vector;

3 M := $(C_0|I_{rxr-1})$;

4 **READ** A_0 vector; { *Initial values of LFSR*}

PROCESS : 5. **FOR** i := 0 **TO** m-1 **DO**

5.1 $C_i := M \times C_{i-1}$;

5.2 $s_i := A_0 \times C_i$;

Fig. 7 LES Constructing Algorithm

The second part is the GA cryptanalysis part, which illustrated in GA cryptanalysis algorithm, shown in fig. 8:

GA Cryptanalysis Algorithm

INPUT : 1. READ Equation Information File of LFSR;

2. READ Actual Key of LFSR;

3. READ G; {*Number of Generations* }

INITILIZE : 4. CALL Initial Population;

5. CALL Evaluation Function; { *Fitness* }

PROCESS : 6. i := 1; Best.Fit := 0;

7. REPEAT

7.1 Selection;

7.2 CALL Crossover;

Fig. 8 GA Cryptanalysis Algorithm

9. Conclusions

This research concludes the following aspects:

1. **Although the proposed system is employed for small shift register length (r≤11), it was provide the base of building GA cryptanalysis system valid for long shift register attacking.**

2. **As be recommended, the mutation probabilities must be as low as possible and probability of crossover must be high in order to approach the correct or heuristic key of the LES obtained from the mentioned LFSR.**
3. **As a logical mathematical situation, if the proposed system gives a fitness value less than 1.0, this mean, no results obtained so we must run the system a gain, since the LES must has unique solution for fixed absolute values, no another solution gives fitness equal 1.0.**
4. **Percentages reported are based on number of tests and different numbers of the tests must be always used, and that what will done in this research.**

References

[1] . P. Ekdhal, "*On LFSR based Stream Ciphers Analysis and Design*", Ph.D. Thesis, Dept. of Economics, West Virginia University, Nov., 2003.

[2] . Yan, S. Y., "*Number Theory for Computing*", Springer-Verlag Berlin, 2000.

[3] . Schneier B., "*Applied Cryptography*", John Wiley & Sons, 1997.

[4] . Juntao G., Xuelian L. and Yupu H., "*Fault Attack on the Balanced Shrinking Generator*", Wuhan University Journal of Natural Science Vol.11 No.6 P.1773-1776, 2006.

[5] . Poli, R., Langdon, W. B. and McPhee, N. F., "*A Field Guide to Genetic Programming*", Lulu.com, freely available from the internet. ISBN 978-1-4092-0073-4, 2008.

[6] . Mitchell M., "*An Introduction of Genetic Algorithms*", Abroad Book 1998.

[7] . Golomb, S.W., "*Shift Register Sequences*" San Francisco: Holden Day 1967, (Reprinted by Aegean Park Press in 1982).

[8] . Papoulis, A. "*Probability Random Variables, and Stochastic Process*", McGraw-Hill College, October, 2001.

[9] . Rothlauf, F, "*Representations for Genetic and Evolutionary Algorithms*", 2nd Edition, Springer-Verlag Berlin Heidelberg, 2006.

[10] . M. Sabah Salmo, "*A Comparative Study between Traditional Genetic Algorithms and Breeder Genetic Algorithms*", M.Sc., Thesis, AL-Nahrain University, 2004.

حل نظم المعادلات الخطية باستخدام الخوارزمية الجينية

م.م. فائز حسن علي
الجامعة المستنصرية

م.م.احمد شوقي جابر
الجامعة المستنصرية

المستخلص :

الخوارزميات الجينية (Genetic Algorithms) تمثل مجموعة الخوارزميات الامثلية. الخوارزمية الجينية تحاول حل المسائل من خلال بناء نموذج جيل بسيط من العملية الجينية. لقد نجحت الخوارزمية الجينية في حل الكثير من المسائل. وبالطبع فان هذه الخوارزمية ستكون غير تقليدية في حالة كون تحليل الشفرة هو احد هذه المسائل.

هذا البحث يهدف إلى حل نظم المعادلات الخطية لأي عدد من المتغيرات باستخدام الخوارزمية الجينية. إن مجال تطبيق هذا البحث هو تحليل الشفرة (Cryptanalysis)، وهذا يتم من خلال مهاجمة نظم التشفير الانسيابي (Stream Cipher Systems)، باختيار مسجل زاحف خطي ذو تغذية مرتدة (Linear Feedback Shift Register)، باعتباره الوحدة الأساسية التي تدخل في بناء نظم التشفير الانسيابي، معتمدين على انجاز الخوارزمية الجينية. التطبيق في هذا البحث يتم في مرحلتين، الأولى تتمثل ببناء نظام معادلات خطية من مخرجات المسجل الزاحف، والمرحلة الثانية هي حل نظام المعادلات الخطية ومعرفة قيم المجاهيل والتي تمثل قيم المفتاح الابتدائي للمسجل الزاحف.