Cryptanalysis of Stream Cipher System Using Particle Swarm Optimization Algorithm

Hussein Ali Mohammed Al_Sharifi
M.Sc. in Mathematics /College of Education/Karbala University
Email: hussein7712a@yahoo.com

مستخلص

أن تحقيق أمثلية السرب الجزيئي Particle Swarm Optimization (PSO) تعني الآشارة الى عائلة جديدة من الخوارزميات التي تستخدم لايجاد حلول مثالية (أو أقرب الى المثالية) للمسائل العددية والكمية.

يعتبر التشفير الانسيابي Stream Cipher عائلة مهمة من خوارزميات التشفير. لقد استخدمت المتتابعات المولدة من المسجل الزاحف Shift Register Sequences في نظم التشفير ونظرية المعلومات، حيث توجد نظريات كثيرة بصددهم. ان لنظم التشفير الانسيابي التي تعتمد على المسجل الزاحف لها الدور الكبير في نظم التشفير وخصوصا العسكرية منها.

هدف هذا البحث تنفيذ خوارزمية تحليل الشفرة (Cryptanalysis) على انظمة التشفير الانسيابي باختيار مسجل زاحف خطي ذو تغذية مرتدة، باعتباره الوحدة الأساسية التي تدخل في بناء نظم التشفير الانسيابي، معتمدين على انجاز خوارزمية أمثلية السرب الجزيئي من خلال حل نظم المعادلات الخطية لأي عدد من المتغيرات التي تمثل مخرجات المسجل الزاحف.

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.

Abstract

The meaning of the Particle Swarm Optimization (PSO) refers to a relatively new family of algorithms that may be used to find optimal (or near optimal) solutions to numerical and qualitative problems.

Stream ciphers are an important class of encryption algorithms. Shift register sequences are used in both cryptography and coding theory. There is a wealth of theory about them; stream ciphers based on shift registers have been the workhorse of military cryptography since the beginnings of electronics.

This paper aims to implement cryptanalysis attack algorithms on stream cipher systems using plaintext attack (or part from it), choosing one Linear Feedback Shift Register (LFSR), since its considered as a basic unit of stream cipher systems, in the performance of PSO by solving Linear Equations System (LES) for any number of variables of the output of LFSR.

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.

Keywords: Particle Swarm Optimization, Cryptography, Stream Cipher Systems, Linear Feedback Shift Register, Linear Equations System.

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.

One of the important new learning methods is a **Particle Swarm Optimization** (PSO), which is simple in concept, has few parameters to adjust and easy to implement. PSO has found applications

in a lot of areas. In general, all the application areas that the other evolutionary techniques are good at are good application areas for PSO [2].

In 1995, Kennedy J. and Eberhart R. [3], introduced a concept for the optimization of nonlinear functions using particle swarm methodology. The evolution of several paradigms outlined, and an implementation of one of the paradigms had been discussed.

In 1999, Eberhart R.C. and Hu X. [4], arranged a new method for the analysis of human tremor using PSO which is used to evolve a NN that distinguishes between normal subject and those with tremor.

In 2004, Shi Y. [2], surveyed the research and development of PSO in five categories: algorithms, topology, parameters, hybrid PSO algorithms, and applications. There are certainly other research works on PSO which are not included due to the space limitation.

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 [5]. 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 ($e_k=d_k=k \in K$), 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 [6].

3. Stream Cipher systems

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

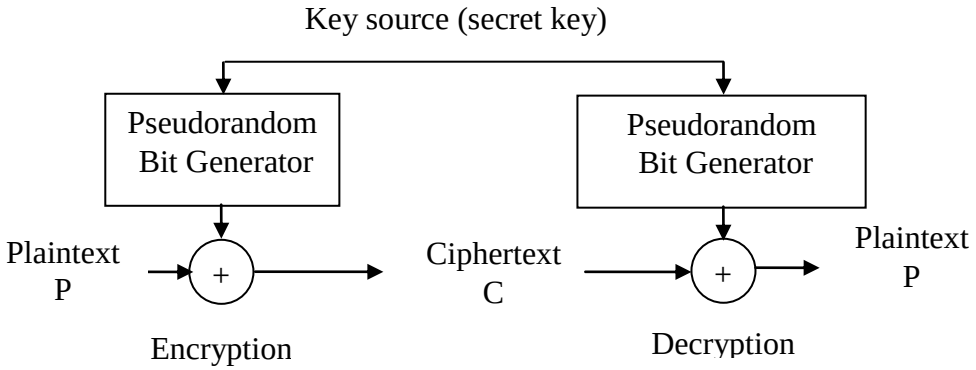

Figure (1) Stream cipher system.

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 [7].

4. Particle Swarm Optimization (PSO)

PSO was originally developed by a social-psychologist J. Kennedy and an electrical engineer R. Eberhart in 1995 and emerged from earlier experiments with algorithms that modeled the "flocking behavior" seen in many species of birds. Where birds are attracted to a roosting area in simulations they would begin by flying around with no particular destination and in spontaneously formed flocks until one of the birds flew over the roosting area [8]. PSO has been an increasingly hot topic in the area of computational intelligence. It is yet another optimization algorithm that falls under the soft computing umbrella that covers genetic and evolutionary computing algorithms as well [9].

4.1 Fitness Criterion

One of these stopping criterions is the fitness value. Since the PSO algorithm is chosen to be a supervised learning algorithm, then there are observed values of (t_i) and desired output values of (f_i). These two values have to be compared, if they are closed to each other then the fitness is good, else the algorithm must continue its calculations until this condition is satisfied or the specified number of iterations is finished.

The corrections are selected to minimize the residual error between t_i and f_i output. The Mean Squared Error (MSE) is one solution for the comparison process:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (t_i - f_i)^2 \quad \ldots (1)$$

Where n is the number of the compared categories.

4.2 PSO Algorithm

The PSO algorithm depends in its implementation in the following two relations:

$$v_{id} = w * v_{id} + c_1 * r_1 * (p_{id} - x_{id}) + c_2 * r_2 * (p_{gd} - x_{id}) \quad \ldots (2a)$$

$$x_{id} = x_{id} + v_{id} \quad \ldots (2b)$$

where c_1 and c_2 are positive constants, r_1 and r_2 are random function in the range [0,1], $x_i=(x_{i1},x_{i2},\ldots,x_{id})$ represents the i^{th} particle; $p_i=(p_{i1},p_{i2},\ldots,p_{id})$ represents the best previous position (the position giving the best fitness value) of the i^{th} particle; the symbol g represents the index of the best particle among all the particles in the population, $v=(v_{i1},v_{i2},\ldots,v_{id})$ represents the rate of the position change (velocity) for particle i [2].

The original procedure for implementing PSO is as follows:

1. Initialize a population of particles with random positions and velocities on d-dimensions in the problem space.
2. PSO operation includes:
   a. For each particle, evaluate the desired optimization fitness function in d variables.
   b. Compare particle's fitness evaluation with its pbest. If current value is better than pbest, then set pbest equal to the current value, and p_i equals to the current location x_i.
   c. Identify the particle in the neighborhood with the best success so far, and assign it index to the variable g.
   d. Change the velocity and position of the particle according to equation (2a) and (2b).
3. Loop to step (2) until a criterion is met.

Like the other evolutionary algorithms, a PSO algorithm is a population based on search algorithm with random initialization, and there is an interaction among population members. Unlike the other evolutionary algorithms, in PSO, each particle flies through the solution space, and has the ability to remember its previous best position, survives from generation to another. The flow chart of PSO algorithm is shown in figure (2) [10].

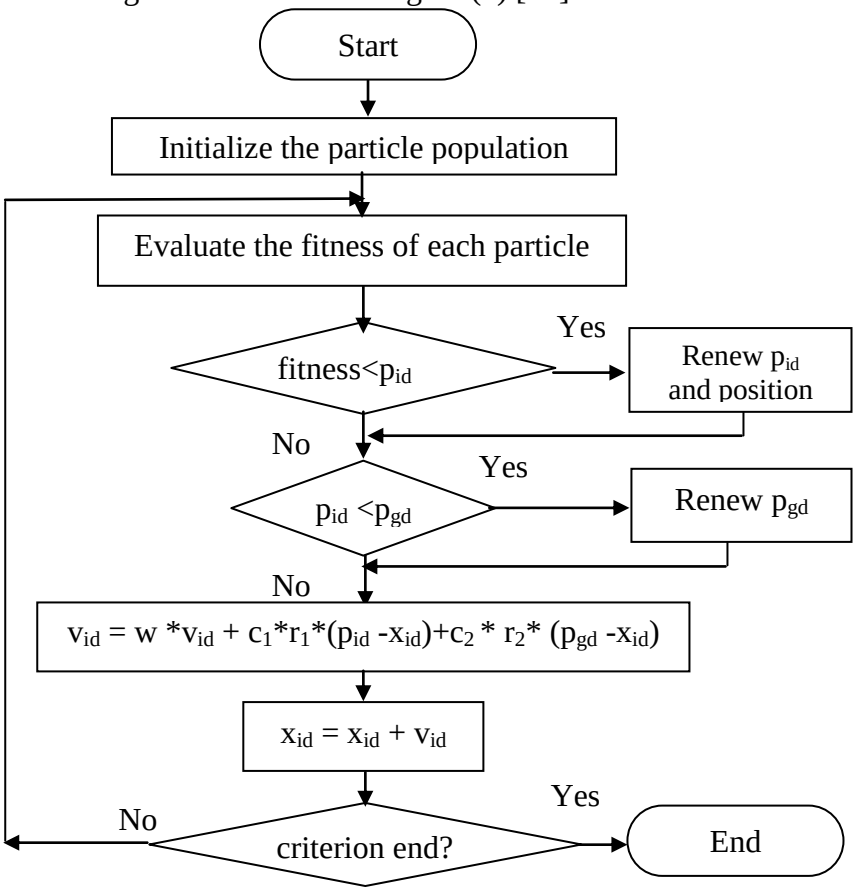

Figure (2) Flowchart of PSO Algorithm [10].

4.3 The Parameters of PSO [11],[12]

A number of factors will affect the performance of the PSO. These factors are called PSO parameters, these parameters are:

1. Number of particles in the swarm affects the run-time significantly, thus a balance between variety (more particles) and speed (less particles) must be sought.
2. Maximum velocity (v_{max}) parameter. This parameter limits the maximum jump that a particle can make in one step.
3. The role of the inertia weight w, in equation (2a), is considered critical for the PSO's convergence behavior. The inertia weight is employed to control the impact of the previous history of velocities on the current one.
4. The parameters c_1 and c_2, in equation (2a), are not critical for PSO's convergence. However, proper fine-tuning may result in faster convergence and alleviation of local minima, c_1 than a social parameter c_2 but with $c_1 + c_2 = 4$.
5. The parameters r_1 and r_2 are used to maintain the diversity of the population, and they are uniformly distributed in the range [0,1].

5. Particle Swarm Optimization Implementation

PSO is an extremely simple concept, and can be implemented without complex data structure. No complex or costly mathematical functions are used, and it doesn't require a great amount of

memory [13]. The facts of PSO has fast convergence, only a small number of control parameters, very simple computations, good performance, and the lack of derivative computations made it an attractive option for solving the problems.

In this paper, the Binary PSO (BPSO) was implemented in optimization applications. From the optimization application the solving linear equations system problem is chosen.

6. 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}, $1\le j\le r$, 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, $c_j\in\{0,1\}$, 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:

$$s_i=a_i=\sum_{j=1}^{r}a_{i-j}c_j \quad i=0,1,\ldots \qquad \ldots(3)$$

Equation (3) represents the linear recurrence relation [14].

The objective is finding A_0, when r, C_0 and S are known. Let M be a r×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.

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

In general,

$$A_i=A_{i-1}\times M,\ i=0,1,2,\ldots \qquad \ldots(4)$$

Equation (4) can be considered as a recurrence relation, so we have:

$$A_i=A_{i-1}\times M=A_{i-2}\times M^2=\ldots=A_0\times M^i \qquad \ldots(5)$$

The matrix M^i represents the i phase of SR_r, equations (4) and (5) 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 [15].

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 $1\le i<r$.

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:

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

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

$$\Psi = (C_0 C_1 \ldots C_{r-1}) \quad \ldots(7)$$

The LES can be formulated as:

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

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:

$$\begin{aligned} a_{-3} + a_{-4} &= 1 \\ a_{-2} + a_{-3} &= 0 \\ a_{-1} + a_{-2} &= 0 \\ a_{-1} + a_{-3} + a_{-4} &= 1 \end{aligned}$$

Then the LES after using formula (8) is:

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

7. Use PSO to Solve LES

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

7.1 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.

7.2 Initial Swarm

For the initialization process we can initialize the swarm 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 swarm must submit to what we called non-zero initial condition. The ***Initial Swarm Algorithm*** shown in figure (3).

Initial Swarm Algorithm
INPUT : 1.**READ** NumParts; {*Number of Particles*}
2.**READ** m; {size *of Particle*}
PROCESS : 3.**FOR** i := 1 **TO** NumParts **DO**
BEGIN
a. **FOR** j = 1 **TO** m **DO** s_j := **RANDOM**(2);
b. $Part_i := (s_1,s_2,\ldots,s_m)$;
END;
OUTPUT : 4.Swarm.
END.

Figure (3) Initial Population Algorithm

7.3 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 swarm, a particle 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:

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

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.

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

4. The Fitness value is

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

where
m : The size of the particle 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 figure (4):

Fitness Algorithm

INPUT : 1. **READ** X vector; {*Initial string with size m from swarm*}
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 $Dif_i := |O_i - K_i|$;

5. $MAE := \frac{1}{m} \sum_{i=1}^{m} Dif_i$;{ *MAE is the Mean Absolute Error*}

6. Fitness := 1-MAE;

OUTPUT : 7. Fitness value;

END.

Figure (4) Fitness algorithm

7.4 PSO Parameters

The following parameters are been used: swarm size =10, $c_1 \in [0.5,2]$, $c_2=c_1$, $v_{max}=2$, $v_{min}=-v_{max}$, $w \in [0.4,0.9]$, $r_1, r_2 \in [0,1]$ and 100 or 200 generations.

8. Experimental Results

Two stopping criterions are be used to stop the PSO 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→Fitness: 0.64
Key 2: 10001011111→Fitness: 0.55
Key 3: 11111101111→Fitness: 0.55
Key 4: 10100111101→Fitness: 0.64
Key 5: 00000001110→Fitness: 0.45
Key 6: 00111110101→Fitness: 0.36
Key 7: 01110110110→Fitness: 0.55
Key 8: 11110100110→Fitness: 0.45
Key 9: 11000111110→**Fitness: 0.73**
Key 10: 11011101110→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→Fitness: 0.82
Key 2: 10010100001→**Fitness: 0.91**
Key 3: 10010000001→Fitness: 0.73
Key 4: 10010000001→Fitness: 0.82
Key 5: 00101110001→Fitness: 0.64
Key 6: 10100100010→Fitness: 0.45
Key 7: 10100100011→Fitness: 0.64
Key 8: 10100100001→Fitness: 0.55
Key 9: 10110100001→Fitness: 0.73
Key 10: 10010100101→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→**Fitness: 1.00**
Key 2: 10101000001→Fitness: 0.82
Key 3: 10010100001→Fitness: 0.91
Key 4: 10100000001→Fitness: 0.82
Key 5: 10101100001→Fitness: 0.64
Key 6: 10011010001→Fitness: 0.73
Key 7: 10000010010→Fitness: 0.36
Key 8: 10100010001→Fitness: 0.82
Key 9: 10000100001→Fitness: 0.73
Key 10: 10111001001→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

9. Design of PSO Cryptanalysis System

The PSO cryptanalysis system can be view as two main parts, first, is the LES constructing which described LES constructing algorithm which is shown in Figure(5)

LES Constructing Algorithm
INPUT: 1. **READ** C_0^T vector, A_0; { *initial value of LFSR* }
2. M := $(C_0|I_{r\times r-1})$;
3. **READ** A_0 vector; { *Initial values of LFSR*}
PROCESS : 4. **FOR** i := 0 **TO** m-1 **DO**
4.1 $C_i := M\times C_{i-1}$;
4.2 $s_i := A_0\times C_i$;
5. S := $(s_0, s_1,\ldots,s_{m-1})$;
6. **Ψ**:= $(C_0, C_1,\ldots, C_{m-1})$;
7. A := $[\mathbf{\Psi}^T,\mathbf{S}^T]$;
OUTPUT : 8. Augmented matrix A;{*store in a file*}
END.

Figure (5) LES Constructing Algorithm

The second part is the PSO cryptanalysis part, which illustrated in PSO cryptanalysis algorithm, shown in figure (6):

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 Swarm;

PROCESS : 5. i := 1;
6**. REPEAT**
6.1 **CALL** Evaluation Function; { *Fitness* }
6.2 **IF** Fitness<p_{id} **THEN**
6.2.1 Renew p_{id} and position;
6.3 **IF** p_{id} <p_{gd} **THEN**
6.3.1 Renew p_{gd}**.**
6.4 $v_{id} = w * v_{id} + c_1 * r_1 * (p_{id} - x_{id}) + c_2 * r_2 * (p_{gd} - x_{id})$
6.5 $x_{id} = x_{id} + v_{id}$
6.6 Increment(i);
UNTIL (i > G) or (Best.Fit = 1.0);

OUTPUT : 7. Best Particle which has highest Best.Fit;

END.

Figure (6) PSO Cryptanalysis Algorithm

10. Conclusions

This research concludes the following aspects:

1. Although the proposed system is employed for small shift register length ($r \leq 11$), it was provide the base of building PSO cryptanalysis system valid for long shift register attacking.
2. 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.
3. 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] . Chen C.,"Classification of Under Water Signals Using Wavelet Transforms and Neural Networks", Math. & computer Modeling, Vol.22, No2, pp. 47-60, 1998.

[2] . Clow B. "A Comparison of Neural Network Training Methods for Character Recognition", Department of Computer Science Carleton University, 2003.

[3] . Eberhart R. C. and Hu X. "Human Tremor Analysis Using Particle Swarm Optimization" Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1999), Washington D.C. pp. 1927-1930, 1999.

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

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

[6] . 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.

[7] . Kennedy J. and Eberhart R. C. "Particle Swarm Optimization", Proceedings of IEEE International Conference on NN, Piscataway, pp. 1942-1948, 1995.
[8] . Papoulis, A. "Probability Random Variables, and Stochastic Process", McGraw-Hill College, October, 2001.
[9] . Parsopoulos K. E. and Vrahatis M.N., "Recent Approaches to Global Optimization Problems through Particle Swarm Optimization", Kluwer Academic Publishers, Netherlands, Natural Computing 1, pp 235–306, 2002.
[10] . Pomeroy P. "An Introduction to Particle Swarm Optimization", Article, March, www.adaptiveview.com, page 1-7, 2003.
[11] . Ribeiro P. F. and Kyle W. S., "A Hybrid Particle Swarm and Neural Network Approach for Reactive Power Control", Member, 2003. http://engr.calvin.edu/WEBPAGE/courses/Reactivepower-PSO-wks.pdf
[12] . Schneier B., "Applied Cryptography", John Wiley & Sons, 1997.
[13] . Settles M. and Rylander B., "Neural Network Learning using Particle Swarm Optimizers", Advances in Information Science and Soft Computing, pp. 224-226, 2002.
[14] . Shi Y., "Particle Swarm Optimization", Electronic Data Systems, Inc. Kokomo, IN 46902, USA Feature Article, IEEE Neural Networks Society, February 2004.
[15] . Yan, S. Y., "Number Theory for Computing", Springer-Verlag Berlin, 2000.