A New Keystream Generator Based on Swarm Intelligence

Ismail K. Ali / ismailkhlil747@yahoo.com
Abdulelah I. Jarullah /Abdul567@yahoo.com

Receiving Date: 2011/7/24 - **Accept Date:** 2011/9/13

Abstract

Advances in the design of keystream generator using swarm intelligence techniques are reported.In this paper particle swarm optimization algorithm for generating random keystream with large complexity is presented. Particle swarm technique is adapted to locate the requirements(larg period ,larg linear complexity, good randomness and high order of correlation immunity). The definitions for some cryptographic properties are generalized, providing a measure suitable for use fitness function in a particle swarm algorithm, seeking randomness that satisfy both correlation immunity and the large linear complexity. Results are presented demonstrating the effectiveness of the proposed method.

Keywords: Particle Swarm Optimization, Stream Cipher, Keystream Generator, Randomness, and Linear Complexity.

الخلاصة

يهدف البحث إلى تقديم طريقة تصميم متقدمة في أنظمة حماية البيانات لمولد مفاتيح انسيابي باستخدام تقنية الحشود الذكية حيث تم استخدام خوارزمية أمثلية حشد الجريئة لتوليد متتابعات مفاتيح انسيابية تمتلك عشوائية جيدة وتعقيد كبير. تم تكيف دالة تقيم مناسبة لخوارزمية حشد الجريئة لتحقيق المتطلبات العامة بمتتابعات المفتاح الانسيابي من حيث (عشوائية جيدة، دورة متتابعة كبيرة، تعقيد خطي كبير، ممانعة ارتباط عالية).

Introduction

Protecting secret data from interception is the large problem of communication and computer security. It is well known that the designers and users of encryption algorithms used in cipher systems needed the ways to find a systematic approach in examining their ciphers prior to use, to ensure that they are safe from various forms of cryptoanalytical attack. [3]

A stream cipher denotes the process of encryption where binary plaintext is encrypted one bit at a time. The simplest and most often used stream cipher for encrypting binary plaintext is where the bit at time interval t of a pseudo random sequence K(t), is combined using XOR (modulo two addition) with plaintext bit, P(t), at time interval t to produce the ciphertext bit at time interval t, denoted by C(t). The sequence K(t) is called the keystream for the stream cipher (See Figure 1). The encryption process can be expressed as:

$$C(t) = P(t) \oplus K(t) \qquad (1)\ [7]$$

Where $\oplus$ denotes modulo two addition. The decryption process can be expressed as:

$$P(t) = C(t) \oplus K(t) \qquad (2)$$

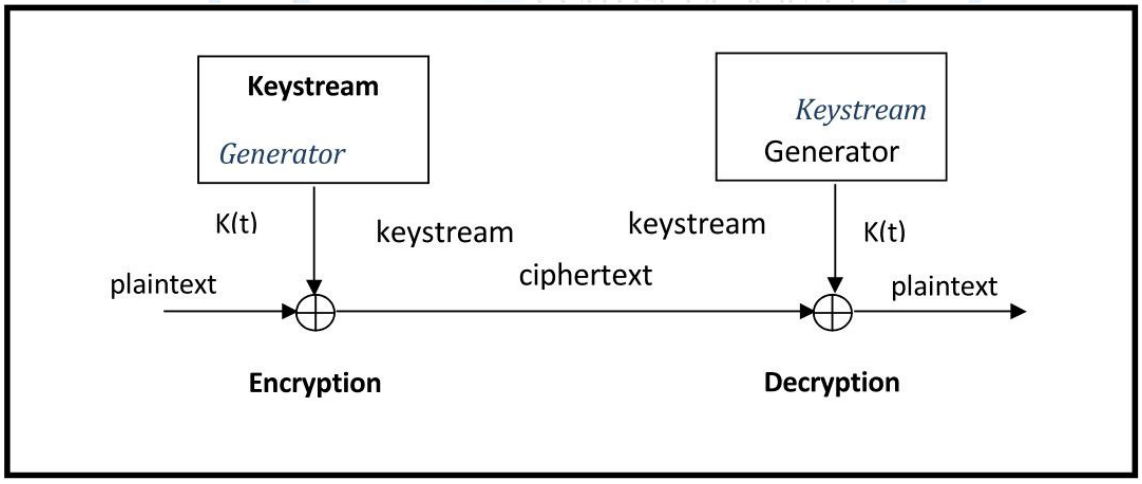

Figure 1: Stream Cipher

It should be noted, as indicated by equations 1 and 2, that both the encryptor and decryptor need to able to generate the same key stream sequence K(t). The key k for the stream

cipher is the initial seed to start the generator. Both the encryptor and dycryptor need to process this key. The crypto key used for encryption is changed randomly so that the ciphertext produced is mathematically impossible to break. The changing of random keys will not allow any pattern to be repeated which would give a clue to the cracker to break the ciphertext. The stream cipher can be either hardware [1, 2, 4] or software [8]. It is clear, that the security of the stream ciphers depends entirely on the keystream generator and can be analyzed in terms of randomness, linear complexity, and correlation immunity [3, 7].

In general there are four basic properties in designing the keystream of figure 1. [7]

1. The period of K(t) should be large.
2. The linear complexity of K(t) should be large.
3. The sequence K(t) should have good randomness.
4. The sequence K(t) should have a high order of correlation immunity.

This paper build a new keystream generator using a new approach based on application of an optimization technique inspired by social behavior observable in nature, such as flocks of birds and schools of fish [6], the proposed approach makes use of a Particle Swarm Optimization (PSO) to generate keys needed for encryption. It will generate a keystream sequence that satisfies the general basic properties (large period, good randomness, large linear complexity, and high order degree of correlation immunity).

Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is a population based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995 [5], inspired by social behavior of bird flocking or fish schooling and swarm theory.

The basic PSO model consists of a swarm of particles, which are initialized with a population of random candidate solutions. They move iteratively through the d-dimension problem space to search for the new solutions, where the fitness, f, can be calculated as the certain qualities measure. Each particle has a position represented by a position-vector $\mathbf{x}_i$ (i is the index of the particle), and a velocity represented by a velocity-vector $\mathbf{v}_i$. Each particle remembers its own best position so far in a vector i-th, and its d- dimensional value is **pbest**(p_{id}).

The best position-vector among the swarm so far is then stored in a vector i-th, and its d-th dimensional value is **gbest**(p_{gd}). During the iteration time t, the update of the velocity from the previous velocity to the new velocity is determined by Eq. 3. The new position is then determined by the sum of the previous position and the new velocity by Eq. 4.

$$v_{id} = w * v_{id} + c_1 * r_1 * (p_{id} - x_{id}) + c_2 * r_2 * (p_{gd} - x_{id}) \quad (3)$$

$$x_{id} = x_{id} + v_{id} \quad (4)$$

where, i =1,2,…..,N; w is the inertia weight, r_1 and r_2 are the random numbers, which are used to maintain the diversity of the population, and are uniformly distributed in the interval [0,1] for the d-th dimension of the i-th particle. c_1 is a positive constant, called as coefficient of the self-recognition component; c_2 is a positive constant, called as coefficient of the social component. From Eq. 3, a particle decides where to move next, considering its own experience, which is the memory of its best past position, and the experience of its most successful particle in the swarm. In order to guide the particles effectively in the search space, the maximum moving distance during one iteration must be clamped in between the maximum velocity $[-v_{max}, v_{max}]$.

The Proposed Algorithm

The following is an algorithmic description of the generating keystream sequences using particle swarm optimization (PSO):

Input: Length of plaintext(denoting the particles keystream length), the algorithm parameters (c_1, c_2, w, v_{max}, Swarm_Size, Max_Iter).

Output: The key having the highest fitness as found by PSO.

Step 1: Randomly generate the initial particles (keys) and velocities to form a swarm, as clarified in subsection (3.1).

Step 2: Calculate the fitness function of each of the particles (keys) as shown in subsection (3.2)

Step 3: If the current position of the particle is better than the previous history, update the particles to indicate this fact.

Step 4: Find the best particle of the swarm. Update the positions of the particles by using equations 3 and 4:

$$v_{id} = w * v_{id} + c_1 * r_1 * (p_{id} - x_{id}) + c_2 * r_2 * (p_{gid} - x_{id})$$

$$x_{id} = x_{id} + v_{id}$$

Step 5: If the maximum number of iterations has exceeded or if the key with very high fitness value is found, then go to step 6 or else go to step 2.

Step 6: Copy the best key obtained so far in the output key variable and exit.

Representation (Initialization)

To apply the PSO algorithm for generating a keystream sequence, a representation of solutions to the problem must be appropriately chosen first. The standard stream cipher uses the binary representation in which each solution (keystream) coded as a binary string.

Fitness Function Calculation

The objective function (fitness function) value for each sequence is calculated by examining the keystream. The objective function used is related to the required properties. Thus, the objective function calculation includes the following:

- Since the required sequence should pass the frequency test, we check that an equal proportion of ones and zeros in the bit stream.

Then:

$$\text{Error\_Frequency} = | n_0 - n_1 | \qquad (5)$$

- Since the required sequence should pass the binary derivative test, then this test is applied to the sequence by taking the overlapping two-tupels in the original bit stream,

thus:

01 becomes 1

10 becomes 1

00 becomes 0

11 becomes 0

We check that an equal proportion of ones and zeros in the new bit stream.

Then:

$$\text{Error\_B-Derivative} = |(count\_n_0 - count\_n_1) - 1| \qquad (6)$$

• Apply the change point test; by checking the change point which is the maximum difference between the proportions of ones including the point and the proportion of ones after the point is noted, thus:

n = the total bit in the stream.

S [n] = the total number of ones in bit stream.

t = the change point.

S [t] = the total number of ones to bit t.

U[t] = n. S[t]-t. S[n]

M = the maximum of ABS (U [t]), for t=1…n.

Where ABS denotes the absolute value

$$\text{Error\_C-point} = \exp(-2M^2/n.S[n].(n-S[n])) \qquad (7)$$

• Since required sequence should pass the serial test, we check the probability of a consecutive entry being equal or different is about the same. This will then give same level of confidence that each bit is independent of its predecessor i.e. from the fact that in random sequence $n_{00}=n_{01}=n_{10}=n_{11}= n/4$.

Then:

$$\text{Error\_Serial} = |(n_{00} - n/4) + (n_{01} - n/4) + (n_{10} - n/4) + (n_{11} - n/4)| \qquad (8)$$

• Since required sequence should pass the run test, then the run test counts the number of runs of ones (blocks) and runs of zeros (gaps) for each possible run length. Which is $1/2^i * n_r$ runs in the sequence are of length i, where n_r is the number of runs in the sequence, M is maximum run length i, and n_i is the number of runs of length i.

Then:

$$\text{Error\_Run} = \sum_{i=1}^{M} \left| \left(\frac{1}{2^i} * n_r \right) - n_i \right| \qquad (9)$$

• Furthermore, the linear complexity of a keystream sequence can be included in the objective function value using the BerleKamp-Massy algorithm.

Then:

$$\text{Error\_L\_complexity} = L - n / 2 \qquad (10)$$

The summation error of every test is calculated, and then the objective function for algorithm is:

$$\text{Fitness\_Function} = 1 / \sum \text{Error} \qquad (11)$$

Results

The work reported in this paper has shown how one of the swarm intelligence techniques called (PSO) can be used to generate a keystream sequence that has (good statistical properties, long period, large linear complexity, and highly order degree of correlation immunity).

During the implementation of the algorithm, the technique has different principle parameters (See Table 1).

Table 1: Parameters selection of keystream generator

Parameters	Symbol	Value
Self confidence	c_1	1.5 – 2.0
Swarm Confidence	c_2	1.5 – 2.0
Inertia weight	W	1.2
Number of particles in the swarm	Swarm_Size	20-40
Maximum number of iteration	Max_Iter	100-500

The PSO algorithm for keystream sequence generator is run a number of times for different parameters values has shown in Table 1. At each factor, ten runs were performed. The results of applying our technique (PSO) algorithm to produce a keystream sequence that satisfies the requirements, for the PSO components the search was terminated when maximum number of iteration is reached. The results of maximizing the objective function values,are also show.

The algorithm performance of algorithm is applied on the sequence of length (2000-5000) bits. The maximum number for each iteration to find the solution is calculated. The statistical tests are applied for the best solution of each iteration that generated from applyed PSO algorithm to decide whether a sequence has passed or failed the test. So there must be a statistical values corresponding to truly random sequences and then set a pass mark. As an illustration, if the pass mark is 95%. This means that a given sequence passes the test if its value lies in the range in which there is exception to find 95% of all sequence. It is usual to denote the pass mark as (100-α), where α is called the significance level of the test. Throughout the tests that applied are (frequency test, binary derivative test, change point test, serial test, poker test, run test, and autocorrelation test).

Table 2 gives iterations that have the best solution which passed the statistical tests. Further more these results passed the statistical tests were applied by Berle_Kamp Massey algorithm [7] for each best solution of each iteration and the solutions give the perfect linear complexity.

Table 2: Test results with (2000 bits, Swarm_Size = 20, Max_Iter=100)

Run No.	C_1=1.5, C_2=1.5	0_1=1.5, 0_2=2	0_1=2, 0_2=1.5	0_1=2, 0_2=2	Statistical Tests
	Best Value	Best Value	Best Value	Best Value	
1	53	80	66	64	Pass
2	54	57	51	73	Pass
3	92	63	89	66	Pass
4	67	66	72	52	Pass
5	50	34	44	83	Pass
6	52	58	70	47	Pass
7	61	62	72	55	Pass
8	75	82	83	66	Pass
9	65	74	51	80	Pass
10	88	80	63	52	Pass

This approach reduces the encryption time and also the keystream length will always increase in each iteration until a solution is obtained. Figure.2 shows the comparison between the times consuming by encryption using PSO algorithm for a keystream with varying plaintext bits length.

Figure 2: Results for time consumed with varying plaintext bits length

Conclusion

Heuristic approach to the design of cryptographiclly strong keystream have been presented. These pseudo random techniques provide a suitable alternative to systematic methods for the construction of cryptographiclly strong random key generator. From studying the solutions of iterations by applying our new generator and according to randomness level and degree of complexity, it is easy to notice that the new generator has generated a large number of solutions that satisfied our requirements (large period, good randomness, large linear complexity, and high order degree of correlation immunity).

The algorithm is robust, since under the generation mechanism and acceptance criterion rules, it's find solution that does not strongly depend on the choice of the initial state, for this reason it can be public.

The best values of the PSO parameters may depend not only the problem being solved, but also on the type and size of instance at hand. Generally to get best results, long iterattions runs must be allowed.

Designing of cryptosystem using this new approach does depend on the number of parameters that considered the secret key of the algorithm. Thus, that increases the complexity of the algorithm against the cryptanalytic attack.

References

1. Deepthi.P.P,Deepa Sara John,P.S.Sathidevi,"**Design and Analysis of a Highly Secure Stream Cipher Based on Linear Feedback Shift Register**", Computers and Electrical Engineering, Volume 35, Issue 2, pp 235-243, March 2009.
2. Good T. and Benaissa M., "**Hardware Results for Selected Stream Cipher Candidates**". State of the Art of Stream Ciphers (SASC 2007), pp 191-204, 2007.
3. Gustafson H.,"**A Computer Package Measuring the Strength of Encryption Algorithms**", Elsevier Science Ltd, 1994.

4. Hell.M, Johansson.T, Maximov.A. Meier.W." **A Stream Cipher Proposal: Grain**". IEEE International Symposium on Information Theory, pp 1614 – 1618, July 2006.
5. Kennedy J. and Eberhart R., "**Particle Swarm Optimization**", in Proceedings of the IEEE International Conference on Neural Networks, pp. 1942-1948, 1995.
6. Reynolds C.W,"**Flocks, Herds and Schools: a distributed behavioral model**", In Computer Graphics, volume 21, pp: 25–34, 1987.
7. Schneier B.,"**Applied Cryptography, Protocols, Algorithms, and Source Code in C Language**", John Wiley and Sons, 1996.
8. Common wealth Office of technology, Monthly cyber security tips, Volume 3 Issue 5, May 2008.