European Journal of Scientific Research
ISSN 1450-216X Vol.55 No.2 (2011), pp.237-244

http://www.eurojournals.com/ejsr.htm

Cryptanalysis of DES using Computational Intelligence

Vimalathithan. R.
Sri Krishna College of Engineering and Technology, Coimbatore, India
E-mail: athivimal@gmail.com

M. L. Valarmathi
Government College of Technology, Coimbatore, India

Abstract

Cryptanalysis of block cipher is a challenging task due to non-linearity in nature. Recently Cryptanalysis using Computational Intelligence pave the way to break the block ciphers. In this paper, by combining the effectiveness of Genetic algorithm and Particle Swarm optimization, a novel approach, called Genetic Swarm optimization is proposed and applied in the field of cryptanalysis for attacking DES. A known plaintext attack is used and varieties of optimum keys are computed. Through this approach, the optimum key can be found faster without searching the entire key space. The experimental result indicates that Genetic Swarm Optimization can be used in the field of cryptanalysis efficaciously in order to break the key and also can be applied to attack other block ciphers.

Keywords: Cryptanalysis, DES, Fitness function, Genetic Algorithms, Particle Swarm Optimization, Genetic Swarm Optimization.

1. Introduction

Cryptology is the art and science of making secret codes and breaking secret codes. Cryptography is the technique to camouflage the message, and only intended recipients can remove the camouflage and read the message. The message we want to send is called plaintext and the camouflage message is called ciphertext. The process of converting a plaintext into ciphertext is called encryption and the reverse process is called decryption. Cryptanalysis is a study of mathematical techniques to break cryptographic algorithms and attack the cipher text, without accessing the secret key [1].

Brute-Force attack is one type of attack used for cryptanalysis, which makes an attempt to guess every possible key that might be in use. Data Encryption standard (DES) is a symmetric block cipher which takes 64 bit plain text and generates 64 bit cipher text using 56 bit key [2]. To do Brute-Force attack on a given DES cipher text block, the antagonist needs to check all possible keys i.e., 2^{56} keys. However, with the available technology, it is possible to check one million keys per second. This means we need two thousand years to do Brute Force attacks on DES using a single computer with single processor. If 3500 networked computers can find the key in 120 days, a secret society with 42,000 members can find the key in 10 days [3]. Cryptographic algorithms are almost designed to make the brute force attack unworkable. Generally, the key length considered by any secret key based encryption algorithm is large enough so that it is not possible for an attacker to try for every possible key. But Computational Intelligence (CI) effectively solves the problem without searching the entire key space.

CI can be considered as the study of adaptive mechanisms that enable intelligent behavior of a system in complex and changing environments like Genetic Algorithms (GA) and Particle Swarm Optimization (PSO) [21]. GA and PSO is a population based optimization which could be applied to solve optimization problems. Unlike the GA, PSO has no evolution operations like crossover and mutation. The strength of PSO is its fast convergence, which compares favorably with global optimization algorithm like GA. Both GA and PSO share common elements and initialize a population in a similar manner and evaluate a cost or fitness function. At last both are generational. By combining the effectiveness of GA and PSO, a new hybrid evolutionary technique called Genetic Swarm Optimization (GSO) is used here. GSO strongly integrates the vantage characteristics of GA and PSO.

Several researches are carried out in this area. Biham (1992) attacked DES using Differential cryptanalysis, where 247 chosen plaintexts were required for attacking [4]. Matsui (1993) attacked DES using Linear Cryptanalysis [5]; in this case 243 known plaintexts were required for attacking DES and it is easier to acquire known plaintext rather than chosen plaintext. J. song (2007) used GA for the cryptanalysis of DES reduces to two rounds [8]. Their results show that GA can be used to attack DES, in the same year J. Song used GA to attack DES reduced to four rounds [9]. This is easy to attack since only two rounds and four rounds were considered for attacking. Yang.F (2007) used Evolutionary algorithm in the field of cryptanalysis to attack DES reduced to six rounds [11]. Their results were effective to attack the 49 bits key and the 42 bits key for DES cipher reduced to six rounds or less. The paper by Waseem Shahzad (2009) presents a PSO algorithm based approach for the cryptanalysis of DES reduced to four rounds [6]. They compared the results for DES reduced to four rounds and eight rounds using PSO and GA. Hamdani (2010) used Binary Artificial Immune system to attack DES reduced to four rounds [7]. S.Khan (2010) used Ant Colony Optimization for the cryptanalysis of four round DES [10]. In recent years many researches are going in the field of Cryptanalysis using Evolutionary computation.

Researches considering DES with reduced rounds are found easy to attack as the number of rounds is considerably reduced. Instead, the actual DES consists of sixteen rounds which are difficult to attack. Our goal is to take a difficult problem and solving it effectively using Computational Intelligence. Hence our work concentrates on attacking actual DES using CI. In this paper, we adapt a known plaintext attack where the attacker has samples of plaintext and its corresponding ciphertexts. Here we present a new approach by combining the GA and PSO, called Genetic Swarm Optimization which is applied in the field of cryptanalysis of DES using known plaintext attack. Experimental results indicate GSO is an efficacious tool to break DES.

The rest of the paper is organized as follows: In Section 2 we present a brief overview of DES and background of Computational Intelligence. In section 3, we describe the used fitness function and our proposed approach using CI to attack DES in section 4. Experimental results are presented in Section 5. Finally, section 6 concludes our paper.

2. Overview

2.1. Basics of Data Encryption Standard

The most widely used encryption scheme is symmetric block cipher called as Data Encryption standard (DES) published by National Institute of standards and Technology (NIST). As with any encryption scheme, there are two inputs to the encryption function: the plaintext to be encrypted and the key. At the encryption site DES takes a 64 bit plan text using 56 bit key and creates a 64 bit cipher text. The encryption process is made of two permutations and sixteen Feistel rounds, where the 64-bit plain text is divided into 32-bit parts, the left part L_i and the right part R_i. The round i, $1 \leq i \leq 16$ is defined as follows

$$Li = R_{i-1},$$

$$R_i = L_i \oplus F(R_{i-1}, K_i)$$

Where K_i is the cipher key derived from the key scheduling algorithm, which is the 48 bits sub key used in the i^{th} round, and F is a round function. The round function F is the non-linear component

of DES and consists of three operations: substitution, permutation and XOR operations. There are eight S-boxes where each one is a substitution mapping 6 to 4 bits. The details of DES are given in [2, 3]. The weakness of DES is out of the 2^56 keys there are four weak keys, six semi weak keys and 48 possible weak keys. The weak key is the one that, after parity drop operation, consist of either of all 0s, all 1s or half zeros and half ones. For example if we take the key '0101010101010101' the actual 56 bit key contains all 0s and the key 'FEFEFEFEFEFEFEFE', the actual 56 bit key is all 1s. A semi weak key creates only two different round keys and a key that creates only four distinct round keys is possible weak keys [3, 13].

2.2. Background of Computational Intelligence

This section briefly explains GA, PSO and GSO.

2.2.1. Genetic Algorithm

Genetic Algorithm is an optimization technique which based on the principles of genetics and natural selection. These algorithms are computationally simple. A GA uses set of chromosomes (population) to evolve under specified selection rules in order to maximize the fitness. A simple genetic algorithm that yields good results in many practical problems is composed of three operators: Selection (Reproduction), Crossover and Mutation [15-18].

Selection strategies determine which chromosome will take part in the evolution process. The different Selection strategies are Population Decimation, Proportionate selection and Tournament Selection [17]. After the selection, the next operation is mating scheme. Selection Strategies are involved with selecting which individuals will take part in the evolution process, the mating scheme select which two parent chromosomes will mate with one another. The mating scheme that exists includes Best-Mates Worst, Adjacent Fitness Pairing and Emperor Selective Mating. The next operation is Crossover, which selects genes from parent chromosome and creates a new offspring. This operator randomly selects some crossover point and everything before this point copy from a first parent and everything after this point copy from the second parent. The two newly generated chromosomes may be better than their parent chromosome and the evolution process may continue. Crossover is continued by Mutation. This operator randomly changes one or more bits in the Chromosome. The purpose of this operator is to prevent the population to escape from minimum value. The mutation is carried out according to the mutation probability $P_{mutation}$. Mutation rate must be low.

2.2.2. Particle Swarm Optimization

Swarm Intelligence is an innovative paradigm used for solving the complicated problems. PSO is a population based optimization tool which could be implemented and applied to solve various optimization problems [19, 20].

The Canonical 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 the new solutions, where the fitness F_k, can be calculated as the certain quality measure. Each particle has a position represented by a position-vector x_i (i is the index of the particle), and a velocity represented by a velocity-vector v_i. Each particle remembers its own best position so far in the vector $x_i^{\#}$ (pbest), and its j-th dimensional value is $x_{ij}^{\#}$. The best position-vector among the swarm so far is then stored in a vector x*(gbest), and its j-th dimensional value is x_j*. During the iteration time t, previous velocity ($v_{ij}(t)$) is updated to the new velocity ($v_{ij}(t+1)$), determined by Eq.(1). The new position is then determined by the sum of the previous position and the new velocity, as given by Eq.(2).

$$v_{ij}(t+1) = w.v_{ij}(t) + c_1r_1(x_{ij}^{\#}(t) - x_{ij}(t)) + c_2r_2(x_j^{*}(t) - x_{ij}(t)) \quad (1)$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \quad (2)$$

Where 'w' is called as the inertia factor, r1 and r2 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 j-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. (1), 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.

2.2.2.1. Binary PSO

The canonical PSO is basically developed for continuous optimization problems. In our case, the cryptanalysis problem deals with binary information. Hence the canonical PSO cannot be applied directly. In our problem, the variable x_{ij} represents the key in binary form, hence x_{ij} (t) should take 0 or 1, but from the equation (1), we can see that the results of v_{ij} (t+1) may not be an integral, and from equation (2) we can see x_{ij} (t+1) may take numerical other than 0,1 after iteration. The equations have to be adjusted in such a way that the velocity and position are to be in binary form. In the binary PSO, we can define a particle's position and velocity in terms of changes of probabilities that will be in one state or the other [14].At each time step, each particle updates its velocity and moves to a new position according to the Eq.(3) and (4):

$$v_{ij}(t+1) = w.v_{ij}(t) + c_1r_1(x_{ij}^{\#}(t) - x_{ij}(t)) + c_2r_2(x_j^{*}(t) - x_{ij}(t)) \quad (3)$$

$$x_i(t+1) = 1 \text{ if } \rho \leq s(v_i(t)), 0 \text{ otherwise.} \quad (4)$$

Where 'ρ' is a random function in the closed interval[0, 1]. The Velocity update equation is similar to that of canonical PSO. The only difference is in the position update as given in the equation (4).

2.2.3. Genetic Swarm Optimization

Genetic Swarm Optimization is a hybrid evolutionary technique that combines the effectiveness of GA and PSO. The basics of GSO are discussed in [12]. In each iteration, the population is randomly divided into two parts. Each part is taken as the population for GA and PSO respectively and for the new set of population, fitness is computed. The populations were recombined in the updated population which is again divided into two parts in the next iteration for the next run of GA or PSO. This process continues until the fitness value is converged.

An important parameter called hybridization coefficient h_{coeff} drives the GSO algorithm. h_{coeff} is expressed in terms of percentage of the population. In each iteration h_{coeff} percentage of the total population is processed by GA and the remaining (1- h_{coeff}) percentage of the total population is processed by PSO. For example, if $h_{coeff} = 0$, then the whole population is processed by PSO operators, i.e., the algorithm becomes purely PSO and if $h_{coeff} = 1$, then the whole population is processed by GA operators, i.e., the algorithm becomes purely GA. While $0 < h_{coeff} < 1$ means the corresponding percentage of h_{coeff} population is processed by GA and the remaining population is processed by PSO.

3. Fitness Function

The main task in formulating the cryptanalysis problem using CI is to find out the effective fitness function. Defining a fitness function is a difficult task. Once the effective fitness function is defined, the problem becomes facile. The fitness function for Known Plaintext attack must relate the plaintext, ciphertext and key used. Hence the correlation between the known ciphertext and generated ciphertext is used as fitness function. Using the randomly generated keys, the known plaintexts were encrypted to generate ciphertexts C_g. The fitness function used for Known plaintext attack is given by,

$$F_{kp} = (C_k * C_g)/64 \quad (5)$$

where C_k is the known cipher text and C_g is the generated cipher text and the $C_k * C_g$ represents the number of identical position between C_k and C_g. The fitness value lies in the range (0, 1). If all the bit patterns between C_k and Cg are identical then the fitness value takes one.

4. Proposed Research Method

In this section, we describe our proposed research approach and illustrate the effectiveness of our algorithm. The goal is to maximize the fitness function.

4.1. Attacking the Key using GA

In this subsection, the procedure to carry out the cryptanalysis using GA in order to break the key is described.

1. Initial keys were generated randomly. The number of keys considered initially represents the population size. The results show that it is better to consider a low population size and increase the number of generations. So that the crossover rate is high.
2. Using the randomly generated keys, encrypt the known plaintext to generate the ciphertext, compute the Fitness function F_{kp}.
3. The computed fitness value is compared with the $Fitness_{max}$. If the computed fitness value is greater than or equal to the $Fitness_{max}$, we can conclude that the corresponding key with maximum fitness is the optimum key.
4. If the condition in step 3 is not met, then apply GA parameters.
5. Select the parent keys to generate a new set of children keys.
6. Parent keys were mated to generate children keys. Do the crossover and mutation. Random point crossover is preferred.
7. Compute the fitness for the new set of keys and go to step 3.

Ingeminate the steps from 2 to7 until the fitness is maximized or the maximum number of generation are reached. If the maximum number of generation is reached then the key with maximum fitness in the final generation corresponds to optimum key.

4.2. Attacking the Key using PSO

Here we propose how PSO can be applied to break the Cipher key. Now the problem is to find the key using PSO. In a swarm of particles, each particle represents a key. Initialize the swarm Particles X_i. Using the generated particles, encrypt the known plaintext and generate the ciphertext C_g. Now compute the fitness function using the equation (5). Update the velocity and position of the particle (key) using the equation (3) and (4).The best position is associated with the maximum fitness value i.e., F (P_{ibest}) of the particle P_{ibest} and Global best P_{gbest} is the best position among all particles in the swarm which is achieved so far. The global position is associated with the global fitness value, F (P_{gbest}) of the particle P_{gbest}.

Table 1: Algorithm for Cryptanalysis of DES using PSO

1.	Set i) Number of iterations (ii) $Fitness_{max}$. (iii)Initialize the swarm Particles
2.	Generate the ciphertext from the known plaintext using generated particles.
3.	Compute Fitness Value using equation 5.
4.	Update the Velocity and Particle's position according to equation 3 and 4
5.	Update the position of gbests and pbests.
6.	If $F(X_i(t)) > F(P_{ibest})$ then $P_{ibest} = X_i(t)$ and
7.	if $F(X_i(t)) > F(P_{gbest})$ then $P_{gbest} = X_i(t)$
8.	Check for Stopping Criteria. Repeat steps 3-5 until the stopping criteria are satisfied.
9.	Display Key found: Key = P_{gbest}

The process is continued either until the fitness function is maximized or maximum number of iteration is reached. If there is no improvement in the fitness value for some iteration continuously then the algorithm is stopped. Algorithm for finding the key using PSO is shown in Table 1.

4.3. Genetic Swarm Optimization

In this section, we propose our novel approach GSO to attack DES. Initialize the population randomly and select the hybridization coefficient. Compute the fitness value for the entire population and check

for the maximal fitness value. Then apply GSO, here we take $h_{coeff} = 0.3$; i.e., 30% of the population were processed by GA and the remaining 70% of the population were processed by PSO to generate new population. Figure1 shows how keys can be evaluated for next generation using GSO. Continue the evolution process using GSO algorithm until the stopping criteria are met. The algorithm is shown in table2.

Figure 1: Evolution process for key generation using GSO

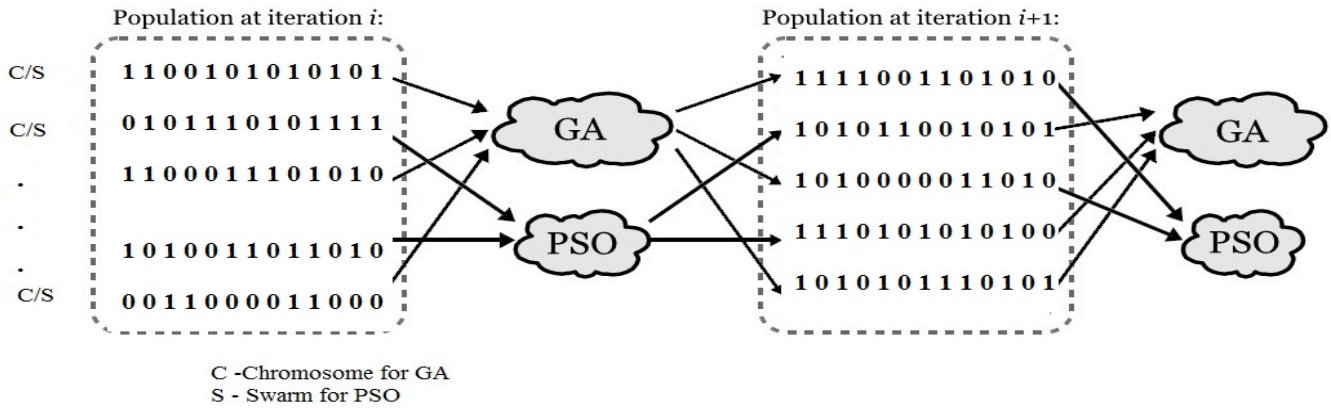


Table 2: Algorithm for Cryptanalysis of DES using GSO

1.	Initialize POPGSO Randomly
2.	Select hcoeff POPGA =hcoeff * PGSO individuals for GA POPPSO =(1-hcoeff)* POPGSO individuals for PSO
3.	For POPGA : Generate new population by Applying GA For POPPSO : Generate new population by applying PSO
4.	Update New population POPGSO= POPGA + POPPSO
5.	Compute the fitness for POPGSO
6.	Check for Stopping criteria. Repeat steps 2-5 until stopping criteria is satisfied.
7.	If stopping criteria is met, then Key =$Key_{optimum.}$

4.4. Procedure to Find the Key

In all the cases, we use 500 runs (R) and in each run 1000 iterations were performed. Each run will generate optimum keys. The optimum keys is selected based on the fitness value greater than ψ ($Fitness_{max}$). The number of ones and zeroes were counted at each position and each count is divided by the number of Runs 'R', if the result is greater than a threshold value χ, these bits can be decided. The bits were fixed in particular position and the algorithm is preceded until all the bits were found. Suppose three optimum keys were found in three runs then with a threshold value χ =0.67, the key deduction is shown in table 3. For simplicity 8 bits are considered here (but the actual key is 56 bits). The proposed algorithms were run, until all the bits were deduced.

Table 3: Example for key deduction

Bit Position/ key	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Key 1	1	1	0	1	1	0	0	0
Key 2	1	1	1	1	0	1	0	1
Key 3	0	1	1	0	1	0	0	1
Count 0	1	0	1	1	1	2	3	1
Count 1	2	3	2	2	2	1	0	2
(Count 0) /R	0.33	0	0.33	0.33	0.33	0.67	1	0.33
(Count 1)/R	0.67	1	0.67	0.67	0.67	0.33	0	0.67
Assigned bits	1	1	1	1	1	0	0	1

5. Experimental Setup and Results

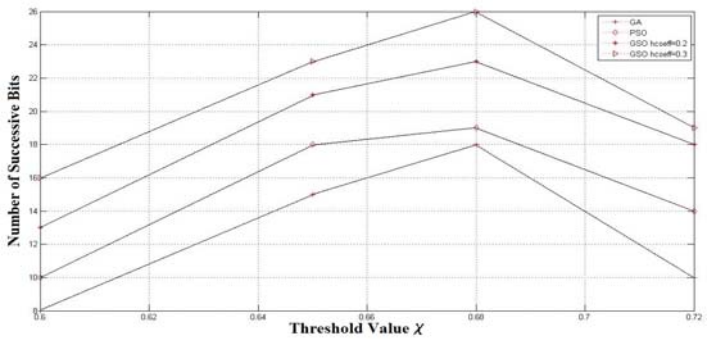
Numbers of experiments were carried out to show the effectiveness of GSO. The experiments were conducted using Matlab 7.5 in an Intel Core^{i3} (2.4GHz) System. The fixed number of Generation N is 1000 and run R is 500. The known plaintexts and its corresponding cipher texts pair were taken for the analysis. The known plaintext is encrypted using the initial population to generate the ciphertext and the fitness value is calculated using equation (5). The objective is to maximize the fitness function. If a key have a fitness value greater than $Fitness_{max}$ value, then the corresponding key is the optimum key. In each run an optimum key is generated, then count the number of ones and zeros in each position in the optimum keys (as explained in section 4.4) and divide it by number of runs. The threshold parameter χ determines the bits in each position. If the count value of ones exceeds the threshold value χ then fix the bit as '1' else fix the corresponding bit as '0' and continue this until all the bits were ascertained. Using these bits as seed and run the algorithm. The algorithm is run three times to deduce the key.

The table 4 shows the results for attacking the key used for DES. In case of Genetic algorithm, the hybrid coefficient of GSO is taken as 1.0, and for PSO, the hybrid coefficient is taken as '0'. Hence GA and PSO can be considered as a special case of GSO with h_{coeff}=1.0 and '0' respectively. In case of GSO, the hybrid coefficient is taken as 0.2 and 0.3. Results show that when the hybrid coefficient takes 0.3, GSO performs better than GA and PSO.

Table 4: Comparison between GA, PSO and GSO for the cryptanalysis of DES.

$Fitness_{max}$ ψ	Threshold Value χ	Number of iterations	Runs R	Successive bits (GA)	Successive bits (PSO)	Successive bits (GSO with h_{coeff}=0.2)	Successive bits (GSO with h_{coeff}=0.3)
0.8	0.62	1000	500	8	10	13	16
0.8	0.72	1000	500	10	14	18	19
0.75	0.65	1000	500	15	18	21	23
0.78	0.69	1000	500	18	19	23	26

Figure 2: Threshold Value Vs Succesive Bits



The above results show that, in each case the GSO performs effectively than GA and PSO which can be observed in figure which shows a plot of threshold value Vs. Number of successive bits. By further tuning the parameters of GA and PSO, effective results can be obtained from GSO. From the valuable bits obtained, fix these bits and continue until the entire key is found.

6. Conclusions

In this paper, we have presented a novel approach Genetic Swarm optimization algorithm for the cryptanalysis of Data Encryption Standard to breaks the key successfully. The fitness function used here is not restricted and can be applied for other block ciphers also. The fitness function used here can be used only for known plain text attack and cannot be applied for Cipher text only attack. The future work is to create a better fitness function and to apply GSO for attacking cipher text only for DES. In

addition our attention is to extend our research in the Cryptanalysis of other block ciphers like S- AES and AES using GSO.

References

[1] Neal Koblitz, A course in Number Theory and Cryptography, Springer International Edition, 2008.

[2] William Stallings, Cryptography and Network Security Principles and Practices, Pearson Education, 2004.

[3] Behrouz A. Forouzan, Cryptography and Network Security, Tata McGraw hill Education, 2nd edition 2008.

[4] Biham and Adi Shamir, Differential Cryptanalysis of the Full 16-round DES. Advances in Cryptology -- CRYPTO '92. Springer-Verlag. 487-496. (1992)

[5] M.Matsui. Linear Cryptanalysis Method for DES Cipher, EUROCRYPT'1993. LNCS 765, Springer, 1993, 386-397

[6] Waseem Shahzad, A.B siddiqui, F.Aslam Khan, Cryptanalysis of Four Round DES using Binary Particle Swarm Optimization, GECCO'09 pp:2161-2166.

[7] Hamdani, S.A.A., Shafiq, S., Khan, F.A.: Cryptanalysis of Four-Rounded DES using Binary Artificial Immune System. In: ICSI 2010, Part I, LNCS 6145, pp. 338-346 (2010)

[8] Song. J., Zhang, H., Meng, Q., Wang, Z.: Cryptanalysis of Two-Rounded DES Using Genetic Algorithms. In: ISICA 2007, LNCS 4683, pp. 583-590 (2007).

[9] Song J, Zhang, H., Meng, Q., Wang, Z.: Cryptanalysis of Four-Rounded DES Based on Genetic Algorithms, IEEE 2007, pp. 2326-2329

[10] Khan, S., Shahzad, W., Khan, F.A. : Cryptanalysis of Four-Rounded DES using Ant Colony Optimization, IEEE 2010.

[11] Yang, F., Song, J., Zhang, H., Quantitative Cryptanalysis of Six-Rounded DES using Evolutionary Algorithms. In: ISICA 2008, LNCS 5370, pp. 134-141 (2008)

[12] A. Gandelli, F. Grimaccia, M. Mussetta, P. Pirinoli, R.E. Zich, "Development and Validation of Different Hybridization Strategies between GA and PSO", Proc. of the 2007 IEEE Congress on Evolutionary Computation, Sept. 2007, Singapore, pp. 2782–2787.

[13] A Menezes, P.Vanoorschoot, S.Vanstone Handbook of Applied Cryptography, CRC Press, 1996.

[14] J, kennedy, R.Eberhart, "A discrete Binary version of the Particle Swarm Algorithm," International Conference on Neural network, Vol. IV, pp:4104-4108, Australia, 1997.

[15] Collin R.Reeves, J,E Rowe, Genetic Algorithms-Principles and Perspectives, A guide to GA theory, Kluwer Academic Publishers.

[16] Haupt R.L, Haupt S.E., Practical Genetic Algorithms, 2nd ed., Wiley, 2004.

[17] M.Mitchell, An Introduction to Genetic Algorithms, First MIT press paperback edition, 1998.

[18] GoldBerg D.E, "Genetic Algorithm in Search, Optimization and Machine Learning", Boston, Addision-Wesley, 1999

[19] J, kennedy, R.Eberhart, "Particle Swarm Optimization," IEEE international Conference on Neural Networks, pp:1942-1948,Australia,1995

[20] Nadia Nedjah, Ajith Abraham, Luzia de Macedo Mourelle, Swarm Intelligent systems, Studies in Computational Intelligence, Vol.26, 2006.

[21] Nadia Nedjah, Ajith Abraham, Luzia de Macedo Mourelle, Computational Intelligence in Information Assurance and Security, Studies in Computational Intelligence, Vol. 57, 2007.