Iranian Journal of Mathematical Sciences and Informatics
Vol. 2, No. 2 (2007), pp 45-56
DOI: 10.7508/ijmsi.2007.02.006

FINDING HIGHLY PROBABLE DIFFERENTIAL CHARACTERISTICS OF SUBSTITUTION-PERMUTATION NETWORKS USING GENETIC ALGORITHMS

MAHDI ABADI[a][1], BABAK SADEGHIYAN[b], ABBAS GHAEMI[c], MOHAMMAD A. ALIPOUR[d]

[a] TARBIAT MODARES UNIVERSITY, P.O.BOX 14115-143, TEHRAN, IRAN
[b] AMIRKABIR UNIVERSITY OF TECHNOLOGY, TEHRAN, IRAN
[c] FERDOWSI UNIVERSITY OF MASHHAD, MASHHAD, IRAN
[d] UNIVERSITY OF KASHAN, KASHAN, IRAN

E-MAIL: ABADI@MODARES.AC.IR

Abstract. In this paper, we propose a genetic algorithm, called GenSPN, for finding highly probable differential characteristics of substitution-permutation networks (SPNs). A special fitness function and a heuristic mutation operator have been used to improve the overall performance of the algorithm. We report our results of applying GenSPN for finding highly probable differential characteristics of Serpent block cipher. A comparison of the resultant characteristics with the previously published works shows that GenSPN can find differential characteristics of higher probabilities.

Keywords: Differential Characteristic, Differential Cryptanalysis, Genetic Algorithms, Substitution-Permutation Network, Serpent.

2000 Mathematics Subject Classification: 94A60, 68P25.

1. Introduction

Differential cryptanalysis is a method, which analyzes the effect of particular differences in plaintext pairs on the differences of the resultant ciphertext pairs

[1]Corresponding author.

Received 20 December 2007; Accepted 17 May 2008

[2]. Differential cryptanalysis is usually a chosen plaintext attack, meaning that the attacker must be able to obtain encrypted ciphertexts for some set of plaintexts of his choosing. There are, however, extensions that would allow a known plaintext or even a ciphertext-only attack. The basic method uses pairs of plaintext related by a constant difference; difference can be defined in several ways, but the eXclusive OR (XOR) operation is usual. The attacker then computes the differences of the corresponding ciphertexts, hoping to detect statistical patterns in their distribution. The resulting pair of differences is called a *differential* [5]. In the basic attack, one particular ciphertext difference is expected to be especially frequent; in this way, the cipher can be distinguished from random. More sophisticated variations allow the key to be recovered faster than exhaustive search [6].

For any particular cipher, the input difference must be carefully selected if the attack is to be successful. An analysis of the algorithm's internals is undertaken; the standard method is to trace a path of highly probable differences through the various stages of encryption, termed a *differential characteristic.*

The process of finding the best differential characteristic for block ciphers is very time-consuming, and the complexity of the process usually increases exponentially with the addition of rounds [3].

In this paper, we propose GenSPN, a genetic algorithm for finding highly probable differential characteristics of substitution-permutation networks (SPNs). We report our results of applying GenSPN for finding highly probable differential characteristics of Serpent block cipher. We also compare the resultant characteristics with the previously published works.

The remainder of this paper is organized as follows: Section 2 introduces SPNs, Section 3 provides an overview of genetic algorithms, and Section 4 describes our proposed genetic algorithm for finding highly probable differential characteristics of SPNs. Section 5 reports the results of the experiments done to find highly probable 5 and 6-round characteristics for differential cryptanalysis of 6 and 7-round Serpent block cipher and finally Section 6 draws some conclusions.

2. Substitution-Permutation Networks

Shannon suggested that practical and secure product ciphers may be constructed using a mixing transformation consisting of a number of layers or rounds of *confusion* and *diffusion.* The substitution-permutation networks (SPNs) [5] structure is directly based on the above concepts.

An r-round SPN requires $(r+1)$ n-bit subkeys. Each round consists of three layers of *key mixing*, *substitution*, and *linear transformation.* In the key mixing layer, the n-bit round input is bitwise XORed with the subkey for that round. In the substitution layer, the resulting block is partitioned into l subblocks of size m, and each subblock becomes the input to a bijective $m \times m$ substitution box (S-box). In the linear transformation layer, the output from the substitution stage is processed through an invertible n-bit linear transformation. The linear transformation is usually omitted from the last round, since it is easily shown that its inclusion adds no cryptographic strength to the SPN. A final subkey is XORed with the output of round r to form the ciphertext.

2.1. Serpent

Serpent [1] is a 32-round SPN, which encrypts a 128-bit plaintext P to a 128-bit ciphertext C in 32 rounds under the control of 33 128-bit subkeys. The cipher itself consists of:

- An initial permutation IP;
- 32 rounds, each consisting of a key mixing operation, a pass through S-boxes, and (in all but last round) a linear transformation. In the last round, this linear transformation is replaced by an additional key mixing operation;
- A final permutation FP.

The rounds are numbered from 0 to 31, where the first round is round 0 and the last is round 31. Serpent has a set of eight 4-bit to 4-bit S-boxes. Each round function $R_i (i \in \{0, \dots, 31\})$ uses only a single replicated S-box. For example, R_0 uses S_0, 32 copies of which are applied in parallel. The set of eight S-boxes is used four times. Thus, after using S_7 in round 7, again S_0 in round 8, S_1 in round 9, and so on are used.

3. Genetic Algorithms

Genetic algorithms (GAs) are adaptive methods, which may be used to solve search and optimization problems [4]. In genetic algorithms, the term *chromosome* typically refers to a candidate solution to a problem, often encoded as a bit string. The *genes* are either single bits or short blocks of adjacent bits that encode a particular element of the candidate solution. A collection of chromosomes is called a *population*.

To solve a problem with genetic algorithms, a fitness function must be devised for that problem. Given a particular chromosome, the fitness function returns a single numerical fitness, or figure of merit, which is supposed to be proportional to the utility or ability of the individual, which that chromosome represents. Selection, crossover, and mutation operators [8] are usually used in genetic algorithms for reproduction. Each of the operators is introduced as the following:

- **Selection**
  This operator selects some chromosomes in the population for reproduction. The fitter the chromosome, the more times it is likely to

be selected to reproduce. Different selection methods such as Fitness-Proportionate, Elitism, Boltzmann, and Tournament [8] are proposed to be used in genetic algorithms.

- **Crossover**
  This operator takes a pair of chromosomes amongst current generation and produces a new pair of chromosomes called *offspring*. Crossover is not usually applied to all pairs of chromosomes selected for mating. Normally, the likelihood of crossover for each pair of chromosomes is considered between 0.60 to 0.95, which is called *crossover rate* or *crossover probability* and represented as P_c. If crossover is not applied, offspring are produced simply by duplicating the parents. This gives each chromosome a chance of passing on its genes without the disruption of crossover.
- **Mutation**
  Mutation is applied to each offspring individually after crossover. It randomly alters each gene with a small probability called *mutation rate* or *mutation probability* and represented as P_m. After the mutation is done, the generated chromosomes, considered as new generation, are sent for the next run of the algorithm.

4. GenSPN

In this section, we describe GenSPN, a genetic algorithm for finding highly probable differential characteristics of SPNs. First, the chromosome structure and the process of initial population generation are presented. Then, the fitness function and the genetic operators of the proposed genetic algorithm are introduced.

4.1. Chromosome Structure

To find a k-round differential characteristic of an n-bit SPN, we assume that each chromosome consists of $k+1$ genes, each of which is an n-bit string. The first gene represents input difference of round 1 and the next k genes represent output difference of rounds 1 to k. By input/output difference of a round, we mean the input/output difference of substitution functions in that round. In Figure 1, the structure of chromosomes used to find a 5-round characteristic is illustrated. Regarding the chromosome structure, the input difference of round 2 onwards can be calculated by applying a linear transformation to output difference of its prior round.

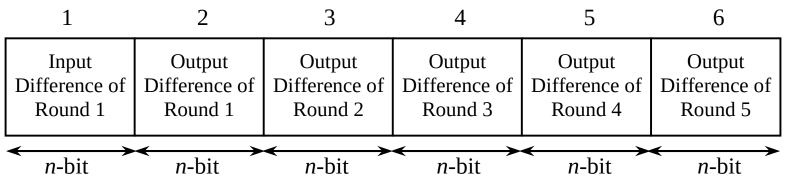
1
2
3
4
5
6
Input Difference of Round 1
Output Difference of Round 1
Output Difference of Round 2
Output Difference of Round 3
Output Difference of Round 4
Output Difference of Round 5
n-bit
n-bit
n-bit
n-bit
n-bit
n-bit

FIGURE 1. The chromosome structure used to find as 5-round characteristic.

Let the chromosome a be represented as $(g_{a,1}, g_{a,2}, \dots, g_{a,k+1})$. The k-round differential characteristic corresponding to the chromosome a consists of k 1-round differential characteristic:

Round 1 differential characteristic: $g_{a,1} \longrightarrow g_{a,2}$
Round 2 differential characteristic: $\theta_L(g_{a,2}) \longrightarrow g_{a,3}$
... ...
Round k differential characteristic: $\theta_L(g_{a,k}) \longrightarrow g_{a,k+1}$

If we assume that substitution functions of size m are used in the round function of the cipher, each n-bit gene can be regarded as l substrings in which the substring $j (1 \leq j \leq l)$ has the size of m bits and corresponds to the input or output difference of the jth substitution function in the round function. Hence, the gene i in the chromosome a can be represented as

$$\begin{aligned} & g_{a,i} = (g^1_{a,i}, g^2_{a,i}, \dots, g^l_{a,i}) \\ & \text{where} \\ & |g_{a,i}| = n, \\ & |g^j_{a,i}| = m, \quad 1 \leq j \leq l \end{aligned} \tag{1}$$

4.2. Initial Population

To generate the initial population, the gene corresponding to the output difference of some round is considered as the *fixed gene*. We call the genes that are placed after the fixed gene as *forward genes* and the genes that are placed before the fixed gene as *backward genes*.

The position and value of the fixed gene, which are determined by cryptanalyzer, are fixed for all chromosomes. Then, the initial values of other genes in each chromosome are determined using a forward-backward procedure:

Let x_f be the position of the fixed gene and the chromosome a be represented as $(g_{a,1}, \dots, g_{a,x_f}, \dots, g_{a,k+1})$. The forward genes $g_{a,i}, i = x_f + 1$ to $k + 1$, are initialized as the following: First, by applying the linear transformation to the gene $g_{a,i-1}$ the input difference of round $i - 1$ is calculated. Then, the forward gene $g_{a,i}$ (that corresponds to output difference of the round $i - 1$) is considered as l substrings of size m. Each substring corresponds to the output difference of a substitution function. Hence, its value is chosen randomly from the difference distribution table of the substitution function, based on its corresponding input difference, and proportional to its probability of occurrence.

To initialize the backward genes $g_{a,i}, i = x_f - 1$ down to 2, the gene $g_{a,i+1}$ (that corresponds to output difference of the round i) is considered as l substrings of size m. Each substring would be the output difference of a substitution function. Hence, the corresponding input difference of each substring is chosen randomly from the difference distribution table of the substitution function and proportional to its probability of occurrence. Then, by applying the inverse of the linear transformation to the outcome, the value of the gene $g_{a,i}$ is determined. To initialize the backward gene $g_{a,1}$, after the calculation of input difference of round 1, its value is given to this gene.

4.3. Fitness Function

To evaluate the fitness of chromosomes, we use three fitness functions of f, f_1, and f_2.

For each chromosome a, $f(a)$ represents the fitness of the chromosome itself, $f_1(a)$ represents the fitness of its forward genes and $f_2(a)$ represents the fitness of its backward genes. We call f as the *overall fitness function*, f_1 as the *forward fitness function* and f_2 as the *backward fitness function*.

Let the chromosome a be represented as $(g_{a,1}, \ldots, g_{a,x_f}, \ldots, g_{a,k+1})$. The overall fitness, the forward fitness and the backward fitness of chromosome a are respectively calculated as

$$f(a) = (\log_2 \Pr(g_{a,1} \longrightarrow g_{a,2}) * \prod_{i=2}^{k} \Pr(\theta_L(g_{a,i}) \longrightarrow g_{a,i+1}))^2 \tag{2}$$

$$f_1(a) = (\log_2 \prod_{i=x_f}^{k} \Pr(\theta_L(g_{a,i}) \longrightarrow g_{a,i+1}))^2 \tag{3}$$

$$f_2(a) = (\log_2 \Pr(g_{a,1} \longrightarrow g_{a,2}) * \prod_{i=2}^{x_f-1} \Pr(\theta_L(g_{a,i}) \longrightarrow g_{a,i+1}))^2 \tag{4}$$

where

$$\Pr(\Delta I \longrightarrow \Delta O) = \prod_{j=1}^{l} \Pr(\Delta I^j \longrightarrow \Delta O^j)$$

In the above equations, θ_L is the linear transformation, $\Delta I \longrightarrow \Delta O$ denotes the differential characteristic of each round, $\Pr(\Delta I \longrightarrow \Delta O)$ is differential characteristic probability of that specific round, and $\Delta I^j \longrightarrow \Delta O^j$ shows the differential characteristic of the jth substitution function in that round.

According to Equation (2), the chromosome a is fitter than the chromosome b if $f(a) < f(b)$. Also, according to Equations (3) and (4), the chromosome a

has better forward or backward fitness than the chromosome b if $f_1(a) < f_1(b)$ or $f_2(a) < f_2(b)$.

4.4. Selection

To select two chromosomes for reproduction, we use the Fitness-Proportionate selection. One chromosome is chosen according to forward fitness f_1, and other is chosen according to backward fitness f_2.

We also use a mild form of Elitism selection: chromosome with the best overall fitness, chromosome with the best forward fitness, and chromosome with the best backward fitness is carried onto the next generation.

4.5. Crossover

Regarding the position and value of one of the genes is assumed to be fixed for all chromosomes during the generation of the initial population, the fixed-point crossover operator is used to perform crossover. In this operator, the fixed gene's boundary is considered as the *crossover fixed point*. Then, all genes to the right hand of this position in the parent chromosomes are exchanged to produce the offspring chromosomes.

The crossover point, denoted as x_c, remains fixed during the optimization process. If the values of parent chromosomes are:

$$(g_{a,1}, g_{a,2}, \cdots, g_{a,x_c}, g_{a,x_c+1}, \cdots, g_{a,k+1})$$
$$(g_{b,1}, g_{b,2}, \cdots, g_{b,x_c}, g_{b,x_c+1}, \cdots, g_{b,k+1})$$

Two offspring chromosomes will result as the following:

$$(g_{a,1}, g_{a,2}, \cdots, g_{a,x_c}, g_{b,x_c+1}, \cdots, g_{b,k+1})$$
$$(g_{b,1}, g_{b,2}, \cdots, g_{b,x_c}, g_{a,x_c+1}, \cdots, g_{a,k+1})$$

4.6. Heuristic Mutation

To perform the heuristic mutation on a chromosome, first the mutation point (x_m) is randomly selected. If $x_f < x_m \leq k+1$, then the values of gene g_{x_m} and its following genes will change. The values of these genes are determined similar to the method discussed in Section 4.2 for initializing the forward genes. If $1 \leq x_m < x_f$, then the values of gene g_{x_m} and its prior genes will change. The values of these genes are determined similar to the method discussed for initializing the backward genes. It should be mentioned that if $x_m = x_f$, then the heuristic mutation will not happen.

5. Experimental Results

We applied GenSPN to Serpent. In order to find a k-round characteristic for Serpent, each chromosome consists of $k+1$ 128-bit genes.

Using GenSPN, a highly probable 5-round and a highly probable 6-round differential characteristic of Serpent were found. The characteristics are related to rounds 1 to 5 and 1 to 6, respectively. We performed 10 runs of GenSPN with different random seeds and reported the best results obtained from these 10 runs.

Table 1 shows the parameter settings of GenSPN. Each run of GenSPN for finding 5-round and 6-round differential characteristics was performed in the average time of 13.8 and 137.4 seconds, respectively. We conducted the experiments on a Pentium 4/3.2GHZ/1GB RAM running Windows XP.

Table 1. The parameter settings of GenSPN.

Number of Rounds	**Number of Generations**	**Population Size**	**Crossover Rate**	**Mutation Rate**
5	100	200	0.95	0.20
6	500	400	0.95	0.40

Tables 2 and 3 show the best results obtained. The probability of the found 5-round differential characteristic for rounds 1 to 5 is 2^{-65} and the probability of the found 6-round differential characteristic for rounds 1 to 6 is 2^{-95}. These characteristics can be used for differential cryptanalysis of the 6 and 7-round Serpent. These differential characteristics have better probabilities than previously reported differential characteristics with probabilities of 2^{-80} [7] and 2^{-97} [9].

Figures 2 and 3 show the best overall fitness function at each generation versus the generation number of the best run of GenSPN.

Table 2. The found 5-round differential characteristic of Serpent, rounds 1 to 5.

Round	*S*-box	Input Difference of *S*-box	Output Difference of *S*-box	Prob.
1	S_1	000ED1409D104000E000000000000000	00082A6042A0C0008000000000000000	2^{-20}
2	S_2	00AA0A000A0000000000000000000000	00A20400080000000000000000000000	2^{-11}
3	S_3	00A00000000000000000000000000000	00200000000000000000000000000000	2^{-3}
4	S_4	02000000000000000000000000001000	0600000000000000000000000000A000	2^{-5}
5	S_5	64000000400000001010081810030244	690000009000000060600C6C60080D99	2^{-26}
			Total Probability	2^{-65}

TABLE 3. The found 6-round differential characteristic of Serpent, rounds 1 to 6.

Round	*S*-box	Input Difference of *S*-box	Output Difference of *S*-box	Prob.
1	S_1	000090040D8104DD0E00000000AA0400	0000400602FA06220800000000330600	2^{-25}
2	S_2	A00B000430E000600000000000A00C00	A004000A204000800000000000A00400	2^{-21}
3	S_3	0000400A000000000000000000000000	0000A004000000000000000000000000	2^{-6}
4	S_4	00000000000000400000000000000000	00000000000000300000000000000000	2^{-2}
5	S_5	20000000100002004000000000011000	A000000030000B00500000000006A000	2^{-17}
6	S_6	00700002004000001010080600430154	0020000700A00000E0E00B0100A20E3A	2^{-24}
			Total Probability	2^{-95}

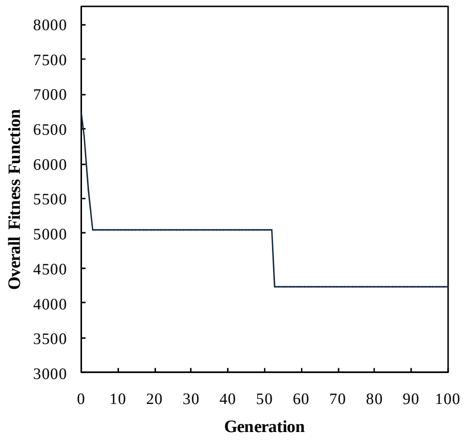

FIGURE 2. The progress of the best overall fitness for 5-round characteristic of Serpent, rounds 1 to 5.

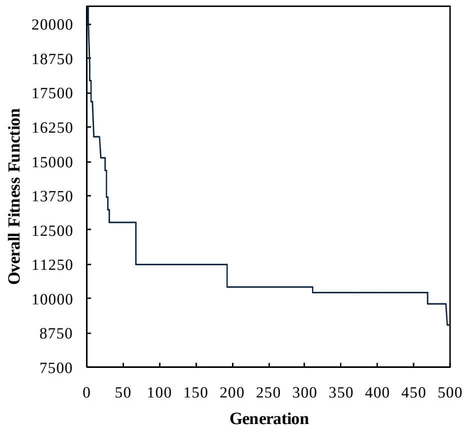

FIGURE 3. The progress of the best overall fitness for 6-round characteristic of Serpent, rounds 1 to 6.

To see how some parameters can affect the performance of GenSPN, we performed some additional experiments for different settings of Population Size and Mutation Rate parameters.

Since approaches based on genetic algorithms are stochastic in nature, we do not obtain the same result every time we run the algorithm. For this reason, we performed 10 runs for each setting of the algorithm. Figures 4 and 5 show the average of the best overall fitness function at each generation of these 10 runs versus the generation number for different settings of Population Size parameter. The settings of other parameters are the same as shown in Table 1. The figures suggest that by increasing the value of Population Size parameter, the average of the best overall fitness function will decrease. Therefore, we will obtain differential characteristics of higher probabilities.

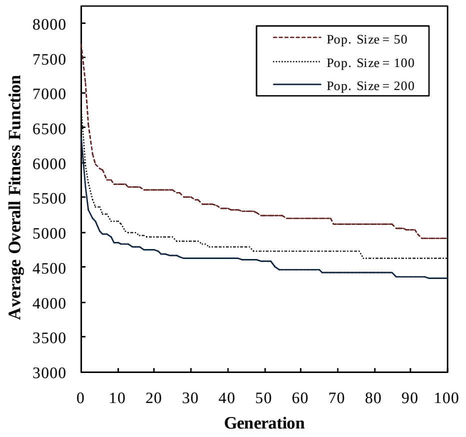

FIGURE 4. The progress of the average of the best overall fitness of 10 runs for 5-round characteristic of Serpent, rounds 1 to 5.

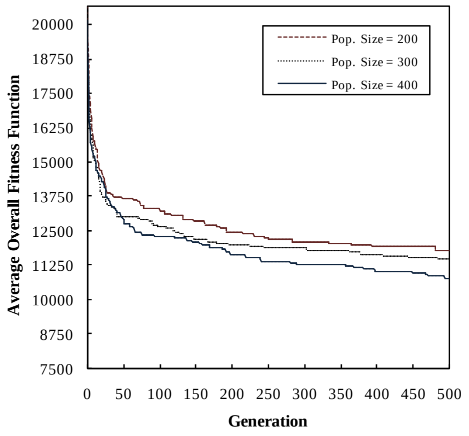

FIGURE 5. The progress of the average of the best overall fitness of 10 runs for 6-round characteristic of Serpent, rounds 1 to 6.

Figures 6 and 7 show the average of the best overall fitness function at each generation of the 10 runs versus the generation number for different settings of Mutation Rate parameter.

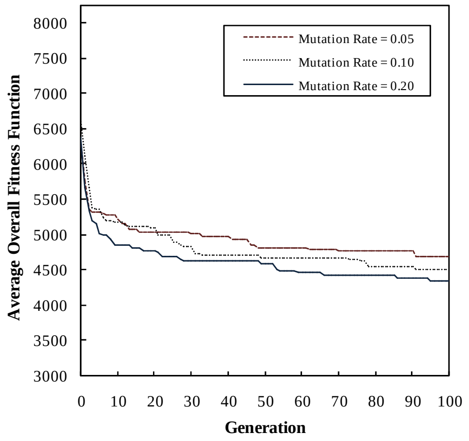

FIGURE 6. The progress of the average of the best overall fitness of 10 runs for 5-round characteristic of Serpent, rounds 1 to 5.

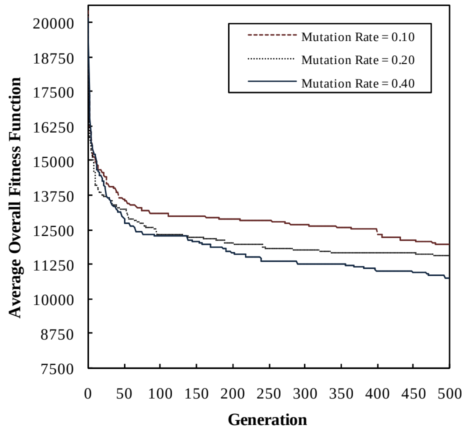

FIGURE 7. The progress of the average of the best overall fitness of 10 runs for 6-round characteristic of Serpent, rounds 1 to 6.

6. CONCLUSIONS

In this paper, we proposed a genetic algorithm, GenSPN, for finding highly probable differential characteristics of SPNs. We also applied the proposed genetic algorithm to find highly probable differential characteristics of Serpent. A 5-round and a 6-round differential characteristic for the rounds 1 to 5 and 1 to 6 were found in a relatively short time. These characteristics can be used for differential cryptanalysis of the 6 and 7-round Serpent. These differential characteristics have better probabilities of 2^{-65} and 2^{-95} than previously reported differential characteristics with probabilities of 2^{-80} [7] and 2^{-97} [9]. So far, no differential characteristic of Serpent with more rounds is reported.

The results of our experiments show that GenSPN can effectively find highly probable differential characteristics of SPNs.

References

[1] R. Anderson, E. Biham and L. Knudsen, *Serpent: A proposal for the advanced encryption standard,* NIST AES Proposal, 1998.

[2] E. Biham and A. Shamir, *Differential cryptanalysis of the data encryption standard*, Springer Verlag, Berlin, 1993.

[3] A. Ghaemi and B. Sadeghiyan, A differential operation presentation model of substitution-permutation block ciphers, *Amirkabir Scientific and Research Journal*, **58** no. 1 (2003), 352-369.

[4] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, Reading, MA, 1989.

[5] H. M. Heys, A tutorial on linear and differential cryptanalysis, *Cryptologia*, **XXVI** no. 3 (2002), 189-221.

[6] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*, Chapman and Hall/CRC Press, Boca Raton, FL, 2007.

[7] T. Kohono, J. Kelsey and B. Schneier, Preliminary cryptanalysis of reduced-round serpent, In *Proceedings of the 3rd AES Candidate Conference*, New York, NY, April (2000), 195-214.

[8] M. Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, MA, 1999.

[9] X. Y. Wang, L. C. K. Hui, K. P. Chow, C. F. Chong, W. W. Tsang and H. W. Chan, *The differential cryptanalysis of an AES finalist-Serpent*, Technical Report, Department of Mathematics, Shandong University, 2000.