Available online at www.sciencedirect.com
ScienceDirect
Applied Mathematics and Computation 184 (2007) 63–72
APPLIED MATHEMATICS AND COMPUTATION
www.elsevier.com/locate/amc

Applying evolutionary computation methods for the cryptanalysis of Feistel ciphers

E.C. Laskari [a,d,*], G.C. Meletiou [b,d], Y.C. Stamatiou [c,d], M.N. Vrahatis [a,d]

[a] Computational Intelligence Laboratory, Department of Mathematics, University of Patras, GR–26110 Patras, Greece
[b] A.T.E.I. of Epirus, P.O. Box 110, GR–47100 Arta, Greece
[c] Department of Mathematics, University of Ioannina, GR–45110 Ioannina, Greece
[d] University of Patras Artificial Intelligence Research Center (UPAIRC), University of Patras, GR–26110 Patras, Greece

Abstract

In this contribution instances of a problem introduced by the differential cryptanalysis of Feistel cryptosystems are formulated as optimization tasks. The performance of Evolutionary Computation methods on these tasks is studied for a representative Feistel cryptosystem, the Data Encryption Standard. The results indicate that the proposed methodology is efficient in handling this type of problems and furthermore, that its effectiveness depends mainly on the construction of the objective function. This approach is applicable to all Feistel cryptosystems that are amenable to differential cryptanalysis.

Keywords: Computational intelligence; Evolutionary computation; DES; Differential cryptanalysis; Feistel ciphers

1. Introduction

A *Feistel cipher* is a cryptosystem based on a sequential r times repetition of a function, called the *round function*, that maps an n-bit plaintext P, to a ciphertext C. In a Feistel cipher the current n-bit word is divided into $(n/2)$-bit parts, the left part L_i and the right part R_i [1]. Then round i, $1 \leqslant i \leqslant r$, has the following effect:

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

where K_i is the subkey used in the ith round (derived from the cipher key K), and F_i is an arbitrary round function for the ith round. After the last round function has been applied, the two halves are swapped and the outcome is the ciphertext C of the Feistel cipher, i.e. $C = (R_r, L_r)$. On Feistel based cryptosystems the decryption function is simply derived from the encryption function by applying the subkeys, K_i, and the round

* Corresponding author. Address: Computational Intelligence Laboratory, Department of Mathematics, University of Patras, GR–26110 Patras, Greece.
E-mail addresses: elena@math.upatras.gr (E.C. Laskari), gmelet@teiep.gr (G.C. Meletiou), istamat@cc.uoi.gr (Y.C. Stamatiou), vrahatis@math.upatras.gr (M.N. Vrahatis).

0096-3003/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2005.11.176

functions, F_i, in reverse order, rendering thus the Feistel structure an attractive choice for both software and hardware implementations.

The most widely used Feistel block-cipher cryptosystem is the Data Encryption Standard (DES) [2]. DES is a *symmetric cryptosystem*, meaning that the parties exchanging information possess the same key. It processes plaintext blocks of $n = 64$ bits, producing 64-bit ciphertext blocks, with effective key size $k = 64$ bits, eight of which can be used as parity bits. The main part of the round function is the F function, which works on the right half of the data, using a subkey of 48 bits and eight S-boxes. S-*boxes* are mappings that transform 6 bits to 4 bits in a nonlinear manner and constitute the only nonlinear component of DES. The 32 output bits of the F function are XORed with the left half of the data and the two halves are subsequently exchanged. A detailed description of the DES algorithm can be found in [3,4].

Two of the most powerful cryptanalytic attacks for Feistel based ciphers, rely on the exploitation of specific weaknesses of the S-boxes of the target cryptoalgorithm. These attacks are the *Linear Cryptanalysis* (see [5,6]) and the *Differential Cryptanalysis* (see [7,8]), and were first successfully applied to the cryptanalysis of DES.

In this contribution we consider different instances of a problem introduced by the Differential Cryptanalysis of a Feistel cryptosystem and formulate them as optimization tasks. In particular, we investigate the problem of finding some missing bits of the key that is used in a simple Feistel cipher, namely the Data Encryption Standard with four and six rounds, respectively. The performance of two Evolutionary Computation methods, the Particle Swarm Optimization (PSO) and the Differential Evolution algorithm (DE), on this problem is studied.

Evolutionary Computation optimization methods are inspired by paradigms of social behavior and/or natural evolution. A common characteristic of all these algorithms is that they do not employ information from the derivatives of the objective function. Therefore, they are applicable to hard real-world optimization problems that involve discontinuous objective functions and/or disjoint search spaces [9–11]. The performance of PSO in handling discrete optimization problems, through the technique of rounding off the real values of the solution to the nearest integer [12], was investigated in [13] with promising results.

In the experimental results for DES reduced to four rounds, the optimization methods considered, located the solution efficiently, in the sense that they required a smaller number of function evaluations compared to the brute force approach. For DES reduced to six rounds the effectiveness of the proposed algorithms depends on the construction of the objective function.

The rest of the paper is organized as follows. In Section 2 the considered problem is analyzed and two different instances of it are formulated as optimization tasks. In Section 3 the considered evolutionary computation methods are briefly described. In Section 4 the experimental setup and results are reported. The discussion of the reported results is given in Section 5. Section 6 summarizes with conclusions and future work directions.

2. Problem formulation

To describe the problem at hand, we briefly review the basic notions of differential cryptanalysis.

2.1. Basic notions of differential cryptanalysis

Differential Cryptanalysis (DC) is a chosen plaintext attack. It analyzes the effect of particular differences in plaintext pairs on the differences of the resultant ciphertext pairs. These differences can be used to assign probabilities to the possible keys and to identify the most probable key. This method usually works on a number of pairs of plaintexts with the same particular difference using only the resulting ciphertext pairs. For cryptosystems similar to DES, the difference is chosen as a fixed XORed value of the two plaintexts.

In order to locate the most probable key, DC uses *characteristics*. As defined in [7], an *r-round characteristic* is a tuple $\Omega = (\Omega_P, \Omega_\Lambda, \Omega_C)$ where Ω_P and Ω_C are n bit numbers and Ω_Λ is a list of r elements $\Omega_\Lambda = (\Lambda_1, \Lambda_2, \ldots, \Lambda_r)$, each of which is a pair of the form $\Lambda_i = (\lambda_I^i, \lambda_O^i)$, where λ_I^i and λ_O^i are $n/2$ bit numbers and n is the block size of the cryptosystem. A characteristic satisfies the following requirements: (a) λ_I^1 is the right half of Ω_P, (b) λ_I^2 is the left half of $\Omega_P \oplus \lambda_O^1$, (c) λ_I^r is the right half of Ω_C, (d) λ_I^{r-1} is the left half of $\Omega_C \oplus \lambda_O^r$, and (e) for every i, $2 \leqslant i \leqslant r-1$, $\lambda_O^i = \lambda_I^{i-1} \oplus \lambda_I^{i+1}$. Each characteristic has a probability, which

is the probability that a random pair with the chosen plaintext XOR, Ω_P, has the round XORs, Λ_i, and the ciphertext XOR, Ω_C, specified in the characteristic. Each characteristic allows the search for a particular set of bits in the subkey of the last round: the bits that enter some particular S-boxes, depending on the chosen characteristic. The characteristics that are most useful are those that have a maximal probability and a maximal number of subkey bits whose occurrences can be counted.

DC is a statistical method that fails in rare instances. A more extended analysis on DC and its results on DES for different numbers of rounds can be found in [7].

2.2. Problem formulation

2.2.1. DES reduced to four rounds

For DES reduced to four rounds, DC uses a one–round characteristic with probability 1 and provides at the first step of the cryptanalysis 42 bits of the subkey of the last round. We consider the case where the subkeys are calculated with the DES key scheduling algorithm, thus the 42 bits given by DC are actual key bits of the key and there are 14 key bits still missing. A possible way for obtaining these key bits is to try all the 2^{14} possibilities in decrypting the given ciphertexts, using the resulting keys. The right key should satisfy the known plaintext XOR value for all the pairs that are used by DC. An alternative approach is to use a second characteristic that corresponds to the missing bits and attempt a more careful counting on the key bits of the last two rounds. The approach of brute force requires testing $2^{14} = 16384$ possible key bits, while the implementation of the second approach is more complicated.

Instead of using the aforementioned approaches to find the missing key bits, we formulate the problem of computing the missing bits as an integer optimization problem. Since the right key should satisfy the known plaintext XOR value for all the pairs that are used by DC, these ciphertexts can be used for the evaluation of possible solutions provided by optimization methods. Thus, let $\mathbf{X}$ be a 14–dimensional vector, where each of its components corresponds to one of the 14 unknown key bits. Such a vector represents a possible solution of the problem. Also, let np be the number of ciphertext pairs used by DC to obtain the right 42 key bits. Construct the 56 bits of the key, using the 42 bits which are known by DC and the 14 components of $\mathbf{X}$ in the proper order. With the resulting key, decrypt the np ciphertext pairs and count the number of decrypted pairs that satisfy the known plaintext XOR value, denoted as $cnp_{\mathbf{X}}$. Thus, the objective function f, is the difference between the desired output np and the actual output $cnp_{\mathbf{X}}$, i.e. $f(\mathbf{X}) = np - cnp_{\mathbf{X}}$. The global minimum of the function f is zero and the global minimizer is with high probability the actual key. A first study of this approach is given in [14].

In this paper we provide a more detailed study of the performance of Evolutionary Computation methods on this problem, and also extend the proposed methodology in a different instance of the problem, namely the missing bits of the six round DES. The two instances are complementary, since every problem of missing bits of keys in Differential Cryptanalysis of Feistel ciphers can be categorized into one of the two cases.

2.2.2. DES reduced to six rounds

The cryptanalysis of DES reduced to six rounds is more complicated than the four round version, since the best characteristic that can be used has probability less than 1. In particular, DC uses two characteristics of probability $p_{sr} = 1/16$ to provide 42 bits of the right key. Again, there are 14 bits of the key missing. However, in this case the right key may not be suggested by all ciphertext pairs. This holds because not all the corresponding plaintexts pairs are *right pairs*. A pair is called *right pair with respect to an r-round characteristic* $\Omega = (\Omega_P, \Omega_\Lambda, \Omega_C)$ and an independent key K, if it holds that $P' = \Omega_P$, where P' is the pair's XOR value, and for the first r rounds of the encryption of the pair using the independent key K the input and output XORs of the ith round are equal to λ_I^i and λ_O^i, respectively [7].

The probability that a pair, with plaintext XOR equal to Ω_P of the characteristic, is a right pair using a fixed key is approximately equal to the probability of the characteristic. A pair which is not a right pair is called *wrong pair* and it does not necessarily suggest the right key as a possible value. The study of right and wrong pairs, has shown that the right key appears with the probability of the characteristic from the right pairs and some other random occurrences from wrong pairs. In conclusion, if all the pairs of DC (right and wrong) are used in the predefined objective function f, the function's minimum value will change depending on the specific

pairs used. On the other hand, if the right pairs are filtered and are solely used in the objective function f, the function's global minimum will be constant with value equal to 0, as in the case of missing bits of DES reduced to four rounds. As the filtering of the right pairs is not always possible and easy, we study the behavior of the proposed approach using the objective function f with all the pairs of DC.

3. The evolutionary computation methods considered

For completeness purposes, in this section we briefly describe the considered Evolutionary Computation methods, namely the Particle Swarm Optimization and the Differential Evolution algorithm. Both methods employ a population of possible solutions to identify promising regions of the search space. In the context of PSO, the population is called *swarm*, while the members of the population are called *particles*. Each particle moves in the search space according to three quantities; its velocity, the memory of its own best, previous, position and the memory of the best, previous, position of its *neighborhood*. In the *global* variant of the method the whole swarm is considered as one neighborhood. Thus, the best position of the swarm is communicated to all the particles. On the contrary, in the *local* variant of the method a prespecified number of particles constitute the particle's neighborhood and the best position of these specific particles is communicated among them [15].

Let $\mathbf{S}$ be a D-dimensional search space, $\mathbf{S} \subset \mathbb{R}^D$, and a swarm of N particles. Then, in the *constriction factor* version of PSO the following equations are used to manipulate the swarm [16]:

$$V_i^{(t+1)} = \chi\left(V_i^{(t)} + c_1 r_1\left(P_i^{(t)} - X_i^{(t)}\right) + c_2 r_2\left(P_g^{(t)} - X_i^{(t)}\right)\right), \tag{1}$$

$$X_i^{(t+1)} = X_i^{(t)} + V_i^{(t+1)}, \tag{2}$$

where $i = 1, 2, \ldots, N$; χ is the constriction factor; c_1 and c_2 denote the *cognitive* and *social* parameters, respectively; r_1, r_2 are random numbers uniformly distributed in the range $[0, 1]$; and t, stands for the counter of iterations. By Eq. (1) the velocity vector of the ith particle, $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})^\top$, is the result of the weighted summation of the particle's current velocity, and the weighted differences of the particle's current position, $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})^\top$, with its best previous position, $P_i = (p_{i1}, p_{i2}, \ldots, p_{iD})^\top$, and with the best position encountered by the particles comprising its neighborhood, $P_g = (p_{g1}, p_{g2}, \ldots, p_{gD})^\top$. The position of the ith particle in the next iteration is obtained by simply adding the velocity to the current position (Eq. (2)). All computations in Eqs. (1) and (2) are component wise.

The value of the constriction factor is typically obtained through the formula

$$\chi = 2\kappa / \left|2 - \phi - \sqrt{\phi^2 - 4\phi}\right|, \quad \text{for } \phi > 4,$$

where $\phi = c_1 + c_2$, and $\kappa = 1$. Different configurations of χ, as well as, a theoretical analysis of the derivation of the above formula can be found in [16].

The Differential Evolution algorithm [17] is a parallel direct numerical search method, that utilizes N, D–dimensional parameter vectors $x_{i,G}$, $i = 1, \ldots, N$, as a population for each iteration (generation) of the algorithm. At each generation, the *mutation* and *crossover* (*recombination*) operators are applied on the individuals, to produce a new population, which is subsequently subjected to the selection phase.

For each vector $x_{i,G}$, $i = 1, \ldots, N$, a *mutant vector* is generated through the following equation:

$$v_{i,G+1} = x_{r_1,G} + F(x_{r_2,G} - x_{r_3,G}), \tag{3}$$

where $r_1, r_2, r_3 \in \{1, \ldots, N\}$, are mutually different random indexes, and $F \in (0, 2]$. The indexes r_1, r_2, r_3, also need to differ from the current index, i. Consequently, to apply mutation, N must be greater than, or equal to, 4.

Following the mutation phase, the crossover operator is applied on the mutant vector yielding the *trial vector*, $u_{i,G+1} = (u_{1i,G+1}, u_{2i,G+1}, \ldots, u_{Di,G+1})$, where,

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1}, & \text{if } (\text{randb}(j) \leqslant CR) \text{ or } j = \text{rnbr}(i), \\ x_{ji,G}, & \text{if } (\text{randb}(j) > CR) \text{ and } j \neq \text{rnbr}(i), \end{cases}$$

for $j = 1, 2, \ldots, D$; where randb(j), is the jth evaluation of a uniform random number generator in the range $[0, 1]$; CR is the (user specified) crossover constant in the range $[0, 1]$; and rnbr(i) is a randomly chosen index from the set $\{1, 2, \ldots, D\}$.

To decide whether or not the vector $u_{i,G+1}$ will be a member of the population of the next generation, it is compared to the initial vector $x_{i,G}$. Thus,

$$x_{i,G+1} = \begin{cases} u_{i,G+1}, & \text{if } f(u_{i,G+1}) < f(x_{i,G}), \\ x_{i,G}, & \text{otherwise.} \end{cases}$$

The procedure described above is the standard variant of the DE algorithm. Different mutation operators that have been applied with promising results [17], are the following:

$$v_{i,G+1} = x_{\text{best},G} + F(x_{r_1,G} - x_{r_2,G}), \tag{4}$$

$$v_{i,G+1} = x_{i,G} + F(x_{\text{best},G} - x_{i,G}) + F(x_{r_1,G} - x_{r_2,G}), \tag{5}$$

$$v_{i,G+1} = x_{\text{best},G} + F(x_{r_1,G} + x_{r_2,G} - x_{r_3,G} - x_{r_4,G}), \tag{6}$$

$$v_{i,G+1} = x_{r_1,G} + F(x_{r_2,G} + x_{r_3,G} - x_{r_4,G} - x_{r_5,G}), \tag{7}$$

where, $x_{best,G}$, corresponds to the best individual of the Gth generation, $r_1, r_2, r_3, r_4, r_5 \in \{1, \ldots, N\}$, are mutually different random indexes and $x_{i,G}$ is the current individual of generation G.

4. Experimental setup and results

Both PSO and DE methods were applied considering each component of the possible solution as a real number in the range $[0, 1]$ and all populations were constrained in the feasible region of the problem. For the evaluation of the suggested solutions, the technique of rounding off the real values of the solution to the nearest integer [12] was applied. For the PSO method we have considered both the global and local variants, and for the DE algorithm all five variants described in Section 3. A maximum value for the velocity, $V_{\max} = 0.5$, of the PSO method was set in order to avoid the swarm's explosion, i.e. avoid velocities from assuming large values that lead the particles to the boundary of the search space, and thus destroy the dynamic of the method. The parameters of PSO were set at the default values, i.e. $\chi = 0.729$ and $c_1 = c_2 = 2.05$, found in the literature [16], and the parameters of DE were set at equal values $CR = F = 0.5$.

The proposed approach was tested for several different initial keys and number of pairs, np. For each setting, the size of each population was equal to 100 and the performance of the methods was investigated on 100 independent runs. A run is considered to be successful if the algorithm identifies the global minimizer within a prespecified number of function evaluations. The function evaluations threshold for both problems was taken equal to 2^{14}. For the missing bits of the key of DES reduced to four rounds, the results for six different keys, k_i, $i = 1, \ldots, 6$, and for test pairs, np, equal to 20 and 50 are reported in Tables 1 and 2, respectively.

Concerning the notation used in the tables, PSOCG is the global variant of the constriction factor version of PSO, PSOCL1 is PSO's local variant with neighborhood size equal to 1, PSOCL2 is PSO's local variant with neighborhood size equal to 2, and DE1, DE2, DE3, DE4, DE5 denote the five DE variants of Eqs. (3)–(7), respectively. Each table reports the success rate of each algorithm, that is the proportion of the times it achieved the global minimizer within the prespecified threshold, and the mean value of function evaluations over the successful experiments.

For the missing bits of the key of DES reduced to six rounds, where both right and wrong pairs are used for the construction of the objective function, the results for the same six different keys tested for DES reduced to four rounds, and for test pairs, np, equal to 200, are reported in Table 3.

5. Discussion

The results for the problem of missing bits of DES reduced to four rounds suggest that the proposed approach is able to locate the global minimizer, i.e. the 14 missing bits of the key, with relatively low computational cost compared to the brute force attack. The success rates of all versions of the two methods are high. For np equal to 20 success rates range from 93% to 100%, with an average of 99.3%. For np equal to 50 the

Table 1
Results for six different keys using $np = 20$ test pairs for DES reduced to four rounds

Key	Method	Success Rate (%)	Mean F.E.
k_1	PSOGC	99	742.42
	PSOLC1	100	1773.00
	PSOLC2	100	1255.00
	DE1	100	614.00
	DE2	100	1406.00
	DE3	100	780.00
	DE4	100	588.00
	DE5	100	1425.00
k_2	PSOGC	99	911.11
	PSOLC1	100	2665.00
	PSOLC2	100	1650.00
	DE1	100	603.00
	DE2	100	1518.00
	DE3	100	879.00
	DE4	100	615.00
	DE5	100	1649.00
k_3	PSOGC	94	1117.02
	PSOLC1	99	2447.48
	PSOLC2	100	1688.00
	DE1	99	693.94
	DE2	100	1497.00
	DE3	100	805.00
	DE4	100	690.00
	DE5	100	1427.00
k_4	PSOGC	96	876.04
	PSOLC1	100	2089.00
	PSOLC2	100	1418.00
	DE1	99	701.01
	DE2	100	1378.00
	DE3	100	843.00
	DE4	100	568.00
	DE5	100	1362.00
k_5	PSOGC	97	900.00
	PSOLC1	99	1979.80
	PSOLC2	100	1496.00
	DE1	100	662.00
	DE2	100	1493.00
	DE3	100	848.00
	DE4	100	662.00
	DE5	100	1542.00
k_6	PSOGC	93	1457.00
	PSOLC1	95	4475.79
	PSOLC2	99	2913.13
	DE1	100	651.00
	DE2	100	1717.00
	DE3	100	1063.00
	DE4	99	725.25
	DE5	100	1583.00

success rates lie in the region from 90% to 100%, with mean 99.4%. The improvement of the success rate as the number of ciphertext pairs, *np*, increases is expected as the larger number of ciphertext pairs used for the evaluation of the possible solutions reduces the possibility of a wrong 14-tuple to be suggested as the right one. The mean number of function evaluations required to locate the global minimizer (over all variants) is

Table 2
Results for six different keys using $np = 50$ test pairs for DES reduced to four rounds

Key	Method	Success Rate (%)	Mean F.E.
k_1	PSOGC	99	860.61
	PSOLC1	100	1698.00
	PSOLC2	100	1141.00
	DE1	100	485.00
	DE2	100	1215.00
	DE3	100	785.00
	DE4	100	553.00
	DE5	100	1382.00
k_2	PSOGC	94	741.49
	PSOLC1	100	1367.00
	PSOLC2	100	1100.00
	DE1	99	490.91
	DE2	100	1081.00
	DE3	100	669.00
	DE4	100	521.00
	DE5	100	1128.00
k_3	PSOGC	99	631.31
	PSOLC1	100	1217.00
	PSOLC2	100	1035.00
	DE1	100	385.00
	DE2	100	1006.00
	DE3	100	546.00
	DE4	100	409.00
	DE5	100	1016.00
k_4	PSOGC	90	947.78
	PSOLC1	98	2292.88
	PSOLC2	100	1588.00
	DE1	98	666.33
	DE2	100	1342.00
	DE3	100	838.00
	DE4	99	649.50
	DE5	100	1294.00
k_5	PSOGC	100	707.00
	PSOLC1	100	1763.00
	PSOLC2	100	1193.00
	DE1	100	445.00
	DE2	100	1127.00
	DE3	100	684.00
	DE4	100	465.00
	DE5	100	1131.00
k_6	PSOGC	96	880.21
	PSOLC1	100	2009.00
	PSOLC2	100	1390.00
	DE1	100	507.00
	DE2	100	1250.00
	DE3	100	692.00
	DE4	100	563.00
	DE5	100	1230.00

1309 in the case of $np = 20$ and 982 in the case of $np = 50$. This implies that, as more ciphertext pairs are incorporated in the objective function, not only the evaluation becomes more accurate, but also the the global minimizer becomes easier to locate. However, the number of ciphertext pairs used by the proposed approach should not exceed the number of the ciphertext pairs that are used by DC for the initial problem, as this would increase the total cost of the cryptanalysis in terms of encryptions and decryptions.

Table 3
Results for six different keys using $np = 200$ test pairs for DES reduced to six rounds

Key	Method	Success Rate (%)	Mean F.E.
k_1	PSOGC	26	7038.46
	PSOLC1	9	2188.89
	PSOLC2	8	3862.50
	DE1	36	5191.67
	DE2	52	5515.39
	DE3	41	5807.32
	DE4	51	6364.71
	DE5	59	6855.93
k_2	PSOGC	24	5037.50
	PSOLC1	3	1500.00
	PSOLC2	7	2357.14
	DE1	34	6535.29
	DE2	58	6968.97
	DE3	40	5945.00
	DE4	39	6897.44
	DE5	61	6932.79
k_3	PSOGC	41	4902.44
	PSOLC1	6	4533.33
	PSOLC2	5	7340.00
	DE1	48	5070.83
	DE2	61	6967.21
	DE3	53	6698.11
	DE4	48	5889.58
	DE5	56	7926.79
k_4	PSOGC	47	4912.77
	PSOLC1	13	4407.69
	PSOLC2	23	4134.78
	DE1	57	6491.23
	DE2	76	7594.74
	DE3	66	6418.18
	DE4	72	5741.67
	DE5	76	7001.32
k_5	PSOGC	36	5575.00
	PSOLC1	4	1950.00
	PSOLC2	5	4700.00
	DE1	51	5688.24
	DE2	62	7803.23
	DE3	57	5229.83
	DE4	53	5377.36
	DE5	64	6387.50
k_6	PSOGC	37	5624.32
	PSOLC1	5	2920.00
	PSOLC2	9	3377.78
	DE1	49	5681.63
	DE2	63	7380.95
	DE3	50	7048.00
	DE4	51	5621.57
	DE5	64	7679.69

With respect to the different variants of the PSO method, the local variant with neighborhood size two accomplished success rates close to 100%, in all instances of the first problem, with an average of 1489 of function evaluations. The global variant of PSO achieved success rates from 93% to 100% in different instances of the problem, but with an average of 898 of function evaluations. This means that, although the global variant

of PSO exhibits overall lower success rates, in the cases where both local and global variants of PSO are able to locate the minimizer, the global variant requires less function evaluations than the local variant.

The DE variants exhibited a stable and similar behavior, with mean success rates 100% in all cases. A minor exception was DE4 that achieved a mean of success rates of 99% on two instances of the problem. DE1 required the lowest mean number of function evaluations (576) among all two methods and their variants.

Regarding the results of DES reduced to six rounds, we observe from Tables 1–3 that there is a considerable difference between the success rates for the case of four rounds and the case of six rounds. This can be attributed to the fact that in the former case we work with a characteristic that occurs with probability 1 while in the latter case we work with a characteristic with smaller probability (1/16). This means that in the set of 200 ciphertext pairs used by the objective function, approximately 12 pairs are right and suggest the right tuple, while the remaining 188 pairs suggest tuples at random, decreasing thus, the possibility of suggestion of the right tuple. Consequently, since the objective function becomes more effective when more right pairs are available or, equivalently, when the probability of the utilized characteristic is large, it is expected that in the four round case the performance of the methods should be better than in the six round case. Although, the wrong pairs used in the objective function of DES for six rounds are misleading for the evaluation of the right tuple of missing bits, the global variant of PSO and all DE variants were able to locate the missing bits on an average of 35% of independent runs for PSOGC and 55% for the DE variants over all six different keys tested. The function evaluations required for the location of the right 14–tuple of missing bits in this case are on average 5600 for all methods.

Finally, an interesting observation from the results of the proposed approach is that in the case of DES reduced to four rounds all methods in independent runs were able to locate 4 different 14–tuples satisfying the condition criterion of the objective function. These 4 solutions of the problem differed in two fixed positions, the 10th and the 36th, of the DES key. In the case of DES reduced to six rounds just one solution, the right one, was located by all methods.

6. Conclusions and future work

In this contribution the problem of finding missing key bits of a Feistel cipher, the DES reduced to four and six rounds, respectively, is formulated as an optimization task. The performance of two Evolutionary Computation methods in addressing the two different instances of the optimization problem is studied and results are reported.

The results indicate that the proposed methodology is efficient in handling this type of problems, since on DES reduced to four rounds it managed to address the problem at hand using an average of 576 function evaluations in contrast with the brute force approach that requires $2^{14} = 16384$ evaluations. Furthermore, the results of DES reduced to six rounds shows that the effectiveness of the proposed approach depends mainly on the construction of the objective function. Also, this approach is applicable to all Feistel cryptosystems that are amenable to differential cryptanalysis, motivating thus its use for other Feistel cryptosystems. Finally, as a future direction, we are interested in studying the effectiveness of the proposed approach not just for missing bits of the key produced by Differential Cryptanalysis but also for all the bits of the key of Feistel ciphers.

Acknowledgement

The authors would like to acknowledge the partial support by the ''Archimedes'' research programme awarded by the Greek Ministry of Education and Religious Affairs and the European Union.

References

[1] H. Feistel, Cryptography and computer privacy, Scientific American 228 (5) (1973) 15–23.
[2] National Bureau of Standards, US Department of Commerce, FIPS pub. 46, Data Encryption Standard, January 1977.
[3] A. Menezes, P. van Oorschot, S. Vanstone, Handbook of Applied Cryptography, CRC Press series on discrete mathematics and its applications, CRC Press, 1996.
[4] D. Stinson, Cryptography: Theory and Practice (Discrete Mathematics and Its Applications), CRC Press, 1995.
[5] M. Matsui, A. Yamagishi, A new method for known plaintext attack of FEAL cipher, Lecture Notes in Computer Science (1992) 81–91.

[6] M. Matsui, Linear cryptanalysis method for DES cipher, Lecture Notes in Computer Science 765 (1994) 386–397.
[7] E. Biham, A. Shamir, Differential cryptanalysis of DES-like cryptosystems, Journal of Cryptology 4 (1) (1991) 3–72.
[8] E. Biham, A. Shamir, Differential Cryptanalysis of the Data Encryption Standard, Springer-Verlag, 1993.
[9] D. Fogel, Evolutionary Computation: Towards a New Philosophy of Machine Intelligence, IEEE Press, Piscataway, NJ, 1995.
[10] J. Kennedy, R. Eberhart, Swarm Intelligence, Morgan Kaufman Publishers, 2001.
[11] H.-P. Schwefel, Evolution and Optimum Seeking, Wiley, New York, 1995.
[12] S. Rao, Engineering Optimization – Theory and Practice, Wiley Eastern, New Delhi, 1996.
[13] E. Laskari, K. Parsopoulos, M. Vrahatis, Particle swarm optimization for integer programming, in: Proceedings of the IEEE 2002 Congress on Evolutionary Computation, IEEE Press, Hawaii, HI, 2002, pp. 1576–1581.
[14] E.C. Laskari, G.C. Meletiou, Y.C. Stamatiou, M.N. Vrahatis, Evolutionary computation based cryptanalysis: A first study, Nonlinear Analysis: Theory, Methods and Applications 63 (2005) e823–e830.
[15] R. Eberhart, P. Simpson, R. Dobbins, Computational Intelligence PC Tools, Academic Press, 1996.
[16] M. Clerc, J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation 6 (1) (2002) 58–73.
[17] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (1997) 341–359.