A Genetic Algorithm for Cryptanalysis with Application to DES-like Systems

Hasan Mohammed Hasan Husei[1], Bayoumi I. Bayoumi[2],
Fathy Saad Holail[3], Bahaa Eldin M. Hasan[4], and Mohammed Z. Abd El-Mageed[5]
(Corresponding author: Hasan Mohammed Hasan Husei)

Research Development Center, National Defense Council, Cairo, Egypt.[1]
Department of Mathematics, Faculty of Science, Ain Shams University, Cairo, Egypt.[2]
Head of C. R. Division, Research Development Center, National Defense Council, Cairo, Egypt.[3]
C. R. Division, Research Development Center, National Defense Council, Cairo, Egypt.[4]
Department of Computer and System, Faculty of Engineering, Al-Azhar University, Cairo, Egypt.[5]
(Received Nov. 25, 2006; revised May 21, 2007; accepted June 12, 2007)

Abstract

Various cryptosystems use exhaustive techniques to search the key space. Such search techniques should be guided in order to be computationally adequate. Here, a Genetic Algorithm, GA, is proposed for the cryptanalysis of DES-like systems to find out the underlying key. The genetic algorithm approach is adopted, for obtaining the exact key by forming an initial population of keys that belong to the key subspace. In the proposed algorithms the premature convergence could be avoided by dynamic variation of control parameters that can affect the fitness function. In this paper a new method has been developed for the first time to break DES-like examples. These examples include both DES and FEAL with eight rounds. The performance of the propsed method, as such, is considerably faster than exhaustive search and differential cryptanalysis, DC. Therefore, it can be directly applied to a variety of DES-like systems instead of the current DC techniques.

Keywords: Data encryption standard, DES-like systems, differential cryptanalysis, fast encryption algorithm, genetic algorithm

1 Introduction

a random search through a finite but large key space is not usually an acceptable cryptanalysis approach. The focus of this work is on the use of a genetic algorithm (GA) to conduct a directed search in a key space. In fact GAs as an evolutionary optimization method [7] could be used with the advantages of:

1) Finding solutions for problems that are not analytical in nature.
2) Natural capability to solve combinatorial optimization problems.
3) Combining the exploitation of past results with the exploration of new locations of the search space.

They provide a guide search technique based on utilizing a fitness function that grows up with evolution of the solution. However, in DES-like ciphers nothing can be observed and subkeys are equally probable [3]. Thus, how the search is guided in such space? This paper presents, for the first time, a fitness function that could be successfully used to find out the underlying key.

The relative merits of the method proposed here are:

1) It outperforms the methods of simple random search and random walk. For DES-like cryptanalysis these methods always diverge and consequently a solution could be obtained only by impractical exhaustive search.
2) Also, Particle Swaron Optimization, PSO [12] and [14], cannot be used for seeking the key of DES-like cipher. The reasons are:
   a. PSO is cooperative, by nature, since a particle is moving in the key space. They communicate with their neighbors to exchange the best new information. Of coarse this is not the case of DES-like ciphers, however, PSO may be applicable for simple substitution cryptosystems.
   b. During a trip each particle places (lays) an amount of virtual pheromone trail. The change in the amount of each pheromone trail represents the change of the swarm information and reflects the experience acquired by particles during the cryptanalysis process. Again, this is not the case of DES-like ciphers.

Actually, in DES-like systems the key has no characteristics and it could not be built incrementally since a change in one key bit will load to totally different ciphertext. Thus neither the exploration provided by neighborhood nor the experience expressed by the amount of virtual pheromone trail can successfully guide the search to find out the key of DES-like systems.

3) Actually, GAs have been recently used in various applications of cryptography [4] and [13]. However, the algorithm proposed here out performs all GA methods that are used to solve simple tinge encryption problems [1]. The proposed algorithms exploit a novel fitness function that is effective, sensible and incorporates a deep analysis specific knowledge.

The ability to add direction to what seems to be a random search can be provided by genetic algorithms, which allow efficient search of a large key space. In what follows the proposed algorithms are applied to basic DES-like ciphers as a start for more complicated implementations.

In Section 2, a background of DES design, FEAL design, and summary of DC technique are given. In Section 3, a background of GA is discussed. In Section 4, new methods of using GA for analysis of DES-like systems DES-8 and FEAL-8 are explained. In Sections 5, 6, there are discussion implementation and conclusion.

2 Preliminaries

In an r-round iterated block cipher such as DES, the ciphertext is computed by iteratively applying a round function g to the plaintext such that

$$C_i = g(C_{i-1}, K_i), i = 1, 2, \ldots, r,$$

where C_0 is the plaintext, K_i is a round key and C_r is the ciphertext. The round function is usually based on using S boxes, arithmetic operations, and bitwise XORing [2].

A Feistel cipher with block length of $2n$ and r rounds is defined as follows. The round function is:

$$g : GF(2)^n \times GF(2)^n \times GF(2)^m \rightarrow GF(2)^n \times GF(2)^n$$
$$g : (X, Y, Z) = (Y, F(Y, Z) + X).$$

Thus, Given plaintext $P = (P^L, P^R)$ and r round keys $K_1, K_2, \ldots, K_r$, the ciphertext $C = (C^L, C^R)$ is computed in each round as follows:

1) Set $C_0^L = P^L, C_0^R = P^R$ and,
2) Compute $(C_i^L, C_i^R) = (C_{i-1}^R, F(C_{i-1}^R, K_i) + C_{i-1}^L)$ for $i = 1, 2, \ldots, r$,
3) Set, $C^L = C_r^L, C^R = C_r^R$, where the round key $K_i \in GF(2)^m$.

A DES-like system is a Feistel cipher, where F is defined as [8]:

- $F : GF(2)^m \rightarrow GF(2)^n$,
- $F(X, K_i) = P(f(E(X) \otimes K_i))$, where $K_i \in GF(2)^m$,
- $f : GF(2)^m \rightarrow GF(2)^n, m \geq n$ be a weak round function,
- $E : GF(2)^n \rightarrow GF(2)^m$ be an affine expansion mapping, and
- $P : GF(2)^n \rightarrow GF(2)^n$ be a permutation.

2.1 Background of DES Cipher

DES, as Feistel cipher, has had a greatest impact upon data security since 1977 in [10].

Algorithm DES-8 [Men 96]

Input: 64-bit plaintext block $M = m_1 \ldots m_{64}$; 64-bit key block $K = k_1 \ldots k_64$ (induced 8 parity bits)

Output: 64-bit ciphertext block $C = c_1 \ldots c_{64}$;

1) Compute eight 48-bit round subkeys K_i from K (key schedule).
2) Use the initial permutation IP to permute bits; and split the result into left and right 32-bit halves $L_0 = m_{58}m_{50} \ldots m_8$, $R_0 = m_{57}m_{49} \ldots m_7$, respectively;
3) For i from 1 to 8 compute $L_i = R_{i-1}, R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$, where $f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i))$;
   a. Expand $R_{i-1} = r_1 \ldots r_{32}$ from 32 to 48 bits $E(R_{i-1}) = r_{32}r_1 \ldots r_{32}r_1$;
   b. $E(R_{i-1}) \oplus K_i$ is eight 6-bit character string $B_1 \ldots B_8$;
   c. $S(E(R_{i-1}) \oplus K_i)$ is the 32-bit result substitution of eight S-boxes $S_1(B_1) \ldots S_8(B_8)$ each of which substitute 6 bits by 4 bits from the S-box tables;
   d. $P(S(E(R_{i-1}) \oplus K_i))$ permutes the 32 bits from permutation round;
4) (L_8, R_8) is the final block $b_1 \ldots b_{64}$;
5) $C = IP^{-1}(b_1 \ldots b_{64}) = b_{40}b_8 \ldots b_{25}$;

Algorithm DES-8 key schedule [Men 96]

Input: 64-bit key block $K = k_1 \ldots k_{64}$ (induced 8 parity bits)

Output: Eight 48-bit round subkeys K_i, $1 \leq i \leq 8$;

1) Define $v_i, 1 \leq i \leq 8$ as follows $v_i = 1$ if $i \in \{1, 2\} v_i = 2$ otherwise are the left-shift values for 28-bit circular rotations;
2) PC1(K) is the 28-bit halves $C_0 = k_{57}k_{49} \ldots k_{36}$, $D_0 = k_{63}k_{55} \ldots k_4$, respectively;

3) For i from 1 to 8 compute $C_i = v_i(C_{i-1}), D_i = v_i(D_{i-1})$ with left circular shift, and select 48 bits from the concatenation $b_1 \dots b_{56}$ of C_i and $D_i(PC2(C_i, D_i) = K_i = b_{14}b_{17} \dots b_{32}$.

In fact, the decryption process executes the encryption algorithm with the same key schedule using the order $k_8, k_7, \dots, k_1$, the effect of IP^{-1} is cancelled by IP in decryption, leaving (L_8, R_8).

2.2 Background of FEAL Cipher

To confirm the applicability of the GA to cryptanalysis of block cipher systems, FEAL is also examined. FEAL was designed, in the initial version with 4 rounds (FEAL-4) as a DES-like system, but with a far simpler f-function, that are augmented by initial and final stages. These stages XOR the two data halves, and as well as they XOR subkeys directly onto data halves. Within the f-function, two byte-oriented data substitution (S-boxes) S_0 and S_1 are each used twice, so that:

$$S_d : GF(2)^8 \times GF(2)^8 \rightarrow GF(2)^8, d \in \{0, 1\}.$$

S_0 and S_1 add a single bit d to 8-bit arguments x and y, ignore the carry out of the top bit, and left rotate the result 2 bits (ROL2).

$$S_d(x, y) = ROL2(x + y + d \bmod 256).$$

The key schedule uses a function f_k-function similar to f-function.

$$f : GF(2)^{32} \rightarrow GF(2)^{32}, f_k : GF(2)^{32} \rightarrow GF(2)^{32}.$$

Take A_i, B_i, Y_i, t_i and $U_i \in GF(2)^8, 0 \leq i \leq 3$, the output $\cup = (\cup_0, \cup_1, \cup_2, \cup_3)$ for FEAL functions f, f_k is defined as shown in the following table.

Table 1: FEAL functions $f, f - k$ and $\cup$ (the output) $= (\cup_0, \cup_1, \cup_2, \cup_3)$

$f(A, Y) \rightarrow \cup$	$f_k(A, B) \rightarrow \cup$	
$(A_0 \oplus A_1) \oplus Y_0$	$A_0 \oplus A_1$	$= t_1$
$(A_2 \oplus A_3) \oplus Y_1$	$A_2 \oplus A_3$	$= t_2$
$S_1(t_1, t_2)$	$S_1(t_1, t_2 \oplus B_0)$	$= \cup_1$
$S_0(t_2, \cup_1)$	$S_0(t_2, \cup_1 \oplus B_1)$	$= \cup_2$
$S_0(A_0, \cup_1)$	$S_0(A_0, \cup_1 \oplus B_2)$	$= \cup_0$
$S_1(A_3, \cup_2)$	$S_1(A_3, \cup_2 \oplus B_3)$	$= \cup_3$

Algorithm FEAL-8 [Men 96]

In the algorithm of FEAL-8, the f-function $f(A, Y)$ maps an input pair of 32-bits to a 32-bit output. Within the f function, two byte-oriented data substitutions (S-boxes). S_0 and S_1 are each used twice. Each maps a pair of 8-bit inputs to an 8-bit output (as in the Table). S_0 and S_1 add a single bit d (0 or 1) to 8-bit arguments x and y, ignore the carry out of the top bit, and left rotate the result 2 bits. This yields

(ROT2): $S_d(x; y) = ROT2(x + y + d \bmod 256)$.

Input: 64-bit plaintext $M = m_1, \dots, m_{64}$; 64-bit key $K = k_1, \dots, k_{64}$.

Output: 64-bit ciphertext $C = c_1, \dots, c_{64}$.

1) (Key schedule) Compute sixteen 16-bit subkeys K_i from K, using algorithm above for FEAL-8 key schedule.
2) Define $M_L = m_1, \dots, m_{32}, M_R = m_{33}, \dots, m_{64}$.
3) $(M_L, M_R) \oplus ((K_8, K_9), (K_{10}, K_{11}))$ becomes (R_0, L_0) (XOR initial subkeys).
4) $R_0 \oplus L_0$ becomes R_0.
5) R_{i-1} becomes $L_i, L_{i-1} \oplus f(R_{i-1}, K_{i-1})$ becomes R_i, for $i = 1, 2, \dots, 8$ (use the table for $f(A, Y)$ with $A = R_{i-1} = (A_0, A_1, A_2, A_3)$ and $Y = K_{i-1} = (Y_0, Y_1)$).
6) $L_8 \oplus R_8$ becomes L_8.
7) $(R_8, L_8) \oplus ((K_{12}, K_{13}), (K_{14}, K_{15}))$ becomes (R_8, L_8) (XOR final subkeys).
8) (R_8, L_8) becomes C.

Algorithm FEAL-8 key Schedule [Men 96]

The key schedule uses a function f_K as a function A_i, B_i, Y_i, t_i, and U_i that are 8-bit variables, for mapping two 32-bit inputs to one 32-bit output. As the operations of 2-bit rotation and XOR are both linear, the only nonlinear elementary operation in FEAL is addition mod 256.

Input: 64-bit key $K = k_1, \dots, k_{64}$.

Output: 256-bit extended key (16-bit subkeys $K_0, \dots, K_{15}$).

1) Take $\cup^{(-2)} = 0, \cup^{(-1)} = (k_1, \dots, k_{32})$ and $\cup^{(0)} = (k_{33}, \dots, k_{64})$.
2) Since $\cup = (\cup_0, \cup_1, \cup_2, \cup_3)$ for 8-bit $\cup_i$, compute $K_0, \dots, K_{15}$ as i runs from 1 to 8.
   a. $f_k(\cup^{(i-2)}, \cup^{(i-1)} \oplus \cup^{(i-3)})$ becomes $\cup$. (use the table for $f_k(A, B)$ with $A = (A_0, A_1, A_2, A_3)$ and $B = (B_0, B_1, B_2, B_3)$).
   b. $K_{2i-2} = (\cup_0, \cup_1), K_{2i-1} = (\cup_2, \cup_3), \cup$ becomes $\cup^{(i)}$.

FEAL decryption may be achieved using the above algorithm with the same key K and ciphertext $C = (R_8, L_8)$, but with the key schedule reversed. More specifically, subkeys $((K_{12}, K_{13}); (K_{14}, K_{15}))$ are used for the initial XOR, (Step 3) while $((K_8, K_9)$ and $(K_{10}, K_{11}))$ are used for the final XOR (Step 7), and the round keys are used fromK7 back to K_0 (Step 5).

2.3 Background of Differential Cryptanalysis

In 1993 Biham and Shamir [2] have developed a type of cryptanalitic attack that can break DES-like cryptosystems, and known as differential cryptanalysis, DC. They described an n-round characteristic, which allowed them to push the knowledge of the plaintext by making use of an XOR operation, to knowledge of an intermediate round. Every round characteristic has a particular plaintext difference $P \oplus P* = \Omega P$, a particular XOR of the data in the n^{th} round Ω_T and a probability (for Ω_T which are produced by using random pairs whose plaintext difference is Ω_P). Any pair whose plaintext difference is Ω_P and whose XOR of the data in the n^{th} round, using a particular key, is Ω_T is called a right pair with respect to that key and the n-round characteristic. Any other pair is a wrong pair. Therefore, the right pairs form a fraction of all possible pairs.

DC attempts to find out the round key K_n. Then for two plaintext $P, P*$ of difference Ω_P the cryptanalyst can solve the following equation for K_n:

$$F^{-1}(C_n, K_n) \oplus F^{-1}(C^*{}_n, K_n)^{-1} = \Omega_T.$$

The solutions are candidate round keys. The method of DC can be summarized as follows [8]:

Step 1. Find a proper round characteristic with high probability.

Step 2. Uniformly select a plaintext pair $P, P*$ with difference Ω_P and get the encryption of this pair. Determine candidate round keys such that each of them could have caused the observed output difference. Increment a counter of each candidate round key.

Step 3. Repeat Step 2 until one round key is distinguished as being counted significantly more often than other round keys. Take this key to be the actual candidate round key.

Biham and Shamir found that, from experiments on restricted versions of FEAL, the complexity of the attack was approximately c/p^{Ω}, where p^{Ω} is the probability of the characteristic being used, and c is a constant bounded as $2 < c < 8$. They used the signal to noise ratio S/N to measure the efficiency of DC. Assume that m pairs of chosen plaintexts are used in DC and that p^{Ω} is the probability of the characteristic used. Then about $m \times p^{\Omega}$ pairs are right pairs, each of which actually can suggest the right key value among other values. In some cases the attacker can classify pairs for the plaintext as wrong pairs using the intercepted ciphertexts. In this case such pairs are discarded and should not be used in the analysis.

The signal to noise ratio S/N determines the number of times the right key is counted over the number of times a random key is counted, i.e., $S/N = \frac{k \times p^{\Omega}}{\gamma \times \lambda}$, where k is the number of possible values of the key we are looking for, γ is the number of keys suggested by each non-discarded pair of plaintexts and λ is the ratio of non-discarded pairs to all pairs [11]. A necessary condition for the success of a DC attack is that $S/N > 1$ and the expected success of the attack increases with that ratio.

Actually, DC attacks need a large number of right pairs that consume memory and time to suggest the encrypted key. On the other hand, GA's cannot be directly applied on the population of keys represented in the form of chromosomes. Therefore, DC is needed to determine the right pairs. This is accomplished by examining - in each round - the input difference, which causes the correct output difference, produces the last actual subkey K_7, which is defined in [2]. Such pairs are needed to obtain the subkeys of the key.

3 Genetic Algorithms

Genetic Algorithms (GA's) had been applied by Holland [6] as an adaptive heuristic search method that depends on the evolutionary ideas of natural selection and genetics. The basic goal of a genetic algorithm is to simulate the process of natural evolution, taking into consideration the principle of survival of the fittest. It is generally used in situations where the search space is relatively large and cannot be traversed efficiently by classical search methods. This is mostly the case with problems whose solution requires evaluation of many apparently unrelated variables. GA's represent an intelligent mapping of a random search space to a guided search space in which the problem solution could be found. The algorithm performs the following steps:

1) Generate an initial population, randomly.

2) Compute the fitness for each individual in the current population.

3) Define selection probability of each individual so that it is proportional to its fitness.

4) Generate the next current population by probabilistically selecting the individuals from the previous current population, in order to produce offspring via the genetic operators represented by: selection, crossover and mutation.

5) Repeat Steps 2, 3 and 4 until satisfactory solution is obtained.

Holland has analyzed the influence of GA operators (selection, crossover and mutation) on the number $m(H, t)$ of schemata H when going from one-generation t to the next $t + 1$. A good discussion can be found in [5]. Holland's schemata theorem can be expressed as:

$$m(H, t+1) \geq m(H, t) \frac{f_H(t)}{f(t)} [1 - p_c \frac{\delta(H)}{l-1} - o(H) p_m],$$

where

$f_H(t)$ is the fitness value of the string representing schema H,

$\overline{f}(t)$ is the average fitness value over all strings in the population p_c, p_m are probabilities of crossover and mutation respectively, l is the schema length,

$\delta(H)$**:** Length of schema H measured as the distance between the first and the last fixed string positions of schema H,

$o(H)$**:** Order of schema H, defined by the number of fixed string positions of schema H.

This implies that the fitness function will grow up when better offspring are used. This fact as well as the ability of Genetic Algorithms to search efficiently huge spaces, would afford GA's as natural candidate for use in cryptanalysis [9].

4 Using GA for Analysis of DES-like Systems

In what follows GA's have been exploited to calculate the key of some DES-like cryptosystems by two methods:

1) Using a number of DC generated right pairs, which are stored in order to be implemented with a proper characteristic.

2) Generating right pairs genetically.

4.1 The Method of Stored Right Pairs

First, the proper number of right pairs, with respect to the key, along with the proposed characteristic is stored for future processing. For each one of these right pairs there exist a number of expected keys, for every S-box. The GA is used to find out the output bits for each S-box, in the last subkey. In any iteration, the S-box output bits constitute the current chromosome of the GA. The chromosome correctness is determined by making use of the following theorem:

Theorem 1. *The chromosome correctness* $C_r = \frac{n_{s_r}}{n_P}$ *(where is the number of right pairs for the current chromosome* r *and* n_P *is the total number of stored right pairs) can be successfully used as a fitness function of a genetic algorithm.*

Proof. Since $C_r = \frac{n_{s_r}}{n_P}$, then it monotonically increases with the increase of n_{s_r} and

$$\lim_{n_{s_r} \to n_P} C_r \to 1.$$

Taking $f(S) = C_r$, for the schema S, then the fitness $f(S)$ monotonically increases with the increase of n_{s_r} and

$$\lim_{n_{s_r} \to n_P} f(S) = 1.$$

That is, $f(S)$ is always less than 1 except when $n_{s_r} = n_P$. This guarantees that Holland's schemata theorem is satisfied and the expected number $m(S, t+1)$ of representative schema S at time $t+1$ is always greater than or equals the number $m(S, t)$ of S at the previous time t. Then $m(S, t+1) \geq m(S, t)$, means that the number of schemata is growing up and proves this theorem. □

Since the fittest chromosome is the one that satisfies the entire number of right pairs n_P, then the fittest chromosome will make C_r reach as 1.

In average, $\overline{C_r} = \frac{1}{l} \sum_{i-1}^{l} C_{r_i}$, where l is the chromosome length that represents the schema S. The population size should be greater than or equal to l. The stored right pairs, which have been prepared by DC are used to obtain some key bits using algorithm denoted by SPCA (Stored Pair Cryptanalysis), emphasized below.

4.1.1 Algorithm SPCA

This is the proposed algorithm that can be applied to DES-like systems using Theorem 1 as fitness.

Input: number of right pairs with respect to the expected key along with the proper characteristic.

Output: some bits of the last round subkey (segment of that key).

Procedure:

1) Read the stored right pairs n_P;

2) For each segment do:

   a. Create an initial population in which each individual (chromosome) has number of bits equal to that segment of the last subkey input of the current segment.

   b. Evaluate the fitness $C_r = \frac{n_{s_r}}{n_p}$ for each individual r of the population in the current generation.

   c. Apply crossover operation.

   d. Apply mutation operation, if needed.

   e. Upon convergence take the fittest chromosome, which may be an expected key in the current segment.

3) Put the correct bits in their positions in the last subkey.

4) Calculate the position of the unknown bits of the key.

5) Apply the exhaustive search on one pair to get the remainder bits of the key.

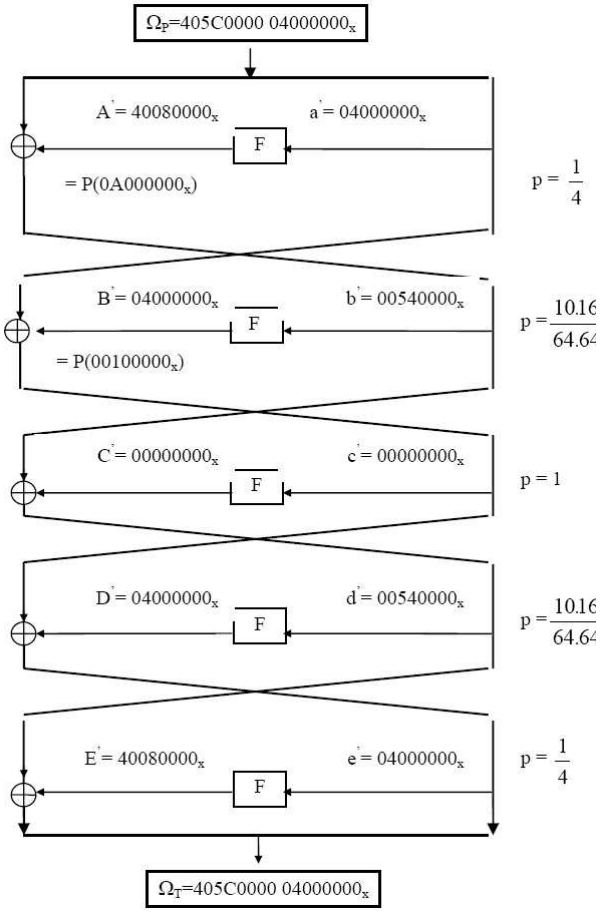

Figure 1: The 5-round characteristic with probability $\frac{1}{10485.76}$

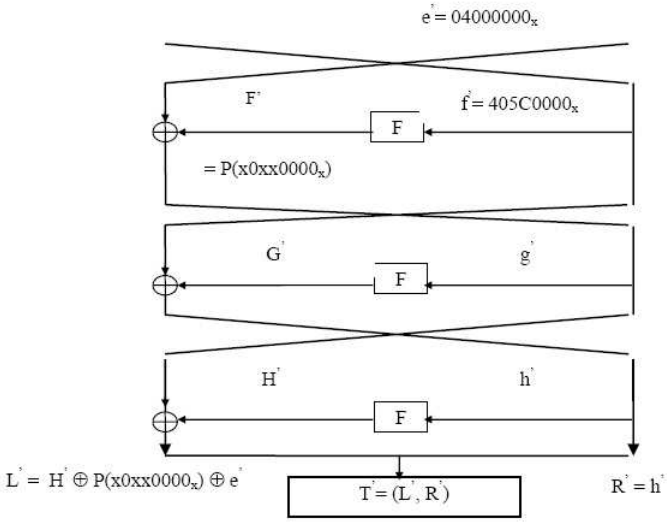

Figure 2: The last 3 rounds of a DES-8 system

4.1.2 Application of SPCA to DES-8

Here the cryptanalysis DES-8 is considered. For each one of the eight S-boxes (as segment of the eighth subkey), the genetic algorithm, SPCA, is used to find out a 6-bit chromosome. The emphasis is on the first 8 rounds while the initial and final permutation are omitted, since they are not important for the attack analysis. Such analysis is based on using a number of right pairs, which were generated differentially and stored in working area. By making use of the 5-round characteristic, $\Omega P = 405C000004000000_x$, with probability $\frac{1}{10485.76}$, Figure 1, in the analysis proceeds. Particularly, these pairs are generated and stored, by satisfying the causing condition for S2, S5, S6, S7, and S8 S-boxes for the subkey K8. Thus one can calculate correct 36 bits in K8, for S-boxes: S1, S2, S5, S6, S7, and S8 using GA. Figure 2 shows the remaining three rounds of the 5-round characteristic that complete the eight rounds of DES-8 system. The right pairs have been satisfied the condition:

$$R' = h', L' = H' \oplus P(x0xx0000_x) \oplus 04000000_x,$$

where P is the permutation round. For each right pair, $\Omega P = P \oplus P^*$ there is a corresponding ciphertext pair T and $T*$, with the difference $T' = T \oplus T^*$. The right half of T' is R', and the left is L' Figure 2. For every S-box there is a corresponding 6 bits SK in K8 satisfying the causing condition.

The algorithm SPCA has been applied to break down DES-8 using a chosen plaintexts/ciphertexts attack. In this case 1000 right pairs are computed "differentially" and stored in a particular list structure. These pairs are used as the input of SPCA which has been carried out with the following parameters:

- Number of right pairs = 100,
- Population size = 5,
- Chromosome length = 8,
- Probability of crossover = 0.6,
- Probability of mutation = 0.2,
- Maximum number of generations = 100,
- Seed of random process = 0.8.

In this algorithm the value of the fitness function C_r should satisfy the threshold condition $C_r \geq 0.15$. Otherwise the underlying S-box is by-passed and the next S-box is considered. The algorithm performance is reported in Figure 4, which shows that increasing the number of right pairs reduces the number of runs and consequently the time required to obtain the correct key.

4.1.3 Application of SPCA to FEAL-8

Here the cryptanalysis of FEAL-8 is considered. For the last actual subkey (as segment of the eighth subkey), the genetic algorithm, SPCA, is used to find out a 16-bit chromosome. Such analysis is based on using a number of right pairs, which were generated differentially and stored in

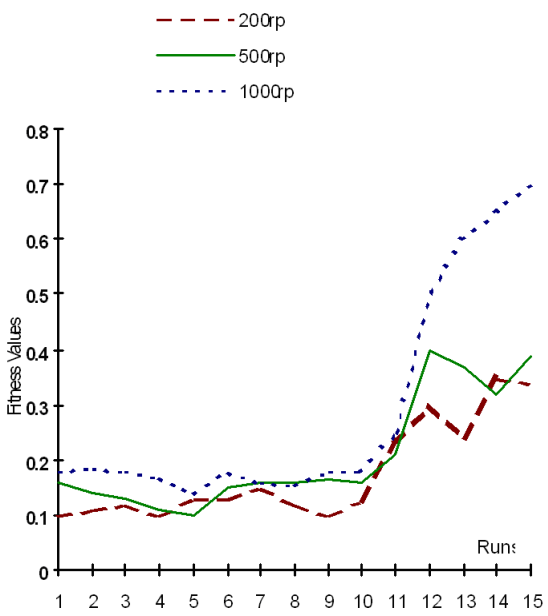

Figure 3: The computational results for 200, 500 and 1000 right pairs are used for finding S1 genetically

working area. By making use of the 5-round characteristic, Figure 1, $\Omega_P = A200800022808000_x$, with probability $\frac{1}{16}$, in which $P' = A200800022808000_x$, the analysis proceeds. Particularly, these pairs are generated and stored, by satisfying the causing condition, with respect to FEAL, for the last subkey K_7. Thus one can calculate correct 16 bits in the last actual subkey AK_7.

For each right pair, $\Omega_P = P \oplus P^*$ there is a corresponding ciphertext pair T and T^*, with the difference $T' = T \oplus T^*$. The right half of T' is R', and the left is L' Figure 2. For every S-box there is a corresponding 6 bits SK in K8 satisfying the causing condition.

The algorithm SPCA has been applied to break down FEAL-8 by using the chosen plaintexts/ciphertexts attack. In this case 1000 right pairs are computed "differentially" and stored in a particular list structure. These pairs are used as the input of SPCA, which has been carried out with the same parameters values of algorithm SPCA.

In this case the value of the fitness function C_r should satisfy the threshold condition $C_r \geq 0.15$. Otherwise, the underlying S-box is by-passed and the next S-box is considered. The algorithm performance is reported in Figure 4, which shows that increasing the number of right pairs reduces the number of runs and consequently the time required to obtain the correct key.

4.2 The Method of Generated Right Pairs

This method is based on a memoryless approach and it exploits the idea, that without storing any pair, the fitness

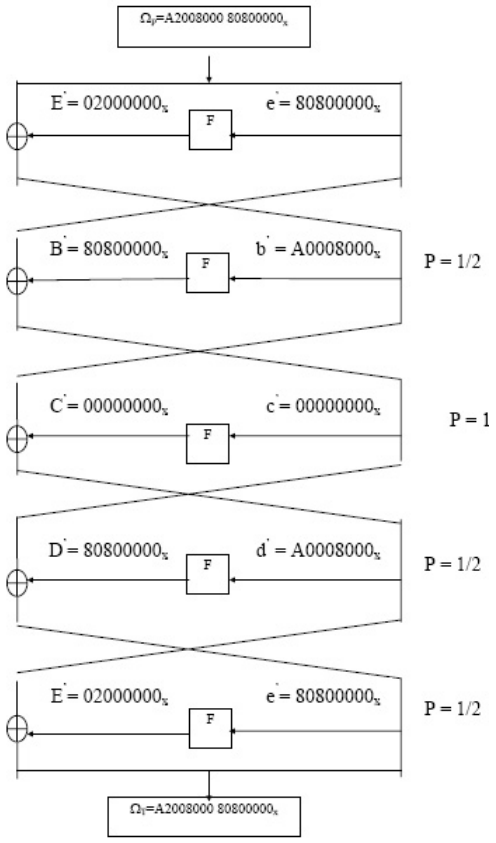

Figure 4: The five-round characteristic with probability 1/16

function can be used to generate right pairs that satisfy a proper characteristic. After generating a proper number of right pairs we get a number of last subkeys. The last subkey which is repeated with the largest number of right pairs is the candidate subkey.

Theorem 2. *Let Ω_P and Ω_T be the input and output pair of the underlying characteristic, respectively. Then any pair P, P^* such that $\Omega_P = P \oplus P*$, enciphered by a DES-like, to $T' = T \oplus T*$ is a right pair. Accordingly, such right pair maximizes the fitness function given by:*

$$Fitness(\Omega_T, T') = 1 - \frac{H_d(\Omega_T, T')}{n},$$

where $H_d(\Omega_T, T')$ is the Hamming distance between Ω_T and T' whereas n is the block length. This function can be successfully used as fitness function for genetic algorithms to break down DES-like systems.

Proof. If P, P^* is right pair then $\Omega_T = T'$. Hence, $H_d(\Omega_T, T') = 0$. Also, when $H_d(\Omega_T, T')$ decreases, the expectation of the right pair increases.

Thus $\lim_{T' \to \Omega_r} Fitness(\Omega_T, T') = 1$ and the fitness, as such, increases monotonically with the decrease in distance. Then, as in Theorem 1, it means that the number of schemata is growing up with the fitness increase, and this proves the theorem. □

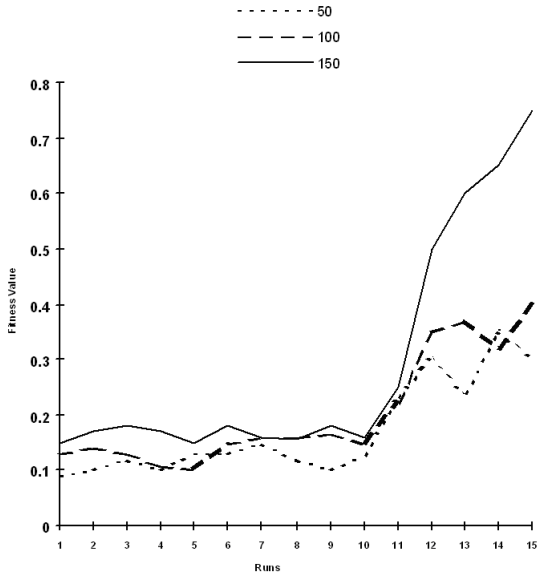

Figure 5: The computational results for 50, 100 and 150 right pairs that have been used for finding mx(ak7) genetically. Where two bytes mx(ak7) = (ak7(0) XOR ak(1), ak7(2) XOR ak(3)), ak(i) is byte, $0 \leq i \leq 3$

The following algorithm, that can be used to generate the required right pairs genetically denoted by GPCA (Generated Pair Cryptanalysis).

4.2.1 Algorithm GPCA

This i algorithm can be applied for DES-like systems and guided by the fitness function given in Theorem 2.

Input: difference of two plaintexts with respect to a proper characteristic.

Output: some bits of that key.

Procedure:

1) Create an initial population in which each individual (chromosome) is the first plaintext P.
2) For each chromosome do:
   a. Evaluate the second plaintext $P* = \Omega_P \oplus P$, where Ω_P is the characteristic difference.
   b. Obtain the ciphertext pair $T' = T \oplus T^*$.
   c. Evaluate the Hamming distance $H_d(\Omega_T, T')$.
   d. Form a two-dimensional table $\tau =< \varepsilon_S, \zeta_S >$, where ε_S is an expected subkey and ζ_S is the corresponding counter for it. Set all counters to zero.
   e. Compute the fitness function $Fitness(\Omega_T, T') = 1 - \frac{H_d(\Omega_T, T')}{n}$, where n is the system block length, for each individual in the current population.

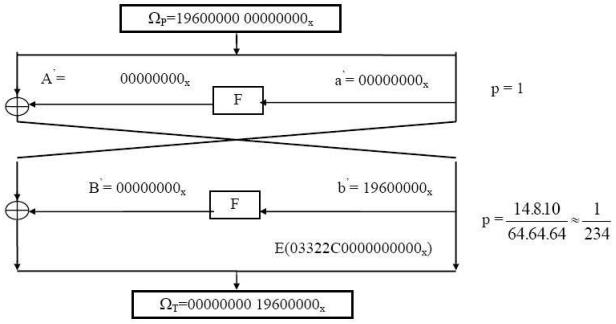

Figure 6: The 2-round characteristic with probability $\approx \frac{1}{234}$

   f. If the fitness value is greater than half then the pair is right pair.
   g. If the pair is right then:
      i. Produce from table τ subkeys for this pair.
      ii. Generate all possible bits that may appear in the last round subkey (associated with the concept of DC) by choosing one subkey, ε_S for the underlying pair from table τ. Denote such bits by σ.
      iii. For each σ, increment the corresponding counter ζ_S.
3) Apply crossover operation.
4) Apply mutation operation, if needed.
5) Generate the next population.
6) Repeat step ii to obtain the counter of the maximum value ζ_{opt}. Such counter is associated with σ_{opt} that is probably a correct expectation for the last round subkey.

4.2.2 Application of GPCA to DES-8

Here the cryptanalysis DES-8 is considered. For each one of the eight S-boxes, the genetic algorithm, GPCA, is used to find out a 64-bit chromosome. By making use the 2-round characteristic with probability $\approx \frac{1}{234}$, $\Omega_P = 1960000000000000_x$, Figure 3. Thus, by using the concept of DC, one can calculate correct 18 bits in K_8 for S-boxes S1, S2, and S3. $H_d(\Omega_T, T')$ measures Ω_P and $FP^{-1}(T')$.

Actually algorithm GPCA is a "deepening" of role of GA's in cryptanalysis. In this case 1000 pairs are generated "genetically" and employed as input to the algorithm GPCA. The algorithm is executed with the same parameters values of SPCA.

Figure 7 shows the change of the fitness function, $Fitness(\Omega_T, T')$, with increasing the number of generations. Also, it shows the effect of increasing the number of right pairs on the required number of runs (generations).

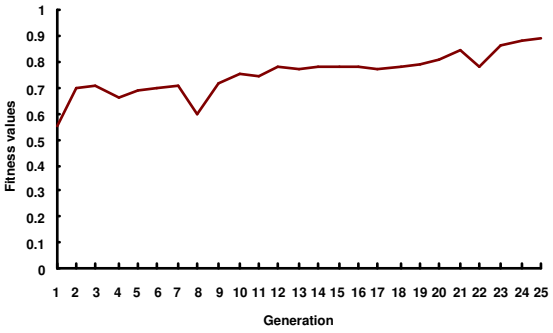

Figure 7: The results of 200 right pairs are used for finding S1 genetically

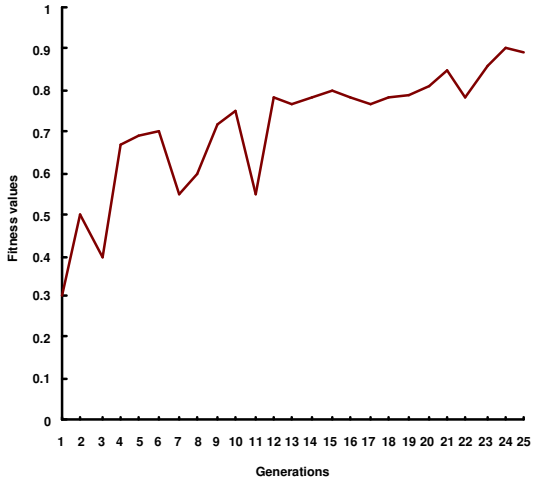

Figure 8: The results of 100 right pairs are used for finding mx(AK7) genetically

4.2.3 Application of GPCA to FEAL-8

Here the cryptanalysis FEAL-8 is considered. For the last actual subkey, the genetic algorithm, GPCA, is used to find out a 64-bit chromosome. By making use of the 5-round characteristic, Figure 1, $\Omega_P = A200800022808000_x$, with probability $\frac{1}{16}$, in which $P' = A200800022808000_x$, the analysis proceeds. $H_d(\Omega_T, T')$ measures Ω_P and T'. Using the concept of DC for FEAL to produc the last actual subkey,AK8.

Actually algorithm GPCA is a "deepening" of role of GA's in cryptanalysis. In this case 1000 pairs are generated "genetically" and employed as input to the algorithm GPCA. The algorithm is executed with the same parameter values that have been mentioned above.

Figure 8 shows the change of the fitness function, $Fitness(\Omega_T, T')$, with the number of generations.

5 Conclusion

In DES-like cryptosystems the language properties do not characterize the ciphertext. Therefore, it is impossible to find out a straightforward fitness function that can guide the search to the target plaintext.

It has been presented that the use of genetic algorithms can improve the cryptanalysis of DES-like cryptosystems. For convenience these algorithms are applied on DES-8 and FEAL-8. In these cases, the following concluding remarks are pointed out.

1) GA's can be either combined with differential cryptanalysis methods or relied upon solely to break down block-ciphered texts. Consequently, first: The total number of right pairs, n_p, is computed "differentially" and stored in the memory. Hence the number n_s of right pairs satisfying the current chromosome, is evaluated. The ratio $C_r = \frac{n_S}{n_p}$ is exploited, for the first-time to serve as fitness function for GA. Second: The total number n_p, of right pairs, is calculated "genetically". Accordingly, a weight $w = \frac{Hamming-distance}{Chromosome-length}$ is calculated. For that w, the hamming distance is the difference between the current right pair and the output of a high probability characteristic. Again $(1 - w)$ is exploited, for the first time as a fitness function. This scheme is superior since it is not required to store any right pair.

2) The problem of using a huge number of right pairs can be solved by generating the right pairs genetically. Such generation process is carried out by exploiting the relation $Y = \Omega_P \oplus X$. Thus if Ω_P is available, then Y can be obtained when X is genetically generated. The pair that satisfies the causing of the underlying S-boxes may be an expected key.

3) Despite the fact that the time complexity of GA's is $O(n^3)$, where n is the input size, computing the right pairs (needed for estimating the key) genetically are faster than differentially. This is due to the fact that no right pair are stored and examined for the former technique.

4) The mutation operation is used to accelerate the break down process. In our experiments the best value of mutation rate is about 0.2. Actually, other improvements such as elitism can be added to accomplish the required cryptanalysis.

5) The performance evaluation of SPCA and GPCA indicates that genetic algorithms can successfully replace the available cryptanalysis methods of DES-like systems. A comparison with Biham, [2] shows that he has used 15000 right pairs of plaintext/ciphertext in order to get 18 bits out of 48, however, the use of GA's could reduce the number of needed right pairs from 15000 to about 5000 pairs, in order to obtain 30 bits out of the same subkey K8.

References

[1] V. R. Andem, *A Cryptanalysis of the Tiny Encryption Algorithm*, M. Sc., The University of Alabama, Tuscaloosa, 2003.

[2] E. Biham, and A. Shamir, *Differential Crypt Analysis of Data Encryption Standard*, Springer-Verlag, New York, 1993.

[3] C. Canniere, A. Biryukov, and B. Preneel, "An introduction to block cipher cryptanalysis", *Proceedings of IEEE*, vol 94, no 2, pp. 346-356, Feb. 2006.

[4] B. Delman, *Genetic Algorithms in Cryptology*, M. Sc in Computer Engineering, Rochester Institute of Technology, Rochester, New York, 2004.

[5] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Company Inc., Reading, Massachusetts, 1989.

[6] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Mich. Ann Arbor, 1975.

[7] P. Isasi, and J. C. Hernandez, "Introduction to the application of evolutionary computation in computer security and cryptography," *Computational Intelligence*, vol. 20, issue 3, pp. 445-449, Aug. 2004.

[8] L. R. Knudsen, *Block Ciphers- Analysis, Design and Applications*, DAIMI PB-485, Ph. D. Thesis, Aarhus University, Denmark, 1994.

[9] M. Kumari, "Genetic algorithms applications in cryptanalysis," in *Proceeding of the National Seminar on Cryptology*, pp. E1-E18, Delhi, Jul. 9-10, 1998.

[10] National Bureau of Standards, *Data Encryption Standard*, Federal Information Processing Standard (FIPS), Publication 46, National Bureau of Standards, U.S. Department of Commerce, Washington D.C., Jan. 1977.

[11] K. Nyberg, "Differential uniform mapping for cryptography," *Cryptology Eurocrypt'93*, LNCS 765, pp. 55-64, Berlin, Springer-Verlag, 1994.

[12] K. E. Parsopoubs, and M. N. Vrahatis, "On the computation of all global minimizers throng particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, issue 3, pp. 211-224, Jun. 2004.

[13] A. K. Verma, M. Dave, and R. C. Joshi, "Genetic algorithms and tabu search attack on the monoalphabetic substitution in adhoc networks," *Journal of Computer science*, vol. 3, no.3, pp. 134-137, 2007.

[14] D. N. Wilke, S. Kok, and A. A. Groenwold, "Comparison of linear and classical velocity update in particle swarm optimization: Notes on diversity," *International Journal for Numerical Methods in Engineering*, vol. 70, no. 8, pp. 962-984, 2007.

Bahaa Hassan was born in Cairo, Egypt on May 29, 1954; He received the B. E. and M. E. degrees in electrical engineering from Zagazig University in 1978 and 1987 respectively. He received the Ph. D. degree in computer and systems engineering from Ain Shams University in 1994. He is currently a General Manager in National Defense Council, Cairo, Egypt. His current research interests include cryptography and their applications, smart cards applications and computer and network security.

Bayoumi Ibrahim Bayoumi was born in El-Sharkia, Egypt on August 14, 1946. He received his B. Sc. from Faculty of Science, Ain Shams University, Egypt, June, 1967. M. Sc. from Faculty of Science, Ain Shams University, Egypt, June, 1970.Ph. D. from Leningrad state University, USSR, September, 1974. Now he is Professor of Mathematics, Faculty of Science, Ain Shams University, Egypt.

Fathy Saad Holail was born in Cairo, Egypt on November 14, 1946. He received the B. Sc., M. Sc., and Ph.D. degrees in mathematics from Cairo university, in 1970,1983 and 1985 respectively. During 1991 and 1992 he was a visiting scholar at Tsujii Laboratory of Information system security, Department of electrical and electronic Engineering, Tokyo Institute of Technology, Japan. His Current job is Head of C. R. Division, Research Development Center, National Defense Council, Cairo, Egypt. His current research interests includes design and implementations of encryption algorithms also cryptanalysis of cryptographic systems. He is a member of IACR, Mathematical Egyptian society and language Engineering society at Egypt.

Hassan Mohammed was born in Sharkia Egypt on May 29, 1951. He received the B. Sc., M. Sc., and Ph. D. degrees in pure mathematics from university of Cairo, Zagazig, and Ain Shams in 1974, 1987, and 2003 respectively. His Current job is one of C. R. Division, Research Development Center, National Defense Council, Cairo, Egypt. His Current research interests includes Design and Implementation of encryption algorithms also cryptanalysis of cryptographic systems.

M. Zaki (azhar@mailer.scu.eun.eg) is the professor of software engineering, Computer and System Engineering Department, Faculty of Engineering, Al-Azhar University at Cairo. He received his B. Sc. and M. Sc. degrees in electrical engineering from Cairo University in 1968 and 1973 respectively. He received his Ph. D. degrees in computer engineering from Warsaw Technical University, Poland in 1977. His fields of interest include artificial intelligence, soft computing, and distributed systems.