A new approach of known plaintext attack with Genetic Algorithm

T.MEKHAZNIA[1], A. ZIDANI[2], M. DERDOUR[3]
[1,2] LASIC Laboratory
University of Batna, ALGERIA
[3]LAMIS Laboratory
University of Tebessa, ALGERIA
mekhaznia@yahoo.fr

Abstract: - Cryptanalysis of modern cryptosystems is viewed as NP-Hard problem. Block ciphers, a modern symmetric key cipher are characterised with the nonlinearity and low autocorrelation of their structure. In literature, various attacks were accomplished based on traditional research algorithms such the brute force, but results still insufficient especially with wide instances due to resources requirement, which increase with the size of the problem. Actual research tends toward the use of bio-inspired intelligence algorithms, which are heuristic methods able to handle various combinatorial problems due to their optimisation of search space and fast convergence with reasonable resource consumption. The paper presents a new approach based on genetic algorithm for cryptanalysis of block ciphers; we focuses especially around the problem formulation, which seems a critical factor that depends the attack success. The experiments were accomplished on various set of data; the obtained results indicate that the proposed methodology seems an efficient tool in handling such attacks. Moreover, results comparisons of the considered approach with similar heuristics such Particle Swarm Optimisation and Brute Force reports its effectiveness in solving the considered problem.

Key-Words: - Block ciphers; Genetic Algorithm; Particle swarm optimisation; Cryptanalysis; Bio-inspired intelligence.

1 Introduction

Cryptography refers to the science of information's protection. Cryptographic techniques denote the mechanisms to achieve security goals such confidentiality and authentication that consists in preventing access even from indiscrete eyes and developing avenues of encryption methodologies [1][2]. They involves an input data called a *plaintext* and a small amount of information denoted a *key* in order to customize an output data called a c*iphertext* which viewed as unintelligible information obfuscated to all not intended parties and appears meaningful only through a legal decryption process. A *cipher algorithm* denotes the mathematical function that enables the encryption and decryption process. The set of plaintexts, ciphertexts, keys and related algorithms is called a *cryptosystem*. Actually, modern cryptosystems becomes a unavoidable tool for real life applications; research in this area becomes very intensive whatsoever on building new concepts or improvement of existent ones.

Cryptanalysis denotes the way of studying the concepts of cryptosystems in order to detect their weakness and attempt to extract parts of corresponded plaintext without knowing the secret data which normally needed for decryption such keys or algorithms. The term *attack* refers to the manner in using cryptanalysis on ciphertexts in order to reveal the original plaintext; if the plaintext is fully restored, the attack is *successful*. So, the strategies in building robust ciphers that prevent effective attacks seem a hard task especially with the increase of data transfer through public networks. The efficiency of cryptanalysis is based upon the knowledge of cryptosystems material such encryption algorithms, language specificities, parts of plain and cipher texts and eventually any *luck*. The knowledge of such information favourite right attacks [3]; its aims is to measure the cryptosystems strength and therefore, help researchers to perform more robust algorithms for upcoming times [4]. Cryptanalysis uses numerous techniques of attacks, depending of information available about cryptosystems such fragments of ciphertexts and their corresponding plaintexts, which permit, in certain circumstances to deduce the encryption keys. Also, success of attacks depends of available resources such processing time allowed, amount of storage memory and available experimental data. In general, there is no systematic way for an eavesdropper to recover a ciphertext; it may be necessary to understand the cryptosystem design and figure out its weakness based on decryption results analysis, flows of implementation or by building encryption keys behaviour.

While the majority of classical ciphers can be easily broken by applying metaheuristic techniques, research in cryptanalysis is focused on modern cryptosystems and especially on block ciphers. In order to evaluate their security, we assume that the attacker has access to all ciphered data, the encryption algorithm and in some cases, to other details such the subject of texts and the literary language in which they were written. However, the cryptanalysis is in general, limited to the *ciphertext only attack* which seems the most challenging kind of attacks. Under this assumption, the attacker generates several chosen keys and proceeds to the decryption. Upon these considerations, we show that blocks with reduced data are insecure; e.g, a key of 10 bits, such the Simplified Data Encryption standard algorithm (SDES), implying 2^{10} different ciphertexts which can be easily attacked by the *brute force attack, man-in-the-middle attack* or *frequency character analysis*. The blocks security can be increased by using wide size keys. Although, it becomes unfeasible for an attacker to try every possible alternative until the desired plaintext is found for keys of more than 56 bits such Data Encryption Standard algorithm (DES). In order to overcome this difficulty, a significant progress has been emergent in recent two decays; the first breakthrough date back to the 90s where Biham and Shamir have proposed the *differential cryptanalysis* which analyses the effect of the produced plaintexts and their correspondent ciphertexts [5]. It was tested on the DES [6][7]; experiments have shown that, a combination of 2^{47} *chosen plaintexts* is enough to reveal the encryption key. This result was improved by Matsui [8][9] who proposed the *linear cryptanalysis* for attack of the DES algorithm [10] which uses a relation between inputs and outputs of decryption algorithms that holds with a certain probability and showed that just 2^{43} *known plaintexts* are sufficient to reveal the decryption key. Later, Biryukov and Wagner [11] proposed the *integral cryptanalysis* or *slide attack;* it is based, in general on *known* or *chosen plaintexts*. These techniques are able to break various ciphers, nevertheless, and given their reduced setting, remain ineffective against a wide class of modern cryptosystems. By another way, the *brute force* is a common attack; it tries the 2^b possibilities of *b*-length key within the search space to find the right key. It has a successful outcome in breaking ciphertexts but need enough resources and so, has less success in practice. However, and in order to avoid this class of attacks, actual research in cryptanalysis of block ciphers tends toward *metaheuristic techniques* and especially, *bio-inspired algorithms* which have been found efficient in resolution of such problem. These methods have been successfully applied in a wide range of research application areas where they gets better results in a faster and cheaper way.

Bio-inspired algorithms are a general purpose approach based on bio-inspired intelligence. It is a well-known paradigm that successfully used as a powerful tools for solving complex combinatorial problems [12] with reasonable amount of resources consumption. Various works [13][14] shown that algorithms based bio-inspired intelligence have a successful potential to handle wide instances and may be adapted to produce approximate solutions for a large variety of optimisation problems. They use intelligent system that offers an independence of movement of agents, which tends to replace the preprograming and centralized control. In last few years, many of such algorithms were emerged [15]. In cryptology, bio-inspired algorithms seem an attractive tool for building block ciphers or encryption key recovery.

The objective of this contribution is to investigate a new way in which bio-inspired algorithms can be efficiency used for attack on encryption keys of block ciphers. The study provides a new approach of known plaintext attack based on *genetic algorithm* to overcome some limitations related to heuristics defectiveness. Its principle is inspired from an intelligent way of selection of individuals that consists and, at every generation, in building new individuals based on the performance of others individuals regardless of usual genetic operators. The goal of the approach is to allow a fast convergence without need of initial approximation to unknown parameters.

The rest of paper is organised as follows; section 2 presents a brief description of block ciphers, their characteristics and application areas. Section 3 begins with an overview of bio-inspired intelligence heuristics and introduces a brief description of genetic algorithm. The section 4 presents a historical background of various attacks of block cipher using bio-inspired techniques and focuses on the performance of the used methodologies. In section 5, a proposal for using genetic algorithm as attack tool of block ciphers is given; followed by a formulation of the problem and a description of algorithms and parameters. Various experiments, results and comments are illustrated in section 6 followed by a brief discussion in section 7. Finally, the section 8 presents the paper conclusion and directions of future work.

2 Block ciphers

Block ciphers can be either public or symmetric keys. In following, we focus on these last class [16] which uses an iterated encryption function $E: \{0,1\}^n x\{0,1\}^k \rightarrow \{0,1\}^n$. It takes an *n*-bits input of plaintext *M* and a *k*-bit key *k* and in return, produces a unique *n*-bits ciphertext *C*. The process is accomplished by a sequential *r* times repetition of a nonlinear complex transformation *E* based on several substitution and permutation of basic operations. Its principle consists, and for every iteration *i* of the encryption process (figure 1), to split the n-bits block of M into two identical halves L_i and R_i. Then, a *round function f* is applied to a one half using a *subkey* k_i, derived from a main secret key *k*; the output is exclusive-ored with the other half. Then, the two halves are swapped as effect presented by relation (1).

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

The *f* function is nonlinear and invertible, usually represented as a substitution boxe (called *sboxe)*. It produces an output block of *m* bits size based on a input block of *n* bits size (m<n) which ensures that all subsequent blocks are different.

The block ciphers are built based on nonlinearity and low autocorrelation and characterized by their simplicity of implementation, high speed of encryption and resistance against various attacks, nevertheless, blocks with small size (n < 64) are vulnerable to attacks based on statistical analysis where the compilation of frequency statistics of plaintexts becomes feasible with reasonable computing resources.

The advantage of such algorithm is that the encryption and decryption functions are identical. To allow a unique decryption, the encryption transformation must be a bijection, defining one-to-one on n-bits of each encrypted block. So, to reverse a round, it is only necessary to apply the same transformation again, which will cancel the changes of the binary operation XOR.

Block ciphers algorithms became a basis component in many encryption schemes such Blowfish [17], LUCIFER [18], CAST [19], DES [20], IDEA [21], RC5 [22], FEAL [23] and AES [24].

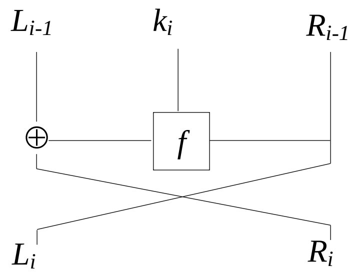

Fig. 1. A bloc cipher round

3 Bio-inspired intelligence heuristics

Bio-inspired intelligence is a sub-branch of artificial intelligence; it denoted the collective behaviour of natural and artificial systems including several agents, interacting locally among themselves and their surrounding environment according to a set of basis rules. The early concept has been emerged many decades ago [25] and takes its inspiration from the comportment of organisms that lives in communities. In fact, organisms are relatively unintelligent on their own but doted of a capability to achieve tasks necessary for their survival (food quest, safety, strategies of evading or relocation) in a synchronized and decentralized way without need dictating how each individual agent should act [26]. Bio-inspired techniques are adaptive strategies and concepts, defined to model the life activity of social swarms in nature. They employs a population of individuals that explore the search space for promising regions and solicited as a robust low-cost tool for resolution of various complex problems [27][28]. This principle is used to develop general-purpose algorithms, characterised by their flexibility and ability in resolving hard tasks with rational resources consumption.

Since the first idea in bio-inspired intelligence was proposed, several related strategies were emerged in literature, namely, Ant Colony Optimisation [29], Bacteria foraging [30], Honey bee swarm [31], Cat swarm optimisation [32], Immune optimisation Algorithm [33], Glowworm [34], Bat algorithm [35], Wolf optimiser [36], etc. The performance of bio-inspire algorithms has increased the attention of researchers to use them into their own activities. The most reporting results show the successful of such algorithms application in numerous real life domains including structural optimisation [37], scheduling [38], data analysis

[39], multi-objective or dynamic problems in machine learning [40] and other real-life disciplines [41].

Bio-inspired algorithms allow handling very large spaces of solutions; however, they do not fully show their competitive edge over certain static problems. They suffer from stagnation situations caused by the lack of central coordination and drawback of parameters tuning. So, they are not suitable for time-critical applications (such online control systems) or deterministic applications given that they cannot guarantee if an optimal solution is ever found; that's, the space exploration becomes useless, if a cross between a local and global solution occurs [42][43].

Genetic Algorithm (GA in short*)*, a sub-class of bio-inspired intelligence optimisation methods, based on genetic inheritance of species in a way related to a competition principle. This fact leads individuals (called *chromosomes*) that exhibit a better experience and adaptation to their environment to survive and transmit their genetic material to subsequent generations by using evolution operators such *crossover* and *mutation*. The selection of fitting chromosomes among each generation is completed based on a fitness criterion in regard of environment behaviour. The first formal model of GA was introduced in literature by [44], followed by numerous extension studies [45] [46] [47] which are successfully applied in resolution of a wide area of optimisation problems such design, scheduling, control and robotics [48][49].

In practice, GA denotes a methodology built on the concept of artificial systems, which have the ability of adaptation to environmental changes and allow providing approximate solutions to combinatory problems. Their specificity is to merge the exploitation of accomplished results with the exploration of new locations in order to accomplish tasks that cannot be performed efficiency by classical search methods. GA mechanism starts usually with a population p_0 of n individuals (vectors of finite elements); each represents a possible solution s_i of the considered problem. The evolution process consists, and through a certain number of generations, on an iterative application of stochastic operators, which alter individuals structure in order to produce better ones; each generation keep a fixed number of individuals based on their fitting results.

The following steps describe the GA principle:

- Evaluation of each current solution based of a given fitness function.
- Selection of a set of solutions $s \in S$ with $s \leq n/2$ according to certain rules related to a fitness criterion.
- Submit selected solutions to crossover and mutation in order to reproduce different further solutions $s' \in S$.
- Build a new set of solution s'' based on s and s' which replaces S.

The process will continue until a satisfactory solution is emerged or in the limit of a stopped criterion.

Genetic operators are 'blind' transformations operate on individuals regardless to the considered problem. The simplest form of GA involves three operators:

- Selection: select a set of individuals for reproduction.
- Crossover: consists on swap of bit sequences between two individuals in order to create new offspring. The aim of such transformation is to produce a variety of structures and avoid the domination of a particular type.
- Mutation: allow to flip some of individual elements with a low probability. Its aim is to keep available all individuals characteristics within each generation.

The efficiency of GA depends on the well choice of various values of used parameters: population size, crossover points, fitness function characteristics and the number of generations, which rely, in general to the experience and the intuition.

The GA has proved their success in resolution of various optimisation problems [45] [50] [51][52][53]. In cryptanalysis, such algorithm has been used as a guided random search toward adequate space areas [54]. It allows a fast and not premature convergence where only better solutions are retained upon each generation [55].

4 Block ciphers cryptanalysis

4.1 Literature review

Over last decay, a significant part of research in bio-inspired intelligence techniques related to cryptanalysis of block ciphers has been reported. In literature, most of contributions have employed a research methodology built upon heuristics where the objective function is based on frequency character analysis. Bafghi and Sadeghiyan [56] proposed a model for finding suitable key characteristics on the last decryption round by using ant colony optimisation technique (ACO); results were acceptable and provided a reduction of search space. In their works, Clark et al. [42] proposed an approach based on Boolean functions for attack of sboxes structures; experiments allow providing well results for reduce data. On another way, Laskari et al. [57][58][59] used a variety of computational algorithms such particle swarm optimisation (PSO) in resolution of cryptographic problems and demonstrate their effectiveness in cryptosystems security. Also, Song et al. [60] showed that GA constitutes an effective tool for attack of Feistel ciphers. Also, the idea proposed by Nalini and Rao [61] included experiments of attack of symmetric ciphers using various heuristics; results proved the effectiveness of such techniques for cryptanalysis of DES and DES reduced to four rounds (DES-4). Later, Husein et al. [55] combined the GA with the differential cryptanalysis in order to develop a fast algorithm for the attack of the DES; experiments were carried out on DES reduced to eight rounds (DES-8); they proved the efficiency of the results obtained compared to exhaustive search and differential cryptanalysis. The GA were also applied by Garg et al. [62] for attack of SDES cryptosystem where the experiments permit retrieval of most key bits. GA also have been used by Yang et al. [63] for attack of DES reduced to six rounds with a specific fitness function based on Hamming distance of relevant binary strings; the results obtained indicate that the proposed approach are effective to attack such cipher. On another way, bio-inspired algorithms have been used by [64] and [65] for attack of block ciphers such DES-4 and DES; they showed that the PSO is more competitive than GA on the attack DES-4. Also, this heuristic has been used by Pandy and Mishra [66]; experiments revealed most bits of used decryption keys and showed that these algorithms may be a powerful tool in cryptanalysis of such ciphers. Also, Vimalathithan and Valarmathi [67] have proved the effectiveness the PSO for reducing the search key space of various symmetric ciphers. The known plaintext attack has also been an interest of many researchers; in this context, an idea that involves reducing search space of 4DES by deducing correct bits from keys themselves has been opted by Hamdani et al. [68]; experiments were carried based respectively on Particle Swarm Optimisation (PSO) and artificial immune system (AI) achieved better results. Another alternative has been proposed by Khan et al. [69]; it consists in the use of the ACO for attack of DES-4 cryptosystem; ants moves throw a generic search space of two vertices labelled '1' and '0' and at every iteration, each ant select a node label based on its fitness value. Experimental results allowed finding 19 bits (among 56) of DES encryption key. Abd-Elmonim et al. [70] proposed another approach based on particle swarm optimisation in attack of DES cryptosystem. They attempts to deduce root key bits by analysing the difference of sboxes inputs in the first and last round; results allowed locating 39 bits among a total of 48 bits. Another interesting idea has been proposed by Jadon et al. [71] as an improvement of differential cryptanalysis. It consists to recover the remaining 14 bits of DES key by using a binary PSO algorithm; the fitness function is based on the difference between the number of ciphertexts pairs used by differential cryptanalysis and the number of pairs that satisfy the known plaintexts. The produced results showed that such idea is a better tool in attack of Feistel ciphers.

4.2 Analysis

The study of various approaches mentioned above shows that most attacks affect reduced versions of DES, namely SDES and DES with a limited number of rounds; this fact proves that DES with 16 rounds remains robust and resistant to various attacks. The few studies and attacks that have addressed DES such [72] where the results seems interesting: more than 26 key bits are revealed. However, they provide no indication about experimental environment such ciphers characteristics which should be, generally extract from known *Corpus* in order to approve the results quality.

The second remark refers to the ambiguity related to the environment parameters. It is easy to note that each metaheuristic algorithm includes not less than ten parameters, which should be evaluated before starting the attack process. Most of works cited have proceeded to an intuitive evaluation or, in some cases, to an empirical estimation of parameters values according some characteristics related to the considered problem, while others contributions precede to experiments without mention about the manner of parameters initialisation. Also, the ciphers size has its importance in the quality of results; A large size cipher allows in general, better results; this rule is not confirmed by certain works such [73].

As it mentioned, the most cited results are divergent. The results obtained are, in some cases unjustified and limited to their experimental environment. This fact prevents the possibility of experiments reproduction and loss of their efficiency and competiveness, especially in absence of standard benchmarks of ciphers database that can be used by the research community in order to measure the quality of results. Based on these considerations, we agree that the character frequency analysis (which adopted as a fitness evaluation by some contributions related to modern symmetric ciphers) cannot be considered as an efficient way of evaluation of block cipher decryption since the attack of such ciphers experience the avalanche effect that hides the statistical information of plaintexts [74]. In addition, such evaluation may be used only for ASCII texts; so, the conversion of binary texts to ASCII form produces in sometimes, non-alphabetic characters, which cannot be evaluated by the considered fitness function. On another way, attacks techniques related to block ciphers are dedicated and not intended other cryptosystems. They also require high computing resources given the non-linearity of the sboxes and the wide size of decryption keys [75].

5 Proposed approach

In general, cryptanalysis is viewed as a discrete search problem through a finite key space. The complexity of such problem is measured based on the average number of candidate keys required to mount the attack; it is therefore, proportional to the space size. In literature, the most attacks have been performed in a way that reducing the complexity of such problem; so, a cipher with wide key spaces is considered secure. This fact needs much attention to the problem formulation in order to avoid modest results not so far than random search [76].

In this context, we have inspired from the idea of Hamdani et al. [68] where the deduction of correct plaintext bits seems a best way of candidate keys. It consists in practice, on the generation of a specific form of keys, which satisfy a maximum of correct bits, and therefore, allows a fast convergence.

5.1 Key representation

In block ciphers, a key is a vector of *n* bits. Therefore and, for convenience, we use the binary GA where each chromosome is represented by a stream of bits. Therefore, genetic operators consist on swapping or inversion of keys bits.

5.2 Fitness function

The fitness function must include the maximum of material that exhibit the ciphertexts properties and may be independent of statistical information of the language of texts. It represents the difference between real and candidate keys. Since the real key is unknown, the distance is evaluated based on the number of correct bits in the produced plaintext in regard of the known plaintext. In usual decryption of block ciphers, a plaintext M_s is obtained based on a ciphertext C_s and a key k_s. In absence of k_s, the attacker proceeds to a decryption of C_s by a trial key k_t in order to obtain a target plaintext M_t. The fitness function $f^{\,t}_{\,k}$ measures the difference between M_s and M_t. This situation requires M_s as an input referential for comparison. A close difference between M_s and M_t denotes an adequate solution. The fitness function is built upon this idea. In literature, such function has been proposed under various combination schemes [63-64][68-72] [77]. The most commonly used is given by equation (2).

$$f_k=1-S/n \tag{2}$$

where S denoted the number of same bits in identical positions between M_s and M_t ; n is the size of text block.

In general, a fitness function must be maximised against the objective function of the problem, which is minimising the cost function given by equation (2). In our case, we denote that the fitness function corresponds to the cost function and may be minimized; its value can achieve an overall minimum 0 when S=n which denotes that k_t and k_s are equivalents.

5.3 Key evaluation

Given a initial population of candidate decryption keys, each of which represents a basic solution of the considered problem. A key is used to produce blocks of plaintexts. The evaluation is accomplished for each plaintext block independently of other blocks. A random Initial population of candidate keys is chosen. Table 1 illustrates an example of a population of five keys of SDES cryptosystem and the fitness values of the correspondent plaintexts M_t (t=1..5). The evaluation is built upon a given M_s where the correspondent decryption key is k_s = ‘1101010100’.

											$f^{\,t}_{\,k}$
k_1	1	0	0	1	1	0	0	1	1	1	0.625
k_2	1	0	0	1	1	0	1	0	1	1	0.875
k_3	0	0	1	1	1	0	1	0	1	1	1.0
k_4	0	1	1	0	0	1	0	1	1	1	0.75
k_5	1	1	0	1	0	0	1	0	1	0	0.5

Table 1. Example of candidate keys evaluation.

5.4 Attack principle

It consists on the following tasks that will be performed at every iteration of the evolution process.

- Compute the amount and average of ‘0’ and ‘1’ separately for keys which have a fitness value $f^t_k \leq \alpha$, where $\alpha \in]0,1]$, a given threshold.
- Generate a referential key k^* built upon statistical average values of ‘0’ and ‘1’ obtained above. Each bit of k* is normalised as 1 if the average of ‘1’ is greater than the average of ‘0’ and vice versa. A random bit value is retained is case of equality.
- Proceed to the crossover which concern just keys with fitness value less than α. The crossover occurs for each two keys selected based on a fitness proportionate strategy. With a single crossover point, each two keys produce a child.
- The mutation is accomplished with a standard rate (2 to 5%) for each selected key separately.

Then, keys that exhibit acceptable result are injected in next generation, whereas other keys will be removed.

Table 2 shows an example of the evolution process; with a given α=0.75, the retained keys of table 1 are k_1, k_4 and k_5. The crossover is accomplished between k_1 and k_4 with a single crossover point; the produced child is $k_{1\text{-}4}$. The retained keys are k_1, k_4, k_5 , $k_{1\text{-}4}$ and k* which constitute the population of the next generation. The (*) symbol in the last row of table 2 means that the corresponding bit value is randomly chosen.

											f^t_k
k_1	1	0	0	1	1	0	0	1	1	1	0.625
k_4	0	1	1	0	0	1	0	1	1	1	0.75
$k_{1\text{-}4}$	1	0	0	1	1	1	0	1	1	1	0.5
k_5	1	1	0	1	0	0	1	0	1	0	0.5
Σ ‘1’	3	2	1	3	2	2	1	3	4	3	
Σ ‘0’	1	2	3	1	2	2	3	1	0	1	
Avg ‘1’	0.75	0.5	0.25	0.75	0.5	0.5	0.25	0.75	1.0	0.75	
Avg ‘0’	0.25	0.5	0.75	0.25	0.5	0.5	0.75	0.25	0.0	0.25	
k*	1	1*	0	1	0*	0*	0	1	1	1	0.375

Table 2. Construction of a referential key k*

5.5 Implementation

The attack consists on applying the encryption algorithm for each candidate key k_i. The function 1 outlines the main steps of attack process of DES ; it takes in input, a block cipher C_s of size n and a decryption key k_i and produces a plaintext M_t. The variables L, R, E, S, P and IP denoted respectively the left and right halves of block cipher, the expansion, substitution, permutation and the initial permutation achieved by the encryption algorithm. The LX and T are temporary arrays.

```
Function 1. Decrypt(Cs, n, k)
structure L[n/2], LX[n/2], R[n/2], T[n]
Sboxe ← Sboxe_DES;; IP ← IP_DES ; IP^-1 ← IP^-1_DES; E ← E_DES; S ← S_DES
IP(Cs)
for i ← 1 to n do
    L[i] ← Cs[i];, R[i] ← Cs[n/2+i]
endfor
for i ← 1 to nbRound do
    Generate (k_i)
    T ← E(R) ; T ← T ⊕ k_i ; LX ← S(T,Sboxe)
    LX ← P(LX) ; L ← R ; R ← L ⊕ LX
endfor
for i ← 1 to n/2 do
    Mt[i] ← L[i]; Mt[n/2+i] ← R[i]
endfor
IP^-1(Mt)
return(Mt)
```

Each output block M_t of size n is evaluated according to the fitness function illustrates by equation (2) and outlined by function 2.

```
Function 2. Cost(Mt, Ms)
Cost ← 0
for i ← 1 to n do
    if Mt[i] = Ms[i] Cost ← +1 endif
endfor
Cost ← 1 Cost/n
```

At every iteration, a referential key k^* is generated based on the keys k_t which have the best performance ($f^t_k \leq \alpha$). The process of generation is illustrated by function 3.

```
Function 3. RefKey()
structure avg[n,2], Refkey<bit[n],fitness>
for j ← 1 to n do
    avg[j,1] ← Sum(k[i].bit[j] for all i and bit[j]='1'
    avg[j,2] ← Sum(k[i].bit[j] or all i and bit[j]='0'
endfor
for j ← 1 to n do
    Refkey.bit[j] ← max(avg[j,1],avg[j,2] )
    if (avg[j,1]=avg[j,2])
        Refkey.bit[j] ← rnd(1)
    endif
endfor
Return(Refkey)
```

5.6 Attack with genetic algorithm

To apply Genetic Algorithm in cryptanalysis of block ciphers and, based on a population of *popSize* individuals, a set of keys k_i *(i=1..popSize)* are randomly generated in first. Each of which is modelled by a structure *key_struc<bit[n],fitness>*. Algorithm 1 outlines the main steps of the attack using GA; the *cross()* function creates a key k_{ij} based on a first part of a key k_i and a second part of a key k_j. Both k_i and k_j have simultaneously a fitness value less than α. k_{ij} will replaces k_p, a chosen key among whose have a fitness greater than α. The mutation *inv()* is merely an inversion of a key bit based on a given rate. Each key k_j is then used to decrypt the proposed ciphertext C_s. Each plaintext M^t_i *(i=1..popSize)* is evaluated; its fitness is denoted by k_i*.fitness*. Then, the referential key k^* should be generated; it will replaces the worst one.

Algorithm 1. AG in attack of block ciphers

```
input: Ms, Cs, key_struc<bit[n],fitness>,n,
        k_popSize, seg_nb, mut_rate, alpha
output: k*, Mt_k*
generate k_i (i=1..popSize)
for i ← 1 to popSize do
    Mt_i ← decrypt(Cs,n,k_i)
    k_i .fitness ← cost(Mt_i , Ms)
endfor
sort(k_i , fitness) (i=1..popSize)
while not <exit criterion>
  i ← 1, j ← popSize
  while i < j-1 and k_{i+1}.fitness≤alpha do
      cross(k_i, k_{i+1}, k_j, alpha)
      i ← i+1, j ← j-1
  endwhile
  for i ← 1 to popSize do
      k_i ← inv(k_i , bit_x, mut_rate)
      k_i .fitness ← cost(decrypt(Cs,n,k_i ) , Ms)
  endfor
  sort(k_i , fitness) (i=1..popSize)
  k_popSize ← refKey()
  k* ← k_1
endwhile
return (k*,Mt_1)
```

The process will be stopped after a fixed number of iterations or if no improvement in solution occurs after a fixed period.

Similar to all heuristic algorithms, GA is unable de reproduce an exact solution, that is, and in the algorithm 1, k* is rarely a right key but rather close to the right one. Hence, the produced plaintext can be partially readable and easily revised and corrected based on manual changes.

5.7 Attack with Particle Swarm Optimisation

Particle swarm optimisation (*PSO*, in short), is a population based method attributed to Kennedy and Eberhart [78], inspired by the social comportment of animals and insects that lives in communities and exhibit both individual and social behaviour. Unlike GA, PSO has no genetic operators but a swarm of particles, which evolves through space search and exchange environmental information in order to identify promising regions, according to an effective strategy, which consists to follow particles with best positions in regard of the food source.

The PSO algorithm uses a population of particles denoted by their positions (in the search space) which kept randomly in first, and represent potential solutions. Through exploration, each particle *i* maintains its best position x_i it ever encountered by moving toward the position x_p of its best neighbourhood [79] and the position x_g of the best particle of the swarm with a moderate velocity v_i according to the following equations:

$$v_i=c_iv_i+c_p(p_i-x_i)+c_g(g-x_i) \quad (3)$$

$$x_i=x_i+v_i \quad (4)$$

where p_i and g are respectively the position of the best neighbourhood of *i* and the position of the best particle of swarm. c_i, c_p and c_g are random numbers (called *learning factors*) uniformly distributed in range [0,1].

Since its inception, PSO gained popularity due to its simplicity and effectiveness in production of good results with low cost and has been object of several improvements and variants [80][81][82] [83] [84][85]. It has been also successfully applied in solving various complex combinatorial problems [86][87][88][89][90][91]. In cryptanalysis PSO has been widely used for attack of various cryptosystems in classical and modern ciphers [92][93][94].

Algorithm 2 describes the main steps of PSO applied to cryptanalysis of block ciphers; each particle *i* corresponds to a key k_i with *keysize* bits length.

At every iteration of the decryption process, each bit *j* of the key k_i is updated according to the following rule: a flip to 1 (if $v_i > \alpha$) and to 0 otherwise; the new position x_i of particle *i* correspond to the key *i* with an updated bit *j*. Both v_i and α may be in range

]0,1]. The obtained keys are then valued according to the fitness function used. Only the best keys may survive. The process will continue until it reaches an acceptable plaintext or after a fixed number of iterations. For simplicity, we omit the near neighbourhood from the process.

Algorithm 2. PSO in attack of block ciphers

```
input: Cipher$_n$, SwarmSize, keySize, $\alpha$
output: $S^*$, $k^*$
    Generate $k_i$, $v_i$, $c_i$ (i=1.. SwarmSize) and $c_g$
    Evaluate $S_i$, (i=1.. SwarmSize)
    $S^* \leftarrow \min(S_i$, i=1.. SwarmSize)
while not (exit criterion)
    for $j \leftarrow 1$ to KeySize do
      for $i \leftarrow 1$ to SwarmSize do
         Pick random number $c_i \in \{0,1\}$
         $v_i \leftarrow c_i v_i + c_g | k_i.bit_j - k^*.bit_j |$
         if $v_i > \alpha$
           $k_i.bit_j \leftarrow 1$ else $k_i.bit_j \leftarrow 0$
         endif
         if $S_i < S^*$
            $S^* \leftarrow S_i$ ; $k^* \leftarrow k_i$
         endif
      endfor
    endfor
endwhile
return ($k^*$)
```

6 Experimentation and results

In this section, various experiments were conducted on GA and other heuristics applied to cryptanalysis of used cryptosystems. The aim of experiments is to emphasize the effectiveness of the considered approach in the resolution of such problem.

6.1 Experimental settings

The experiments have been conducted on a set of sample binary texts in range of 800 to 16000 bits (100 to 2000 alphabet characters) extracts from ICE [95] and converted to upper case letters without spaces. Longer texts are generally easy to decipher. Moreover, the cryptanalysis is usually applied to fragments of data and messages exchanged through networks which are in most cases, processed by small blocks. Encryption algorithms used are: Simplified Data Encryption Standard (SDES) [96], Data Encryption Standard reduced to 8 rounds (DES-8), FEAL-8 [23] and eight rounds of RC-5 [22]. Each key is assimilated as a binary vector of 64 bits (case of DES-8 and FEAL-8), 40 bits in case of RC-5 and 10 bits in case of SDES. The used algorithms are coded on Matlab 2.14 and performed on a CPU 3.2 Ghz. For genetic operators' parameters, we have set a single crossover point and a standard mutation rate of 4%. Numerical results were averaged over 10 runs of each test.

Tests were accomplished in three steps: The first, illustrated by figures 2, 3 and 4 below is intended to locate the optimal values of most common AG parameters such population size and the number of generations. The second, shown in figures 5 and 6, exhibit the performance of the proposed approach where the reference key k* is introduced; whereas, figures 7 and 8 illustrate a comparative study of the proposed approach against PSO and brute force attack.

6.2 Performance of algorithms

The first experiment refers to the impact of varying the size of the GA population with the performance of encryption algorithms in terms of recovered bits within the decryption key. The figure 2 shows the average percentage of correct bits recovered by GA for various population sizes. It turns that the performance increases with population size and becomes stable around a threshold of 60 individuals. Also, AG performs better for SDES algorithm (due to the shortest of its key) while DES-8 seems the hardest one among others.

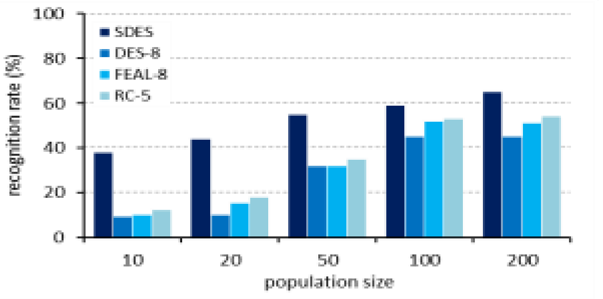

Fig. 2. Performance evaluation of recognition rate of correct key bits vs population size

With a selected population of 60 individuals and a ciphertext of 2400 bits (300 ASCII characters), the bar chart in figure 3 gives a comparison information about the recognition rate of recovered bits with the number of generations. It turns out that, the performance increases with the number of generations; the growth rate becomes constant beyond 3000. As, results of figure 2, AG outperforms with SDES algorithm and allows an acceptable results for RC-5.

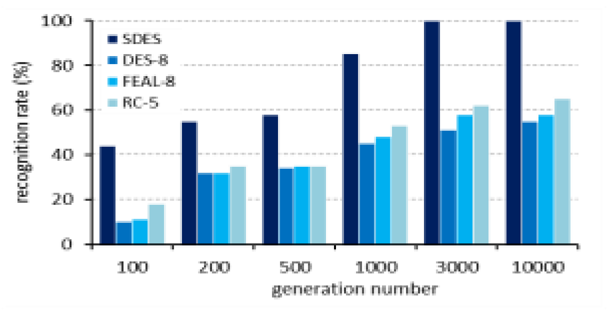

Fig. 3. Performance evaluation of recognition rate of correct key bits vs generation number

The experiment in figure 4 shows the effect of varying the ciphertext size with the performance of AG in terms of the number of recovered bits within the decryption key. The figure shows that is no improvement for small ciphertexts (less than 2000 bits) and noticeable results for texts of more than 2400 bits (300 ASCII characters). In addition, the result seems more significant for SDES algorithm where the complete key bits are recovered. Overall and, except DES-8, AG performs more than 50% for all used algorithms.

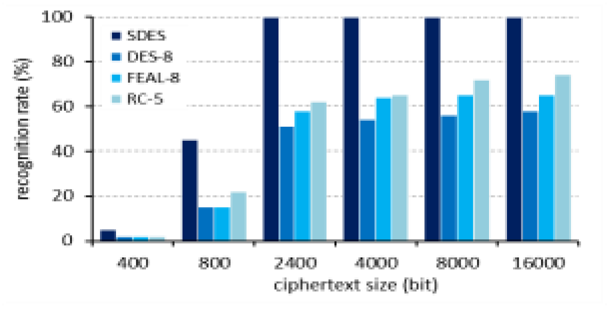

Fig. 4. Performance evaluation of recognition rate of correct key bits vs ciphertext size

6.3 Evaluation of AG_{k^*} approach

In following experiments, we introduce the reference key k^* in AG generations and, based on previous results, we adopted the best parameters values, namely: a population of 60 individuals, a processing execution of 3000 generations and various ciphertexts of 2400 bits each.

As mentioned in §3.3, k^* is built upon generation keys which satisfied a fitness value less than a threshold α. The best value of α is located based on experiment illustrated by figure 5 which shows the performance of recognition rate of correct bits produced by AG_{k^*} (AG with k*) with the variation of α.

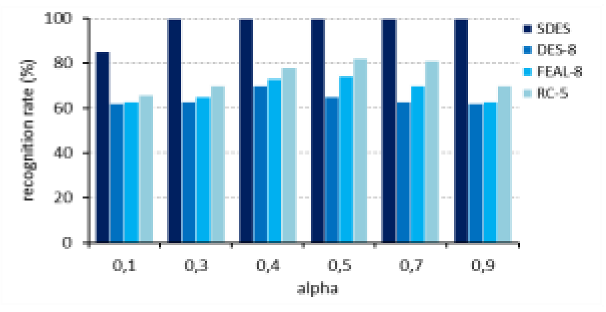

Fig. 5. Performance evaluation of recognition rate of correct key bits vs alpha

In this figure, we note that the best performance is obtained when using a threshold value around 0.6. The best performance is achieved for SDES algorithm (with $\alpha > 0.2$), for RC-5 cryptosystem ($0.8 > \alpha > 0.4$) and in range [0.5, 0.6] for other algorithms.
In following experiments, we opt for $\alpha = 0.6$.

The bar chart in figure 6 illustrated the variation of CPU time for both AG and AG_{k^*} when using optimal parameters values defined above with a duration that corresponds to 3000 generations. It is noticeable that AG uses less processing time than AG_{k^*}. This difference is justified by the extra time spent in computing of sums and averages of key bits needed in building of k*. Overall, DES-8 and FEAL-8 cryptosystems seem the greediest in time consumption.

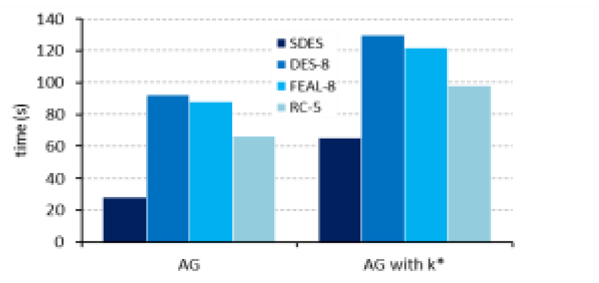

Fig. 6. Performance evaluation of recognition rate of correct key bits vs ciphertext size

The experiments below are not intended to enhance the performance of bio-inspired heuristics but to measure the effectiveness of proposed approach against other similar algorithms. Tests were operated based on the best parameters values given by previous experiments. Figure 7 gives a comparison performance of AG, AG_{k^*}, Brute force attack and PSO with the recognition rate of correct key bits when using a processing duration that corresponds to 5000 generations.

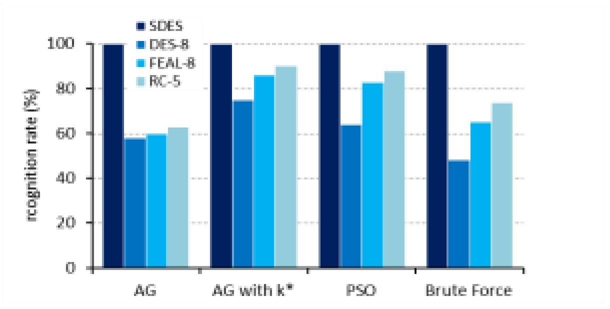

Fig. 7. Performance evaluation of recognition rate of correct key bits vs alpha

The figure 7 shows that the proposed approach outperforms significantly other algorithms and seems competitive with PSO. We note also that the SDES key is fully broken and results related toDES-8, FEAL-8 and RC-5 are acceptable (with more than 70% of recovered bits).

Within the same parameters, the figure 8 gives information about the processing CPU time used by AG, AG_{k^*}, PSO and brute force attack on ciphertexts produced by SDES, DES-8, FEAL-8 and RC-5. It's easy to notice that AG_{k^*} and, as PSO needs more processing time than AG and brute force. Based on this result, it appears that the GA_{k^*} uses more time than other algorithms, but this delay can be improved by the well performance of the approach as mentioned in figure 6 above.

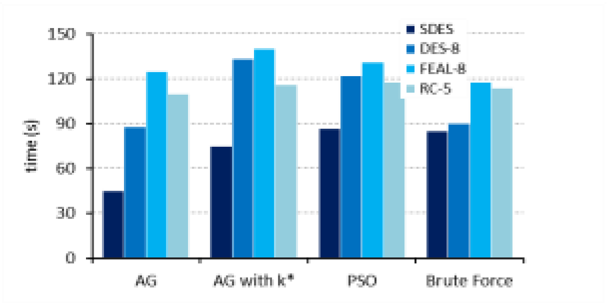

Fig. 8. Performance evaluation of recognition rate of correct key bits vs alpha

7 Discussion

In literature, the above cryptosystems have been objects of various attacks. FEAL-8 cryptosystem is announced to be broken with less than 10000 chosen plaintexts [97] [98] and by 2^{25} known plaintexts with a success rate of 70% [99]. These results have been reduced to 2^{15} known plaintexts [100] and finally to less than 2000 pairs of chosen plaintexts [101]. The RC-5 cryptosystem with reduced rounds (8 and 12) is also declared be broken with 2^{48} chosen plaintexts [102] and less than 2^{44} chosen plaintexts [103]. The most attacks were based on linear, differential, timing and correlation cryptanalysis and are outdated; this fact prevents any way of comparison since our tests uses computational intelligence heuristics. In case of DES cryptosystem variants, the most attacks as mentioned in §4.1 approve the fact that SDES and DES-n (with $n<6$) is breakable whereas is not the case for other cryptosystems such DES-8 where the best success rate is less than 65%. So, attacks tend in general, to reduce the key space instead of locating correct bits. However and, with an average of 70% of recovered bits, our strategy conquers existent results with its efficiency in resolution of such problem.

8 Conclusion

Bio-inspired intelligence algorithms, an active area in artificial intelligence, denoted as a successful research methodology in resolution of complex real-life problems such cryptanalysis with moderate resources consumption.

The paper outlines the main concepts of such methodology in cryptanalysis of block ciphers based cryptosystems and focuses on Genetic Algorithm, a most popular bio-inspired intelligence technique. Also, we proposed a new approach based on genetic algorithm for attack of such ciphers. It consists for each generation, and in addition of usual genetic operators, to build a referential key based on the performance of certain other keys, it will replaces the worst one in subsequent generation. This strategy allows a fast convergence and keeps permanently a high quality of results.

The experiments conducted indicate that the proposed methodology can be successfully applied as a powerful tool in handling such problem. The produced results obtained based on various typical instances, allow locating more than 60% of correct bits-key with acceptable resource consumption for some variants of block ciphers cryptosystems such

FEAL-8, RC-5 and DES eight rounds while SDES keys were fully broken.

The proposed algorithm have not yet been explored to find its full capabilities; that is, tests were operated on a reduced space of data, however, the approach presented can be adjusted in order to achieve other cryptosystems and results can also be extended to more data and improved by the well choice of environment parameters.

Based on the literature related to cryptanalysis problem, we notice that, the construction of the fitness function represents a critical determinant of the results quality, since the various forms of functions used in classical cryptanalysis cannot be used as evaluation tool for modern ciphers due to the nonlinearity of block ciphers. Although and, as a future direction, the proposed approach may open possibilities in investigation of further complicated cryptosystems attacks.

References:

[1] W. Stallings, Cryptography and Network Security, *Netw. Secur.*, pp. 66–74, 2006.

[2] A. J. Menezes, P. C. Van Oorschot, and S. a. Vanstone, Handbook of Applied Cryptography, *Electr. Eng.*, vol. 106, p. 780, 1997.

[3] K. V. S. Rao, M. R. Krishna, and D. Bujji Babu, Cryptanalysis of a Feistel Type Block Cipher by Feed Forward Neural Network Using Right Sigmoidal Signals, *Int. J. Soft Comput.*, vol. 4, no. 3, pp. 131–135, 2009.

[4] F. L. Bauer, *Decrypted Secrets: Methods and Maxims of Cryptology*. Heidelberg: Springer-Verlag, 1997.

[5] E. Biham and A. Shamir, *Diifferential Cryptanalysis of the Data Encryption Standard*. Springer-Verlag, 1993.

[6] E. Biham and A. Shamir, Differential Cryptanalysis of the Full 16-round DES, in *Advances in Cryptology — CRYPTO' 92*, 2001, pp. 487–496.

[7] E. Biham and A. Shamir, Differential cryptanalysis of DES-like cryptosystems, *J. Cryptol.*, vol. 4, no. 1, pp. 3–72, 1991.

[8] M. Matsui and Y. A, new method for known plaintext attack of feal cipher, *Lect. Notes Comput. Sci.*, pp. 81–91, 1992.

[9] M. Matsui, The first experimental cryptanalysis of the data encryption standard, in *14th Annual International Cryptology Conference*, 1994, pp. 1–11.

[10] M. Matsui, Linear Cryptanalysis Method for DES Cipher, *LNCS*, vol. 765, pp. 386–397, 1994.

[11] A. Biryukov and D. Wagner, Advances in Cryptology — EUROCRYPT 2000, in *Lecture Notes in Computer Sciences*, Springer Berlin Heidelberg, 2000, pp. 589–606.

[12] A. H. Gandomi and A. H. Alavi, Multi-stage genetic programming: A new strategy to nonlinear system modeling, *Inf. Sci. (Ny).*, vol. 181, no. 23, pp. 5227–5239, 2011.

[13] C. Blum and X. Li, Swarm Intelligence in Optimisation, *Swarm Intell. Introd. Appl.*, pp. 43–85, 2008.

[14] T. S. C. Felix and K. T. Manoj, Swarm Intelligence, Focus on Ant and Particle Swarm Optimization, in *Numerical Analysis and Scientific Computing*, T. S. C. Felix and K. T. Manoj, Eds. I-Tech Education and Publishing, 2007.

[15] T. M. R. Sharvani.G.S, N.K. Cauvery, Different Types of Swarm Intelligence Algorithm for Routing, in *2009 International Conference on Advances in Recent Technologies in Communication and Computing*, 2009, pp. 604–609.

[16] H. Feistel, Cryptography and Computer Privacy, *Scientific American*, vol. 228, no. 5. pp. 15–23, 1973.

[17] B. Schneier, The Blowfish Encryption Algorithm, *Dr. Dobbs J.*, vol. 23, pp. 38–40, 1998.

[18] H. Feistel, Block cipher cryptographic system, #3 798 359, 1974.

[19] A. Carlisle, Constructing of Symmetric ciphers using the CAST design Procedure," *Des. Codes, Cryptogr.*, vol. 12, pp. 283–316, 1997.

[20] National Bureau Of Standards, Data Encryption Standard (DES), *Technology*, vol. 46–3, no. 46, pp. 1–26, 1999.

[21] X. Lai and J. L. Massey, A proposal for a new block encryption standard, *Adv. Cryptology—EUROCRYPT'90*, pp. 389–404, 2006.

[22] R. L. Rivest, The RC5 Encryption Algorithm, *Technology*, vol. 1008, pp. 86–96, 1995.

[23] A. Shimizu and S. Miyaguchi, Fast Data Encipherment Algorithm FEAL, in *Advances in Cryptology — EUROCRYPT '87*, 1988, vol. 304, pp. 267–278.

[24] S. Heron, Advanced Encryption Standard (AES), *Netw. Secur.*, vol. 2009, no. 12, pp. 8–12, Dec. 2009.

[25] W. S. McCulloch and W. Pitts, A logical calculus of the ideas immanent in nervous

activity, *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, 1943.

[26] P. Tarasewich and P. R. McMullen, Swarm intelligence: power in numbers, *Commun. ACM*, vol. 45, no. August, pp. 62–67, 2002.

[27] E. Bonabeau, M. Dorigo, and G. Theraulaz, *From Natural to Artificial Swarm Intelligence*. Oxford University Press, 1999.

[28] L. M. Hiot, Y. S. Ong, B. K. Panigrahi, Y. Shi, M.-H. Lim, P. K. Tripathi, S. Bandyopadhyay, and S. K. Pal, *Handbook of Swarm Intelligence*, vol. 8. 2010.

[29] M. Dorigo, V. Maniezzo, and A. Colorni, Positive feedback as a search strategy, *Tech. Rep. 91-016*, 1991.

[30] K. M. Passino, Biomimicry of bacterial foraging for distributed optimization and control, *Control Systems, IEEE*, vol. 22, no. 3. pp. 52–67, 2002.

[31] D. F. Signorini and J. M. Slattery, Neural networks., *Lancet (London, England)*, vol. 346, no. 8988, p. 1500, Dec. 1995.

[32] S.-C. Chu, P.-W. Tsai, and J.-S. Pan, Cat swarm optimization, *PRICAI 2006 Trends Artif. Intell.*, pp. 854–858, 2006.

[33] M. Bakhouya and J. Gaber, An Immune Inspired-based Optimization Algorithm: Application to the Traveling Salesman Problem, vol. 9, no. 1, pp. 105–116, 2007.

[34] K. N. Krishnanand and D. Ghose, Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions, *Swarm Intell.*, vol. 3, no. 2, pp. 87–124, 2009.

[35] X. S. Yang, A new metaheuristic Bat-inspired Algorithm, *Stud. Comput. Intell.*, vol. 284, pp. 65–74, 2010.

[36] S. Mirjalili, S. M. Mirjalili, and A. Lewis, Grey Wolf Optimizer, *Adv. Eng. Softw.*, vol. 69, pp. 46–61, 2014.

[37] R. E. Perez and K. Behdinan, Particle swarm approach for structural design optimization, *Comput. Struct.*, vol. 85, no. 19–20, pp. 1579–1588, 2007.

[38] P. Pongchairerks, Particle swarm optimization algorithm applied to scheduling problems, *ScienceAsia*, vol. 35, no. 1, pp. 89–94, 2009.

[39] G. Hanrahan, Swarm intelligence metaheuristics for enhanced data analysis and optimization., *Analyst*, vol. 136, no. 18, pp. 3587–94, 2011.

[40] D. E. Goldberg and J. H. Holland, Genetic Algorithms and Machine Learning, *Mach. Learn.*, vol. 3, no. 2, pp. 95–99, 1988.

[41] A. P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*, vol. 8. Wiley, 2005.

[42] J. A. Clark, J. L. Jacob, and S. Stepney, The design of S-boxes by simulated annealing, *New Gener. Comput.*, vol. 23, no. 3, pp. 219–231, 2005.

[43] J. Olamaei, T. Niknam, and G. Gharehpetian, Application of particle swarm optimization for distribution feeder reconfiguration considering distributed generators, *Appl. Math. Comput.*, vol. 201, no. 1–2, pp. 575–586, 2008.

[44] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, vol. 1, no. 1. 1975.

[45] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[46] M. D. Vose and A. Hall, Modeling Simple Genetic Algorithms, *Evol. Comput.*, vol. 3, no. 4, pp. 453–472, 1996.

[47] J. H. HOLLAND, *Adaptation in natural and artificial systems.* MIT Press, 1992.

[48] L. J. Fogel, A. J. Owens, and M. J. Walsh, Artificial Intelligence through Simulated Evolution, *Wiley*, no. 1965, pp. 27–38, 1997.

[49] I. F. Gonos, N. E. Mastorakis, S. Member, M. N. S. Swamy, and L. Fellow, A Genetic Algorithm Approach to the Problem of Factorization of General Multidimensional Polynomials, vol. 50, no. 1, pp. 16–22, 2003.

[50] D. B. Fogel, Evolutionary algorithms in theory and practice, *Complexity*, vol. 2, no. 4, pp. 26–27, Mar. 1997.

[51] C. W. Ahn, *Practical genetic algorithms*, vol. 18. 2006.

[52] Z. Michalewicz, *Genetic algorithms + data structures = evolution programs (3rd ed.)*, vol. 1. 1996.

[53] K. F. Man, K. S. Tang, and S. Kwong, Genetic algorithms: Concepts and applications, *IEEE Trans. Ind. Electron.*, vol. 43, no. 5, pp. 519–534, 1996.

[54] R. Matthews, The use of genetic algorithms in cryptanalysis, *Cryptologia*, 1993.

[55] H. M. H. Husei, B. I. Bayoumi, F. S. Holail, B. E. M. Hasan, and M. Z. Abd El-Mageed, A genetic algorithm for cryptanalysis with application to DES-like systems, *Int. J. Netw. Secur.*, vol. 8, no. 2, pp. 177–188, 2009.

[56] A. G. Bafghi and B. Sadeghiyan, Finding suitable differential characteristics for block ciphers with Ant colony technique, in *Ninth International Symposium on Computers and Communications, 2004. Proceedings. ISCC 2004*, 2004, vol. 1, pp. 418–423 Vol.1.

[57] E. C. Laskari, G. C. Meletiou, Y. C. Stamatiou, and M. N. Vrahatis, Cryptography and cryptanalysis through computational intelligence, *Stud. Comput. Intell.*, vol. 57, pp. 1–49, 2007.

[58] E. C. Laskari, G. C. Meletiou, Y. C. Stamatiou, and M. N. Vrahatis, Applying evolutionary computation methods for the cryptanalysis of Feistel ciphers, *Appl. Math. Comput.*, vol. 184, no. 1, pp. 63–72, 2007.

[59] E. C. Laskari, G. C. Meletiou, Y. C. Stamatiou, and M. N. Vrahatis, Evolutionary computation based cryptanalysis: A first study, *Nonlinear Anal. Methods Appl.*, vol. 63, no. 5–7, pp. E823–E830, 2005.

[60] J. Song, H. Zhang, Q. Meng, and Z. Wang, Cryptanalysis of four-round des based on genetic algorithm, *2007 Int. Conf. Wirel. Commun. Netw. Mob. Comput. WiCOM 2007*, pp. 2326–2329, 2007.

[61] N. Nalini and G. Raghavendra Rao, Attacks of simple block ciphers via efficient heuristics, *Inf. Sci. (Ny).*, vol. 177, pp. 2553–2569, 2007.

[62] P. Garg, S. Varshney, and M. Bhardwaj, Cryptanalysis of Simplified Data Encryption Standard using Genetic Algorithm, *Am. J. Networks Commun.*, vol. 4, no. 3, pp. 32–36, 2015.

[63] Y. Fan, S. Jun, and Z. Huanguo, Quantitative cryptanalysis of six-round DES using Evolutionary computation, in *Third Internationa Symposium, ISICA*, 2008, pp. 134–141.

[64] W. Shahzad, A. B. Siddiqui, and F. A. Khan, Cryptanalysis of Four-Rounded DES using Binary Particle Swarm Optimization, *Simulation*, pp. 1757–1758, 2009.

[65] R. Vimalathithan and M. L. Valarmathi, Cryptanalysis of Simplified-DES using Computational Intelligence, *WSEAS Trans. Comput.*, vol. 10, no. 7, pp. 210–219, 2011.

[66] S. Pandey and P. M. Mishra, Particle Swarm Optimization in Cryptanalysis of DES, *Int. J. Adv. Res. Comput. Eng. Technol.*, vol. 1, no. 4, pp. 379–381, 2012.

[67] R. Vimalathithan and M. Valarmathi, Cryptanalysis of Simplified-AES using Particle Swarm Optimisation, *Def. Sci. J.*, vol. 62, no. 2, pp. 117–121, 2012.

[68] S. A. A. Hamdani, S. Shafiq, and Farrukh Aslam Khan, Cryptanalysis of Four-Rounded DES Using Binary Artificial Immune System, in *Lecture Notes in Computer Scienc*, Y. T. Ytan, Y. Sh, and T. Kay Chen, Eds. Springer Berlin Heidelberg, 2010, pp. 338–346.

[69] S. Khan, W. Shahzad, and F. A. Khan, Cryptanalysis of four-rounded DES using ant colony optimization, *2010 Int. Conf. Inf. Sci. Appl. ICISA 2010*, pp. 2161–2166, 2010.

[70] W. G. Abd-Elmonim, N. I. Ghali, A. E. Hassanien, and A. Abraham, Known-plaintext attack of DES-16 using particle swarm optimization, *Proc. 2011 3rd World Congr. Nat. Biol. Inspired Comput. NaBIC 2011*, pp. 12–16, 2011.

[71] S. S. Jadon, H. Sharma, E. Kumar, and J. C. Bansal, Application of binary particle swarm optimization in cryptanalysis of DES, *Adv. Intell. Soft Comput.*, vol. 130 AISC, no. VOL. 1, pp. 1061–1071, 2012.

[72] R. Vimalathithan and M. Valarmathi, Cryptanalysis of DES using Computational Intelligence, *WSEAS Trans. Comput.*, vol. 55, no. 2, pp. 237–244, 2011.

[73] F. Teytaud and C. Fonlupt, A Critical Reassessment of Evolutionary Algorithms on the cryptanalysis of the simplified data encryption standard algorithm, *arXiv Prepr. arXiv1407.1993*, p. 12, 2014.

[74] C. De Canniere, A. Biryukov, and B. Preneel, An introduction to Block Cipher Cryptanalysis, *Proc. IEEE*, vol. 94, no. 2, pp. 346–356, 2006.

[75] D. Coppersmith, The Data Encryption Standard (DES) and its strength against attacks, *IBM Journal of Research and Development*, vol. 38, no. 3. pp. 243–250, 1994.

[76] P. Isasi and J. C. Hernandez, Introduction to the Applications of Evolutionary Computation in Computer Security and Cryptography, *Comput. Intell.*, vol. 20, no. 3, pp. 445–449, Aug. 2004.

[77] D. G. N. Hunter and A. R. McKenzie, Experiments with Relaxation Algorithms for Breaking Simple Substitution Ciphers, *Comput. J.*, vol. 26, no. 1, pp. 68–71, 1983.

[78] J. Kennedy and R. Eberhart, Particle swarm optimization, *Proc. ICNN'95 - Int. Conf. Neural Networks*, vol. 4, pp. 1942–1948, 1995.

[79] R. . Eberhart, P. Simpson, and R. Dobbins, *Computational Intelligence PC Tools*. Academic Press, 1996.

[80] Y. Shi and R. C. Eberhart, Empirical study of particle swarm optimization, *Proc. 1999 Congr. Evol. Comput.*, pp. 1945–1950, 1999.

[81] M. Zambrano-Bigiarini, M. Clerc, and R. Rojas, Standard Particle Swarm Optimisation 2011 at CEC-2013: A baseline for future PSO improvements, in *2013 IEEE Congress on Evolutionary Computation*, 2013, pp. 2337–2344.

[82] D. Bratton and T. Blackwell, A Simplified Recombinant PSO, *J. Artif. Evol. Appl.*, vol. 2008, no. 1, pp. 1–10, 2008.

[83] M. Meissner, M. Schmuker, and G. Schneider, Optimized Particle Swarm Optimization (OPSO) and its application to artificial neural network training., *BMC Bioinformatics*, vol. 7, p. 125, 2006.

[84] W. J. Zhang and X. F. Xie, DEPSO: Hybrid Particle Swarm with Differential Evolution Operators, *Proc. 2003 IEEE Int. Conf. Syst. Man Cybern.*, vol. 4, no. 1, pp. 3816–3821, 2003.

[85] Z. H. Zhan, J. Zhang, Y. Li, and H. S. H. Chung, Adaptive Particle Swarm Optimization, *IEEE Trans. Syst. Man Cybern. Part B-Cybernetics*, vol. 39, no. 6, pp. 1362–1381, 2009.

[86] J. Salerno, Using the particle swarm optimization technique to train a recurrent neural model, in *Proceedings Ninth IEEE International Conference on Tools with Artificial Intelligence*, 1997, pp. 45–49.

[87] J. Kennedy, R. C. Eberhart, and Y. Shi, Swarm Intelligence, *Swarm Intell.*, pp. 287–325, 2001.

[88] E. H. Luna, C. A. C. Coello, and A. H. Aguirre, On the use of a population-based particle swarm optimizer to design combinational logic circuits, in *2004 NASA/DoD Conference on Evolvable Hardware, 2004. Proceedings*, 2004, pp. 183–190.

[89] X. Hu, R. C. Eberhart, and Y. Shi, Engineering optimization with particle swarm, *Swarm Intelligence Symposium, 2003. SIS '03. Proceedings of the 2003 IEEE*. pp. 53–57, 2003.

[90] H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, and Y. Nakanishi, A particle swarm optimization for reactive power and voltage control considering voltage security assessment, *IEEE Trans. Power Syst.*, vol. 15, no. 4, pp. 1232–1239, 2000.

[91] J. Robinson and Y. Rahmat-Samii, Particle swarm optimization in electromagnetics, *Antennas Propagation, IEEE Trans.*, vol. 52, no. 2, pp. 397–407, 2004.

[92] M. F. Uddin and a. M. Youssef, Cryptanalysis of Simple Substitution Ciphers Using Particle Swarm Optimization, *2006 IEEE Int. Conf. Evol. Comput.*, pp. 677–680, 2006.

[93] N. Nalini and G. Raghavendra Rao, Cryptanalysis of block ciphers via improvised particle swarm optimization and extended simulated annealing techniques, *Int. J. Netw. Secur.*, vol. 6, no. 3, pp. 342–353, 2008.

[94] K. Dworak and U. Boryczka, Cryptanalysis of SDES Using Modified Version of Binary Particle Swarm Optimization, in *Computational Collective Intelligence*, vol. 9330, Springer International Publishing, 2015, pp. 159–168.

[95] G. Nelson, S. Wallis, and B. Aarts, *Exploring Natural Language*, vol. G29. Amsterdam: John Benjamins Publishing Company, 2002.

[96] E. Schaefer, A Simplified {D}ata {E}ncryption {S}tandard Algorithm, *Cryptologia*, vol. 20, no. 1, pp. 77–84, 1996.

[97] B. den BOER, Cryptanalysis of {F.E.A.L.}, in *Advances in Cryptology. EUROCRYPT'88*, vol. 330.

[98] H. Gilbert and G. Chassé, A Statistical Attack of the FEAL-8 Cryptosystem, in *Advances in Cryptology-CRYPT0' 90*. Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1990, pp. 22–33.

[99] K. Ohta and K. Aoki, Linear cryptanalysis of the fast data encipherment algorithm, *Adv. Cryptology—Crypto'94*, pp. 12–16, 1994.

[100] M. Matsui and A. Yamagishi, A New Method for Known Plaintext Attack of FEAL Cipher, in *Advances in Cryptology — EUROCRYPT' 92*, 1993, vol. 658, pp. 81–91.

[101] E. Biham and A. Shamir, Differential Cryptanalysis of Feal and N-Hash, *547*, pp. 1–16, 1991.

[102] L. R. Knudsen and W. Meier, Improved Differential Attacks on RC5, in *Advances in Cryptology CRYPTO'96*, 1996, pp. 216–228.

[103] A. Biryukov and E. Kushilevitz, Improved cryptanalysis of RC5, Finland, 1998, pp. 85–99.