# Design and Evaluation of Two Symmetrical Evolutionist-Based Ciphering Algorithms

**F. Omary[1], A. Tragha[2], A. Bellaachia[3], And A. Mouloudi[4]**

[1] University of Mohammed V, The Science Faculty, Rabat, Morocco
[2] University of Hassan II, The Science Faculty of Ben M'Sik, Casablanca, Morocco
[3] George Washington University, Washington, DC 20052, USA
[4] University of Ibnou Tofail, The science faculty, Kenitra, Morocco

**Summary**
In this paper, we present two symmetric evolutionist-based ciphering algorithms (SEC and SEC-EX) where the second is an extension of the first. Our evolutionist approach is modeled as a combinatorial optimization problem. The paper addresses the coding of the chromosomes, the identification of a fitness function, and the choice of the adapted genetic operators. In SEC we treat directly the characters of plaintext. While in SEC-EX we develop a new technique, which bars the way in front of all the statistical analysis attacks. In fact, instead of treating characters we treat the binary blocks constituting the plaintext coded in binary. Both of the algorithms generate a resistant session encryption key. The performance of the two algorithms has been evaluated using several experiments. We have also compared our algorithms to the well-known ciphering algorithms such as DES, IDEA, and RSA. The experimental results show that our algorithms have the fastest deciphering time and an average ciphering.

***Key words:***
*Cryptography, symmetric ciphering, key, combinatorial optimization, evolutionist algorithm.*

## 1. Introduction

Cryptography "sciences of the secret" has been reserved strictly to diplomatic and military surroundings (for more than 3000 years). But with the advance of data processing and the evolution of the networks of communications, cryptography has become a vital process in all domains. Cryptography offers efficient solutions to protect sensitive information in a large number of applications including personal data security, medical records, network security, internet security, diplomatic and military communications security, etc.

Encryption is the process of converting a plaintext (original message) into cipher text (the coded message), which can be decoded back into the original message. An encryption (or ciphering) algorithm along with a key is used in the encryption and decryption of data. The degree of security offered by the algorithm depends on the type and length of the keys utilized [1]. An encryption algorithm can be either symmetric or asymmetric. In symmetric encryption a single key is used in both the encryption and the decryption of a message, see figure 1. In asymmetric encryption, the encryption key and the decryption key are different. The encryption process uses a public key to encrypt a message while the decryption process uses a private key to decrypt the message. In this paper, we present a new symmetric encryption algorithm, which allows us to obtain two different symmetric systems. The difference between these two systems is that the first treats directly the characters constituting the plaintext while the second treats binary blocks constituting the binary plaintext and having the same size. By the second algorithm (SEC-EX), we generalize the first algorithm (SEC) and reinforce his security against the statistical analysis attacks.

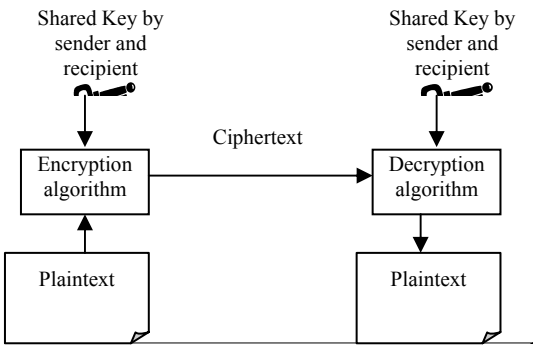


Fig. 1 Symmetric Encryption Algorithm.

True ciphering algorithms are counted on the tips of the fingers. Data Encryption Standard (DES) was developed in 1973. It is a symmetrical algorithm that uses a 64-bit secret key [2]. DES-based encryption makes encrypted documents vulnerable to attack due to the short length of its key. Advanced Encryption Standard (AES) was introduced in 1997. It was a replacement of DES. It is symmetrical and capable to support keys of lengths equal to 128, 192 and 256 bits [2]. International Data Encryption Algorithm (IDEA) is very recent. It is symmetrical and capable to support key of 128 bits [3].

The RSA algorithm is named after Ron Rivest, Adi Shamir and Len Adleman, who invented it in 1977. It is asymmetric and based on the modular arithmetic to define its public and private key [4]. This algorithm always

Manuscript received February 5, 2007
Manuscript revised February 25, 2007

resists to the cryptanalysis. Pretty Good Privacy (PGP) is a combination of the best functionalities of the asymmetric and conventional cryptography [5]. PGP is a hybrid system; it is currently, the dominant crypto-system.
Genetic algorithms are used to mimic natural processes of evolution. They have also emerged as practical, robust optimization and search methods. They are based on the natural selection principle defined by Charles Darwin (1809-1882): "The fittest individuals have more chances to survive into the next generation". J.Holland has introduced them in 1975 [6].
In this paper, we introduce a new encryption algorithm using genetic algorithm approach. The only related work is the attack of the asymmetric ciphering "Knapsack Cipher" [7]. This is inspired by the resolution of back bag problem. Thus efficiency genetic algorithms have been proven in cryptanalysis. In our first system SEC, we modeled the problem of ciphering a message M as a combinatorial optimization problem. Then we present a genetic solution based on the method used to solve the traveling salesman problem (TSP) [8]. In the second system SEC-EX, for scrambling plaintext, we introduce a new technique, which consists to encode plaintext in binary, chooses randomly an integer k and cuts plaintext into blocks of size k. These blocks are treated in the same way that the characters constituting the plaintext in SEC. Thus we avoid using classical cryptography for masking characters and reinforcing the security of our system.
The paper is organized as follows: The next section describes our encryption algorithm using genetic approach. Section 3 describes the extension of SEC, named SEC-EX. Experimental results and discussions are presented in sections 4 and 5. We end by conclusion and future directions in section 6.

## 2. Description of SEC

Let $M_0$ be the message to encode. $M_0$ consists of n characters. It can solely consist of numbers as in bank accounts or credit card numbers. It can also consist of a mixture of alphanumeric and punctuation characters.
We first apply a set of transformations to $M_0$. These transformations are a combination of several simple methods such as substitutions, permutations, affine ciphering, etc. The transformed message is denoted by M

### 2.1 Problem Definition

Let $c_1, c_2,\ldots, c_m$ be the different characters of M ($m \leq 256$). Denote by $L_i$ ($1 \leq i \leq m$) the list of the different positions of the character $c_i$ in M before the ciphering and by card($L_i$) the number of the occurrences of $c_i$ in M. We notice that $L_i \cap L_j$ is empty, $\forall i, j \in [1,m]$.

The vector below can represent the message M:

| | | | |
|---|---|---|---|
| $(c_1,L_1)$ | $(c_2,L_2)$ | … | $(c_m,L_m)$ |

The objective of SEC is to modify to the maximum the events of the characters in M. So, the new list $L_i$' affected to the character $c_i$ is such that the difference of card($L_i$) and card($L_i$') is maximal. This becomes a combinatorial optimization problem that can be solved using evolutionist algorithms.
The steps of the general algorithm are:
**Step 0:** Define a coding schema for individuals
**Step 1:** Create an initial population $P_0$ of q individuals $\{X_1,X_2,\ldots,X_q\}$.
Set i=0;
**Step 2**: Assessment of the individuals
Let F be the fitness function. Calculate $F(X_i)$ for every individual $X_i$ of $P_i$
**Step3:** // Selection
Select the best individuals (in the sense of F) and group them by pairs.
**Step4:** Application of the genetic operators

1. **Crossover**: Apply the crossover operation to the selected pairs.
2. **Mutation:** Apply the mutation to the individuals derived from the crossover.

Arrange the new individuals obtained from (1 and 2) in a new generation $P_{i+1}$.
Repeat the steps 2, 3 and 4 until the obtaining of the desirable performance level.
In the next subsection we explain implementation of the above steps in the context of SEC.

### 2.2 SEC Algorithm

**Step 0:** Coding Schema
An individual (or chromosome) is a vector of size m. In our case, the genes are represented by the $L_{pi}$ lists ($1 \leq i \leq m$). The $i^{th}$ gene, $L_{pi}$, contains the positions of character $c_i$.

**Step 1:** Creation of the initial populations $P_0$ made of q individuals: $X_1,X_2,..,X_q$.
Let Original-Ch be the list of chromosomes whose genes are $L_1,L_2,\ldots$, and $L_m$ lists (placed in this order). These lists represent the message before the application of the algorithm. We apply permutations on Original-Ch in order to get an initial population formed by *q* potential solutions of the problem.
Set i:=0;
**Step 2:** Fitness function
Let $X_j$ be an individual of $P_i$ whose genes are:

$L_{j1}$, $L_{j2}$...$L_{jm}$ . We define the fitness function on the set of the individuals $X_j$ by:

$$F(X_j) = \sum_{i=1}^{m} \left| card(L_{j_i}) - card(L_i) \right| \quad (1)$$

**Step 3:** Selection of the best individuals

We use the classical method of the roulette that allows keeping the strongest individuals [8]. Let us recall the process:

We assign to each individual $X_i$ a probability of apparition $p(X_i)$, (or relative strength) by:

$$p(X_i) = \frac{F(X_i)}{\sum_{k=1}^{q} F(X_k)} \quad (2)$$

An individual's selection is made as follows:

Let $q_i = \sum_{k=1}^{i} p(X_k)$ be the probability of accumulated apparition of individual $X_i$ and let $r$ be a number between 0 and 1, the successful individual is:

$$\begin{cases} X_1 & \text{if } q_1 \le r \quad \text{or} \\ X_i & \text{if } q_{i-1} < r \le q_i \quad \text{for } 2 \le i \le q \end{cases}$$

(q is the number of individuals in the population)

This process is repeated q times. With this principle, a strong individual can be selected several times. On the other hand a weak individual will be unlikely selected. In our process, we have also introduced a control function that is going to eliminate individuals with a small number of changes with respect to Original-Ch. Since the problem is reduced to a problem of permutations with constraints, we then apply the genetic operators adapted to this type of problems as explained in the next step.

**Step 4:** Genetic operators

1. MPX Crossover (Maximal Preservative X):

This crossover has been proposed by Mülhenbein [9] and [10] for the traveling salesman problem. This operator's idea is to insert a segment of parent's chromosome in the chromosome of the other parent so that the resulting crossover is closer to his parents. It is a two-point crossover. The two sons are obtained in a symmetrical manner. The following example illustrates this process:

Parent1: a b c d e f g h i j k l

Parent2: c b a g h i j l k f d e

The zone of crossover lies between the positions 5 and 9. First, we copy the crossover zone of parent1 into son1. Then, the son's gene that is not in the zone of crossover is completed in the following way:

- The $i^{th}$ gene of the parent2 is copied on the $i^{th}$ gene of son1 if this copy respects the constraints (doesn't create a disjointed tour).
- Otherwise, the $i^{th}$ gene of the parent1 is copied on the $i^{th}$ gene of son1 if it does not create any duplicates.
- If the two previous cases cannot apply, the $i^{th}$ gene of the son 1 receives a gene of the crossover zone of the parent 2 that respects the constraints (the first one is not taken).

Son 1 : c b a d e f g h i j k l

Son 2: a b c d h i j 1 k f e g

This crossover is applied to the selected individuals with a very precise rate. According to [11] and [12], the order of the best rate is from 60% to 100%.

2. Mutation of transposition:

We choose a mutation that randomly permutes two genes of a chromosome. This operator is applied to the individuals derived from crossover with an adapted rate preferably from 0.1% to 5% [11] and [12]. Then, we place the new offspring in a new population $P_{i+1}$.

Repeat the steps 2,3 and 4 until a stop criteria.

Define the stop condition:

The function F is bounded because $0 \le F(X) \le 2*n$ for each individual X. In fact:

$$\sum_{i=1}^{m} \left| card(L_{k_i}) - card(L_i) \right| \le \sum_{i=1}^{m} (card(L_{k_i}) + card(L_i))$$

$$\le \sum_{i=1}^{m} card(L_{k_i}) + \sum_{i=1}^{m} card(L_i) \le 2*n \quad (3)$$

Theoretically, the function F admits a maximum since it is bounded. According to some results [13], the convergence of fitness function is guaranteed, but perhaps towards a value close to max. This can be determined through experimental results.

**Final step of SEC:**

Let Final-Ch be the final solution given by SEC. The symmetrical key, also called genetic key, is generated using both the Original-Ch and Final-Ch (by establishing a bijection).

## 2.3 Decoding

The decoding of the encoded message M' is done in two steps:

- Similar to the encoding process, we represent the encoded text, by a vector of lists. Let $c_1$', $c_2$', …, $c_m$' be the different characters of M' and by $L_1$', $L_2$', …, $L_m$' the lists of positions of these characters. Using the genetic key, the positions of the characters in the initial message are recovered. As a matter of fact, the key is a

permutation of {1, 2, …, m} that we represent by a vector Key of size m a follows:

Key(1)=$p_1$, Key(2)=$p_2$, …Key(i)=1, … Key(m)=$p_m$.

Where: The characters $c_{p1}$', $c_{p2}$',…, $c_{pm}$' are going to be associated respectively to the lists $L_1$' , $L_2$', …, $L_m$'
Thus we get the message M.

- In the second step, we proceed to the deciphering of M to get $M_0$. This is easily achieved by applying the reverse functions used in the encoding process.

## 3. Extension of the evolutionary ciphering algorithm (SEC-EX)

The method of ciphering, described in section 2, can be applied to any set of entities (characters, blocks of characters, blocks of bits etc…). To be opposed to any attack based on the study of the frequencies of the characters, we may mask all the characters of the plaintext by cutting out this last, after binary coding, into blocks of bits to which we apply our evolutionary algorithm.

### 3.1 Coding

Let T be a text that we code into binary. We choose randomly an integer k. Then, we cut out T into blocks of the same size k (if the last block contains less than k bits, we complete it by bits `0'). In doing so, we obtain successive entities of the same size. Let $B_1$, $B_2$,…, $B_m$ be the different blocks of T. Denote by $L_i$ the list of the different positions of the block Bi in T and by Ch-initial the vector of components $L_i$, $1 \leq i \leq m$.

### 3.2 Ciphering

The application of the algorithm described in paragraph 2.2, to the lists $L_i$, allows changing the distribution of the lists on the different blocks $B_i$ of T. Let Ch-final be the final solution obtained by the application of this algorithm. The cipher text is obtained by associating to each block $B_i$, $1 \leq i \leq m$, the list $L_i$' of Ch-final. The pair made up of k (size of the blocks) and the permutation transforming Ch-initial into Ch-final, constitute the secret key.

### 3.3 Deciphering

Let T' be the cipher text, coded into binary and k, the first component of the secret key, we cut out T' into blocks of the same size k. Denote by $B_1$, $B_2$, …, $B_m$ the different blocks of T' and by $L'_1$, $L'_2$…, $L'_m$ theirs respective lists of positions in T'. Thanks to the second component of the secret key, the blocks are affected to their corresponding positions lists in the plaintext. Indeed, the second component of the key can be represented by a vector, which we denote by Key, of size m such that:
Key(1)=$p_1$, Key(2)=$p_2$, …, Key(i)=1, …, Key(m)=$p_m$, where : The blocks $B'_{p_1}$, $B'_{p2}$, …, $B'_{pm}$ are going to be associated respectively to the lists $L_1$', $L_2$' …, $L_m$'.
Thus, we obtain the plaintext T.
Example:
Let T be the plaintext given below:

> Abraham Lincoln, the 16th president of the United States, guided his country through the most devastating experience in its national history.

The characters (in ASCII code) and theirs lists of positions in plaintext are:

[65]: {1}
[98]: {2}
[114]: {3,28,75,80,111,139}
[97]: {4,6,53,98,101,126,131}
[104]: {5,19,25,41,66,79,84,87,134}
[109]: {7,90}
[32]:{8,17,21,26,36,39,43,50,58,65,69,77,85,89,94,106,117,120,124, 133}
[76]: {9}
[105]: {10,31,46,61,67,103,112,118,121,128,135}
[110]: {11,15,34,45,73,104,114,119,125,130}
[99]: {12,70,115}
[111]: {13,37,71,81,91,129,138}
[108]: {14,132}
[44]: {16,57}
[116]:{18,24,35,40,47,52,54,74,78,86,93,100,102,122,127, 137}
[101]: {20,29,33,42,48,55,63,88,96,107,110,113,116}
[49]: {22}
[54]: {23}
[112]: {27,109}
[115]: {30,56,68,92,99,123,136}
[100]: {32,49,62,64,95}
[102]: {38}
[85]: {44}
[83]: {51}
[117]: {60,72,82}
[103]: {59,83,105}
[121]: {76,140}
[120]: {108}
[118]: {97}
[46]: {141}

The plaintext coded in binary is:

0100000101100010011100100110000101101000011000010110110100
1000000100110001101001011011100110001101101111011011000110
1110001011000010000001110100011010000110010100100000001100
0100110110011101000110100000100000011100000111001001100101
0111001101101001011001000110010101101110011101000010000001
1011110110011000100000011101000110100001100101001000000101
0101011011001101001011101000110010101100100001000000010100
1101110100011000010111010001100101011100110010110000100000
0110011101110101011010010110010001100101011001000010000001
1010000110100101110011001000000110001101101111011010101110
1110011101000111001001111001001000000111010001101000011100
1001101111011101010110011101101000001000000111010001101000
0110010100100000011011010110111011100110111010000100000001
1001000110010101110110011000010111001101110100001100001011
0100011010010110111001100111001000000110010101111000011100
0001100101011100100110100101100101011011100110001101100101
0010000001101001011011100010000001101001011101000111001100
1000000110111001100001011101000110100101101111011011100110

0001011011000010000001101000011010010111001101110100011011
1101110010011110010010111000101100

After cutting out the text (in binary) to blocks of size 5 bits we obtain 31 different blocks. The blocks with theirs lists of positions are:

01000: {1,8,29,40,62,110,125,142,149,150,198}
00101: {2,10,25,32,50,74,88,98,162,176,194,202, 210,225}
10001: {3,19,138,191}
00111: {4,28,44,124,168,196}
00100: {5,33,41,45,57,122,152,169}
11000: {6,35,43,226}
01011: {7,47,71}
01100: {9,37,61,90,153,161,185,201,209}
10110: {11,20,60,91,116,132,180,211}
10010: {12,51,67,68,99,120,128,165,171,189,205}
00000: {13,34,69,80,93,104,136,192,213,227,228}
10011: {14,36,48,55,167,183,223}
00011: {15,23,31,39,63,87,111,112,119,127,143,151, 175,184,199,215}
01001: {16,178,216}
01101: {17,49,65,73,82,97,105,113,129,144,145, 193,218}
11001: {18,46,78,94,102,147,224}
11110: {21,221}
11011: {22,54,95,155,159,166,182,190,206,207}
01110: {24,72,89,133,137,177,200,208,217}
10000: {26,27,123,173,187}
11010: {30,75,83,115,126,131,134,163,195,203, 214,219}
11101: {38,86,114,118,130,146}
00001: {42,58,66,79,103,106,135,170}
00110: {52,76,84,100,109,140,156,164,188,197, 204,220}
01010: {53,77,81,101,141,181}
10100: {56,64,107,139,160,179,186}
10111: {59,108,148,172}
10101: {70,96,154}
00010: {85,92,157,212}
11100: {117,158,174,222}
01111: {121}

After 48 iterations we obtain the cipher text.

The secret key is:

(5,7 8 9 10 5 6 11 4 17 18 23 24 25 26 27 28 29 30 0 1 2 16 3 21 12 13 14 15 19 20 22)

The characters (in ASCII code) with theirs lists of positions in cipher text are:

[116]:{1,12}
[52]: {2}
[22]: {3,28}
[19]: {4,8,43}
[14]: {5,138}
[68]: {6,101,126,131}
[11]: {7,30,57,132}
[173]:{9}
[84]: {10,38,135}
[189]:{11}
[94]: {13}
[217]:{14,34,104,114}
[82]: {15,125}
[128]:{16}
[132]:{17}
[23]: {18,78,85}
[113]:{19,79,84}
[80]: {20,55,110}
[97]: {21,105}
[200]:{22,27}
[180]:{23}
[25]: {24,74}
[78]: {25}
[12]: {53,98}
[153]:{54}
[146]:{56}
[147]:{58,133}
[214]:{59}
[205]:{60}
[198]:{67}
[144]:{68}
[74]: {70}
[185]:[71,81,89,91}
[184]:{72,82}
[95]: {73}
[81]: {75,80}
[219]:{76,119}
[4]: {77}
[89]: {83,113}
[150]:{86}
[134]:{87}
[170]:{92}
[151]:{93}
[76]: {95}
[108]:{97}
[250]:{99}
[34]: {32,62}
[9]: {33,48,63,88}
[99]: {26,35,36,106}
[210]:{37,130}
[57]: {39,69,94,124}
[67]: {40,96}
[188]:{31,46,61,121}
[87]: {29,90}
[187]:{41,66}
[227]:{42}
[183]:{44}
[18]: {45}
[56]: {47,102,122,127}
[213]:{49,64}
[231]:{50,65}
[157]:{51}
[248]:{52,137}
[195]:{100}
[8]: {103,118,128}
[226]:{107}
[145]:{108}
[121]:{109}
[149]:{111,136}
[6]: {112}
[106]:{115}
[64]: {116,142}
[196]:{117}
[71]: {120}
[16]: {123}
[218]:{129}
[241]:{134}
[249]:{139}
[117]:{140}
[133]:{141}

We note that the cipher text characters are completely different from the plaintext characters; furthermore, they increase in number. Consequently, the frequencies apparition change after the ciphering.

## 4. SEC Experimental results and discussion

### 4.1 SEC Experimental results

We have conducted several experiments using messages of different sizes to illustrate the performance of SEC. Table 1 presents a sample of our experiments. This table gives the value of the convergence of the fitness function and the number of the generations reached at the time of this convergence. Figure 3 shows the encrypted version of message3, and its genetic key. In this example, we have used an affine ciphering defined by: $f(x) = ax+b$ where $a=-1$, $b= 255$ and $x$ is the ASCII code of the character of the initial message.

The experimental results show that the best values of the optimum (value of convergence) are reached for a population of size 24, see Table 1 and Figure 4. In some cases, the sizes 30 and 32 also gave good results, but the number of generations in these cases doubled, which is expensive. We recommend a population size of 24 for this message sizes.

Table 1: Summary statement of results

| | | Size of population | | | |
|---|---|---|---|---|---|
| | | 16 | 24 | 30 | 32 |
| Message1 1026 char. | Number of generations | 35 | 58 | 71 | 64 |
| | Value of convergence | 1288 | 1394 | 1638 | 1354 |
| Message2 684 char. | Number of generations | 26 | 58 | 46 | 53 |
| | Value of convergence | 1068 | 1100 | 982 | 1002 |
| Message3 1149 char. | Number of generations | 44 | 58 | 111 | 53 |
| | Value of convergence | 1458 | 1622 | 1512 | 1444 |
| Message4 803 char. | Number of generations | 34 | 51 | 38 | 130 |
| | Value of convergence | 886 | 1096 | 1088 | 1160 |

Abraham Lincoln, the 16th president of the United States, guided his country through the most devastating experience in its national history--the CIVIL WAR. He is considered by many historians to have been the greatest American president. Abraham Lincoln was born Sunday, February 12, 1809, in a log cabin near Hodgenville, Kentucky. He was the son of Thomas and Nancy Hanks Lincoln, Thomas Lincoln was a. Abraham had gone to school briefly in Kentucky and did so again in Indiana. He attended school with his older sister... Both of Abraham's parents were members of a Baptist congregation which had separated from another church due to opposition to slavery. When Abraham was 7, the family moved to southern Indiana As Abraham grew up, he loved to read and preferred learning to working in the fields. This led to a difficult relationship with his father who was just the opposite. Abraham was constantly borrowing books from the neighbors. Lincoln's declining interest in politics was renewed by the passage of the Kansas-Nebraska Act in 1854. In 1860 he furthered his national reputation with a successful speech at the Cooper Institute in New York.

Figure 2: Message 3 (size=1149)

The genetic key is :
9 10 11 12 13 14 15 16 17 18 19 24 25 26 27 28 29 30 31 32 33 0 1 2 3 22 4 5 7 8 23 20 21 6

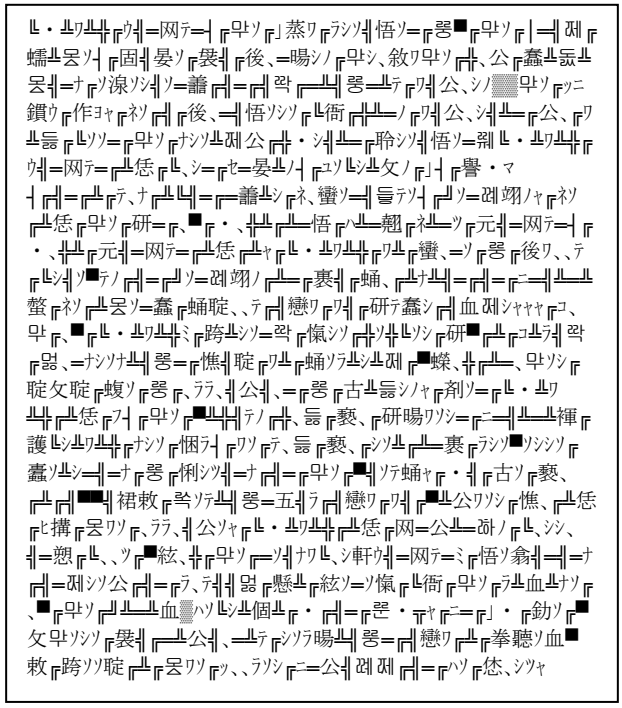

Figure 3: Encoding of the Message3

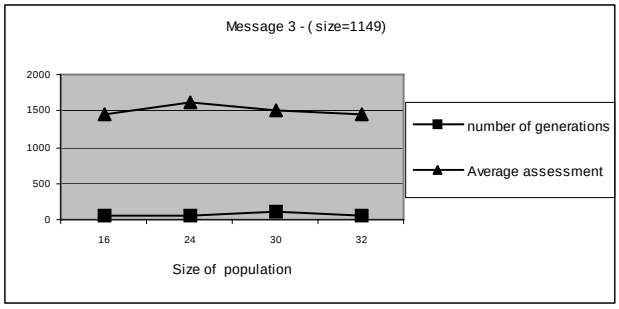


Figure 4: Experimental Results Message 3

## 4.2 SEC Discussion

This section presents a comparison of SEC with IDEA and RSA, as shown in Figure 5. This comparison is based on time of ciphering and deciphering as shown in table 2. The size of the plaintext is 16 K; for RSA we have consider a key composed with p= 103 and q=13. The experimental results show that SEC has the smallest deciphering time.

Table 2: Time of ciphering and deciphering

| | IDEA | RSA | SEC |
|---|---|---|---|
| Time of ciphering | 30 ms | 330 ms | 200 ms |
| Time of deciphering | 40 ms | 390 ms | 20 ms |

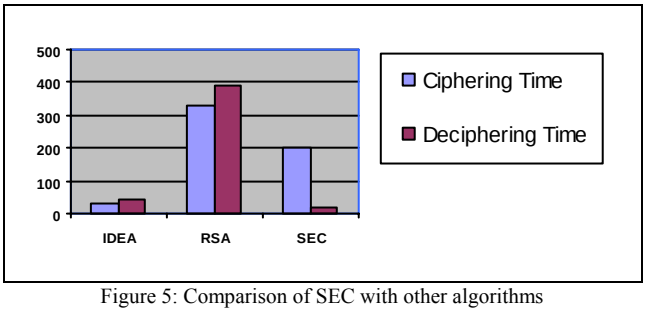


Figure 5: Comparison of SEC with other algorithms

Secondly, we compare SEC with DES, Triple-DES and AES, the most known symmetrical algorithms. These later split the message to be encoded into blocks and separately cipher each block. This will allow differential cryptanalysis to work in parallel on different blocks. SEC ciphers the entire message in only one hold. Therefore, our approach makes an encoded message safer from this type of attacks. The other frightening cryptanalysis is the brute force attack; the short length key permits it. In SEC, we assume the message contains more than 30 different characters, so the key size is at least equal to 240 bits (in the contrary case, we can complete it by others characters in order to reach a size of 240 bits). This makes our approach resistant to brute force attack. The table 3 gives length keys of above algorithms and the numbers of operations required to try all possible keys. Besides, our key is a session key and generated automatically by SEC, while DES, Triple-DES and AES keys are not.

Table 3: complexity of the brute force attack

| | Size key (in bit) | Number of operations |
|---|---|---|
| DES | 64 | $2^{64}$ |
| Triple-DES | 128 | $2^{128}$ |
| AES | 128 , 192 ou 256 | $2^{128}$ , $2^{192}$ ou $2^{256}$ |
| SEC | $\geq$ 240 | $\geq 2^{240}$ |

## 5. SEC-EX Experimental results and discussion

### 5.1 SEC-EX Experimental results

We apply our algorithm to texts of different sizes, using blocks with various sizes too. But, we present below only three examples. In a summary table, we recorded the results regarding the convergence of the fitness function, numbers of generations, for each of them.
Plaintext 1:

Cryptographic knapsacks are based on the Subset Problem: Given a finite set W={w1, ..., wn} of positive integers and a positive integer C. Does there exist a subset W' of W such that the sum of all elements in W' is equal to C?

Figure 6: First plaintext

We cut out the plaintext, coded in binary, into blocks of size equal to 5 bits. We apply the ciphering evolutionary algorithm then we obtain the cipher text (coded in hexadecimal) given bellow.

C9E5CAA1C238F51A766F3A445565D33C9F1FA2E921F11565C9
5C2B5FF74959EA5CE5C13F5FCCD9C6A26110775A262DBF89D
538D8AECA67E65FBFE77C39AEA05D9F251EA25A6B39ACA06
5DA262DBFE24309EFADCD3F3763B44D3F58E3CB65C1275FC
FDEFA39ACAA8F46A7DEAADF42263450235AA25E54493F251
FCFDEFA39ACAA8F46A244540DC620EB1A676E33DEACDEE6
591FCCD9C65C2B5FE5C60A45150D3F251FC50E7B59EB1CE77
1A0E2526771A2651076F9A26CA7F662A26CA7CA1F59EC5B65
C1275EADDDC4A26A5ECA672623CCA25F39A3BFE25CA2451
565C659AC5DDD3F5B235FBE4DA7DFF

Figure 7: Cipher text of the First plaintext

The secret key is:
(5, 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 23 39 43 44 45 46 47 48 49 50 51 0 1 2 3 4 5 6 7 8 40 41 9 10 11 12 13 14 15 16 17 18 19 20 21 22 42)

Table 4: Results of application to plaintext 1

| | Size of blocks (in bits) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Number of Blocks | 31 | 52 | 95 | 39 | 139 | 114 | 135 | 99 |
| Nber of generations | 58 | 98 | 64 | 54 | 62 | 73 | 84 | 48 |
| Average evaluation | 238 | 280 | 200 | 302 | 104 | 114 | 58 | 96 |

Plaintext 2:

Travelling salesman problem (TSP) is an NP-complete problem, implying that there is no known algorithm to solve it exactly in better than O(an) time (for n cities). In fact, there are (n-1)!/2 possible combinations, so to try every one requires factorial time with the problem size. Just to remind, there are cities and given distances between them. Travelling salesman has to visit all of them, but he does not want to travel very much.

Figure 8: The second plaintext

Like the first example, we use blocks of size equal to six. Then we obtain the cipher text given in Figure 9.

467E6FDD9468E33616547261766B186C696FE2E2726D932EE33
44D2A4BD04257362A42766C726FE2E275424962E06972E334705
7D2726D932EE3344DB64276E082680A66165472705330302A7A
4E548E582B36212B38F42B3356E12DD62B3028546307508A4EE
3D270E07261E06B0657D276DE42580B3022DF3B232B36162B3E
58DE7A586EE27053301628B5CE7667B62A42705269582A4BD9
E12281E2E262508A76548136B6E279E2E259766F30B642705334
4757D26F6D940183378D13DC6FB612816F33216C662EE334017
4630D759616748A76E068E1B64261E07270E072706DAC4154855
86DAC41E068D82A7E586C84766D94612B356F752A746D962FE
242705269582A7DF6DF30412A7A4E57D2726D932EE3344D2A7
F36099416F5A461DE4270E07247566976E191682A7A4E548E582
B300757D7017466305264612B301656425A508558E2E271508F30
7668E2548101759470DC6458E2E27053344DB6E2506D9006566B
285268DA2A7F2FE33461E260162B3BEF6C7270E07246508F36D
E426FE33C812B33192A7A4E566C282B3E65DE424E57D271E06
4612B38F4DE42527668F02A7A742A7A47748558E24246548E632
A424DDE6F0EB6E70104

Figure 9: The second cipher text

The secret key is:
(6,11 12 13 14 15 16 17 18 19 20 21 5 22 47 8 9 10 51 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 52 53 0 6 2 3 4 48 49 7 50 23 24 25 26 1 27)

Table 5: Results of the application to plaintext 2

| | Size of blocks (in bits) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Number of Blocks | 31 | 54 | 108 | 42 | 184 | 175 | 223 | 147 |
| Nber of generations | 85 | 84 | 54 | 85 | 85 | 55 | 85 | 84 |
| Average evaluation | 360 | 466 | 390 | 630 | 264 | 248 | 160 | 242 |

Plaintext 3:

Astronauts: In August 1983 Guion Bluford became the first African American to go into space, while serving on a mission aboard the Challenger space shuttle. Bluford said that the blastoff of the shuttle was like riding in a high-speed elevator through a bonfire. He also recognized that, "From a black perspective, my flight on the shuttle represented another step forward. "Astronaut Mae Jemison became the first African American woman to travel in space when she flew on the space shuttle Endeavor in a September 1992 mission. After her space flight, Jemison resigned from NASA and established the Jemison Group, a company that researches, develops, and markets advanced technologies.

Figure 10: The third plaintext

Corresponding Cipher text is :

4DECEA956E7DCE0F2D8EAD26849D468C8F02B4012146ED02B
B9BA3A90E4DEE1746AE8E0ECE881568E39220B8756AB9A3A1
E8313C1D53205A0EB56D6D5899582DACA5D6865206E8B87567
4692E446339861133DA193498C9118D85BACB58E1893175B462C
73556CD91DD54A561746B926D04C8C9DF3369D460DBC80B8E0
40471A541CA6A0A52DAE5308672BDB38B58C9118D85B462F98
D87A528E21D8A5D57C420293EF772114793820B58B0898D2A60
FE5DB210F174E392E561D40E3E4D4B58B0899A3DAE9188A945
C52262AB889011C595521498C8ED1591DD54A556746B926E9E5
BE1B86416720CDBF2118175B2A0C8B00B4A3A1E82E8818718B
A62D49DC80A737383832612D9652262DAD8E89A6A05BE3AC76
1B5310DB20B58B0898D2B746B697756E49A62D49DC8B18D855
462DDB5574615BE32E536D599726D1B53C1D5C5977E91AEC10
A6E3A1E83138DDF642210FBDDB2118926738308E886712DB20B
5DD6B4E52A521ECA0DDFA52674632881572F90620D0A6C138C
12139BF98269412A690B9BB2D995273E9EF9E17468A5BE2B9EF
0CA692A52149D82ED88121468DBF7573184E92B56D6AB82038E
2B9E92112F6F32DACA0D457A3A1380C882865A3B892B58C911
8D85B471B9C1866E3DAE8313C1D5C520147BF9211A4BDB462C
EC16929226384D8B545A575B4715E82458396193ECA41EC6E3C
E479873511A5273E230898345031C18B6570ED4413893AD612D9
8F72121ECA41898202639EDDD6DF9A382D1E5BE1885D7269831
EF0ED4769D462F735DF32B9D20CEEC1D58B6F346AE960F3763
CE9232EC187452CEE3BE536DF65240471A541CA6B35BD04C8E
80A6E3A939BCEB5345A3CE460F9E1AB7391DBD498B0898D2A
638308E8878E058E8308909C6A95B2A31EB00B70137460DE82B2

Figure 11: The third cipher text

Table 6: Results of the application to third plaintext

| | Size of blocks (in bits) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Number of Blocks | 31 | 57 | 107 | 46 | 222 | 202 | 307 | 187 |
| Nber of generations | 62 | 40 | 144 | 73 | 145 | 73 | 84 | 48 |
| Average evaluation | 548 | 870 | 524 | 850 | 458 | 432 | 286 | 380 |

The secret key is:
(7, 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 21 39 40 41 42 43 44 45 46 47 48 86 87 88 97 98 99 100 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 101 102 103 104 105 106 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 90 94 95 96 89 91 92 93)

Regarding results of application, we can deduce that the number of generations and average evaluation vary randomly from one size block to another and also from size text to another. So, cryptanalysts can never know the value of size block. This is an important advantage of our system.

### 5.2 SEC-EX Discussion

We will compare our ciphering algorithm with well-known symmetrical ciphering algorithms. Let us take, for example, IDEA [3]. It is used by the PGP [5] and considered by the specialists as one of the best symmetrical ciphering algorithms. The length of its key is 128 bits (better than of DES, which is 64 bits) and defined independently of the algorithm. Concerning our algorithm, the key is generated automatically by the algorithm, and its size is at most $k*2^k$, where k is the size of blocks. Such size is more important since it resists to the cryptanalysis by exhaustive research. Moreover, as we have signaled in the interpretation above, the size block can't be determined, so this reinforces the resistance of our algorithm.

IDEA ciphers the plaintext, block per block; what is an asset for the differential cryptanalysis [2], [14]; while our algorithm doesn't. Therefore, on one hand, it is inexpensive and on the other hand, it is immune from such cryptanalysis. However, the frightening cryptanalysis in our case is that based on the study of the apparition frequencies of the characters in the text. But, by cutting plaintext into blocks, new characters are introduced with new apparition frequencies, which avoid this kind of attacks.

## 6. Conclusion

Most of cryptologists even think that all algorithms of ciphering are long time breakable but the essential is to conceive its ciphering system in such a way that the wished life span for the encoded text must be lower than the one set by the cryptanalyst to break this system. This paper presents two encryption algorithms using genetic algorithm approach. The genetic approach used in our algorithms is similar to the one used in solving the Traveling Salesman problem. Both of algorithms use a new encoding schema and formalize the problem of encryption as a combinatorial optimisation problem.

To analyze the performance of the first algorithm SEC, we have conducted several experiments. The experimental results show that a range of a population size between 24 and 32 gives better results and therefore it is recommended for the corresponding message sizes. The experimental results also show that SEC has a fast deciphering time.

In SEC-EX, for masking the characters of the plaintext, we incorporate a new technique that is described in section 3. Thus by SEC-EX, we reach two main goals. The first one is getting an encryption algorithm more resistant against any kind of attacks. The second goal is the extension of evolutionary algorithm to a plaintext coded in binary.

Future work includes the application of the new technique developed in SEC-EX to our ciphering system, which is already published [15], to obtain a more efficient ciphering system. More precisely, our principal goal is to build a ciphering algorithm satisfying the theory of Shanon [1].

## References

[1] Shannon C. : Communication Theory of Secrecy Systems. Bell Systems Technical Journal (1949).

[2] Stinson Douglas R. : Cryptography – Theory and Practice. CRC Press (1995).

[3] Menzes A. J., Paul C. Dorschot V. , Vanstone S. A. :Handbook of Applied Cryptography. CRC Press fifth Printing (2001).

[4] Labouret G. : Introduction à la cryptographie (2001). www.hsc.fr/ressources/cours/crypto/index.html.fr

[5] Zimmermann P. : Guide de l'utilisateur de PGP . Network-Associates, Inc. (USA) (1994).

[6] Goldberg D.E.: Genetic algorithms in search optimisation & Machine Learning. Addison-Wesley. Publishing Company, Inc (1989).

[7] Clark John A. : Nature-Inspired Cryptography Past, Present and Future, Department of computer Science, university of York, York, YO10 5DD, UK.

[8] Caux, C., Pierreval, H., Portmann M.C. : Les algorithmes génétiques et leur application aux problèmes d'ordonnancement. APII. Volume 29-N° 4-5 (1995) 409-443.

[9] Mühlenbein H. : Evolutionary Algorithms Theory and applications. Wiley (1993).

[10] Mühlenbein H., Schlierkamp-Voosen D. : Predictive Models for the Breeder Genetic Algorithm, continuous Parameter Optimization. Evolutionary computation (1993) 25-49.

[11] Goldberg D.E. : Genetic algorithms in search optimization & Machine Learning. Addison-Wesley. Publishing Company, Inc (1989).

[12] Grenfenslette J.J : Optimization of control parameters for genetic algorithms. IEEE translation on systems Man, and cybernitics, Vol 16 N°1 (1986) 122-128.

[13] KhamPhang Bounsaythip C. : Algorithmes heuristiques et évolutionnistes. Thèse de doctorat université de Lille.Octobre 1988.

[14] Meier, W., "On the Security of the IDEA Block Cipher", Advances in Cryptology EUROCRYPT '93 Proceedings, Springer-Verlag, 1994, pp. 371-385.

[15] Omary, F., Mouloudi, A., Tragha, A., Bellachia, A.,. "A New Ciphering Method Associated With Evolutionary Algorithm". M. Gavrilova et al. (Eds.) : ICCSA 2006, LNCS 3984, pp. 346-354, 2006. Springer-Verlag Berlin Heidelberg 2006.

**Fouzia Omary** received the B.S. degree in applied mathematics from Mohammed V University , Morocco, 1983, and Doctorate of High Graduate Studies degree in Theories of Computer Sciences from Mohammed V University, Morocco, 1988 and Doctorate of state degree (or Ph D) July 2006 in Computer sciences from Mohammed V University, Morocco. In 1983, she joined the faculty of science of Mohammed V University, Morocco where she is currently a Professor in Department of mathematics and computer sciences.

**Abderrahim Tragha** received the B.S. degree in applied mathematics from Mohammed V University, Morocco, 1983, and Doctorate of High Graduate Studies degree in Theories of Computer Sciences from Mohammed V University, Morocco, 1988 and Doctorate of state degree (or Ph D) July 2006 in Computer sciences from Hassan II University, Morocco. In 1988, he joined the faculty of science of Ben M'sik, Hassan II University, Morocco where he is currently a Professor in Department of mathematics and computer sciences.

**Abdelghani Bellachia** received the PhD degree in Computer Sciences at USA. He is currently a Professor in Department of computer sciences in Georges Washington University, USA.
Research Interests: Data mining, multi-lingual information retrieval systems, cross-language retrieval systems, database management systems, bio-informatics, design and analysis of algorithms, handheld computing, and parallel processing.

**Abdelaaziz Mouloudi** received the B.S. degree in applied mathematics from Mohammed V University, Morocco, 1982, and Doctorate of High Graduate Studies degree in Theories of Computer Sciences from Mohammed V University, Morocco, 1988. In 1988 he joined the faculty of science Ibn Tofail University, Morocco where he is currently a Professor in Department of mathematics and computer sciences.