ICIGA: Improved Cryptography Inspired by Genetic Algorithms

A. Tragha
Faculty of Science Ben M'Sik University Hassan II Casablanca Morocco.
a.tragha@univh2m.ac.ma

F. Omary
Faculty of Science University Mohammed V Agdal Rabat Morocco
omaryfouzia@yahoo.fr

A. Mouloudi
Faculty of Science University Ibnou Tofail Kenitra Morocco
mouloudi_aziz1@yahoo.fr

Abstract

This work is a contribution to cryptography. We have developed a symmetrical blocks ciphering system, that is an improvement of our cryptosystem inspired from genetic algorithm published in AMSE periodical. His complexity is o(n) because all operations used are simple and they are applied randomly in ciphering. Furthermore, in our system the user can choose the size blocks and the length key.

1. Introduction

The Most of current research in cryptography concerns asymmetrical cryptography, while research in symmetrical cryptography has remained relatively stationary since the discovery of the diagram of Feistel.

Moreover, none of the most used symmetrical ciphering systems like DES, IDEA and AES,[1-3] make use of the most recent developments in information processing technology. In this paper, we will describe a new symmetrical block ciphering system named ICIGA (Improved Cryptography Inspired by Genetic Algorithms) which generates a session key in a random process. The block sizes and the key lengths are variable and can be fixed by the user at the beginning of ciphering.

ICIGA is an enhancement of our first system (GIC) "Genetic algorithms Inspired Cryptography" published in the AMSE periodical of November 2005 [4]. To the original system we added other transformations in order to strengthen its resistance to cryptanalysis. These transformations have no influence either on the size of the key or on the complexity of ICIGA which is *o*(n). Furthermore, ICIGA uses an entirely new random process for the choice of the blocks allowing for their total modification. The ciphering key is a session key, generated during ciphering, and in this improved version, the user according to his needs determines its size.

This paper is composed of three principal parts followed by a conclusion. The first part contains an informal description of ICIGA, followed by a formal definition. In the second part we will first study the complexity of ICIGA and then we will evaluate its effectiveness by showing its resistance to cryptanalysis. The third part is devoted to the results obtained following the implementation of ICIGA in Java. We end this paper by a discussion based on the comparison between ICIGA and the most well known symmetrical cryptosystems such as DES, IDEA and AES.

2. Description of the ICIGA System

To improve on our original GIC (Genetic algorithms Inspired Cryptography) system presented in a previous paper [4], we introduce new ciphering techniques that lead to a new symmetrical ciphering system which we have named ICIGA (Improved Cryptography Inspired by Genetic Algorithms). ICIGA is a block cipher system whose secret key is generated during each session using a random process. The user can fix the size of the blocks as well as the length of the key.

The operation of ICIGA depends on the length of the secret key selected by the user. ICIGA uses this length to divide the plaintext into parts of equal size. During the ciphering, the first part is broken up into blocks of the same size which are used to generate the secret key. This key will then be used to cipher the other parts of the message. If the user did not set the length of the secret key the plaintext is composed of only one part and ICIGA generates a key of maximum length.

2006 International Conference on Hybrid Information Technology (ICHIT'06)
0-7695-2674-8/06 $20.00 © 2006 IEEE

The genetic operations of crossover and mutation used for ciphering in GIC are improved by the new system ICIGA as follows:

- A left shift is added to each block that is processed, and
- Another left shift is added to the part being processed after the processing of its last block.

The number of bits to be shifted is determined by the parameters of the genetic operations. The goal of these shifts is to reinforce resistance to cryptanalysis, and in particular to techniques of exhaustive search.

2.1. Informal description

In ICIGA, the principal operations used in ciphering are substitutions and permutations of the bits of the plaintext. The four steps of ciphering are:

- Step 1:
  - Break up the plaintext (encoded into binary) into parts of equal sizes depending on the selected key length.
  - Break up each part into blocks of the same fixed size.
- Step 2:
  - Apply the genetic operations using as arguments randomly selected blocks and positions. The trace of these operations (type of the operation and its arguments) make it possible to generate the secret key K.
  - Mask the positions of each genetic operation by the application of a left shift on the corresponding block(s).
- Step 3:
  - Apply another left shift on the whole part in order to mask the distribution of its blocks.
- Step 4:

The other parts are treated sequentially using the same operations as above. The arguments of the genetic operations will not be this time randomly drawn, but deduced from the key K.

2.2. Formal description

2.2.1. Definition of the genetic operations. U and V are two blocks of bits with size t; p and q are two positive integers such that $1 \le p \le q \le t$.

U= | a_1 | a_2 | … | a_{p-1} | a_p | a_{p+1} | … | a_{q-1} | a_q | a_{q+1} | … | a_t |

V= | b_1 | b_2 | … | b_{p-1} | b_p | b_{p+1} | … | b_{q-1} | b_q | b_{q+1} | … | b_t |

The crossover of U and V at the positions p and q gives two (sons) blocks U' and V' obtained while permuting in U and V the bits of indices ranging between p and q in opposite order [5], [6].

U'= | a_1 | a_2 | … | a_{p-1} | b_q | b_{q-1} | … | b_{p+1} | b_p | a_{q+1} | … | a_t |

V'= | b_1 | b_2 | … | b_{p-1} | a_q | a_{q-1} | … | a_{p+1} | a_p | b_{q+1} | … | b_t |

The mutation of the block U at the positions p and q gives the block U" obtained while replacing the bits of indices ranging between p and q by their logical negations [5], [6].

U= | a_1 | a_2 | … | a_{p-1} | a_p | a_{p+1} | … | a_{q-1} | a_q | a_{q+1} | … | a_t |

U"= | a_1 | a_2 | … | a_{p-1} | $\breve{a}_p$ | $\breve{a}_{p+1}$ | … | $\breve{a}_{q-1}$ | $\breve{a}_q$ | a_{q+1} | … | a_t |

With $\breve{a}_j = 1 - a_j$.

2.2.2. Presentation of ICIGA. Our ciphering system ICIGA is formally the quintuplet (**P**, **C**, **K**, **e**, **D**), where:

P is the set of possible plaintexts, $\mathbf{P} = \{0,1\}^*$.

C is the set of the possible cipher texts, $\mathbf{C} = \{0,1\}^*$.

K is the set of the parts of

$(\mathbb{N} \cup \{-1\}) \times \mathbb{N} \times \{(p, q) \text{ such as } 1 \le p \le q \le t\}$

Thus, a key is a set of quadruplets (i, j, p, q) such that:

$i = -1$ if the block of index j is subject to a change by a mutation in the positions p and q.

$i \ge 0$ if the blocks of indices i and j are subject to change by a crossover to the positions p and q.

e is the ciphering function defined as follows

$\mathbf{e} : \mathbf{P} \rightarrow \mathbf{C} \times \mathbf{K}$

$T \rightarrow \mathbf{e}(T) = (T', k)$

It associates a cipher text T' and a key k generated during ciphering to any plaintext T. The generated key serves also for the deciphering.

D is the set of deciphering functions $\mathbf{d_k}$ with **k** in **K**, the deciphering functions $\mathbf{d_k}$ satisfy:

$\mathbf{d_k}(T') = T$ if and only if $\mathbf{e}(T) = (T', \mathbf{k})$.

The algorithms given below define the functions of ciphering and deciphering.

3. Algorithms in details

3.1. Encryption algorithm

3.1.1. Mutation.

Input: A block U of t bits, two integers n and m such that $1 \le n \le m \le t$.
Output : A block U' obtained from U after replacing the bits of indices ranging between n and m by their logical negations
Method :
For i := 1 to n-1 do U'[i] := U[i] ;
For i := n to m do U'[i] := 1- U[i] ;
For i := m+1 to t do U'[i] := U[i] ;

3.1.2. Crossover.

Input: Two blocks U and V of t bits, two integers n and m such that $1 \le n \le m \le t$.
Output: Two blocks U' and V' obtained while permuting in U and V the bits of indices ranging between n and m in opposite order.
Method :
For i := 1 to n-1 do
  Begin U'[i] := U[i] ;
         V'[i] := V[i] ;
  End;
For i := n to m do
  Begin
  U'[i] := V[n+m-i] ;
  V'[i] := U[n+m-i] ;
  End;
For i := m+1 to t do
  Begin U'[i] := U[i] ;
  V'[i] := V[i] ;
  End;

3.1.3. Left shift

Input: A block X of bits, an integer n; $1 \le n \le |X|$.
Output: A block Y obtained while shifting the n first bits in X to the end.
Method :
For i := 1 to (|X| - n) do Y[i] := X[i + n] ;
For i := (|X|-n+1) to |X| do Y[i] :=X[i-|X|+n] ;

3.1.4. Algorithm 1. (Encryption)

Input: T the plaintext of size N, t the size blocks and m the length keys
Output: a cipher text T' and a ciphering key K of length ≈ m;
Method :
If m is not define then use algorithm 2;
Calculate $n = (1 + \lfloor 4*m/3 \rfloor)* t$;
If n≥N then use Algorithm2 to obtain T' and K Else
Begin
  Cut out T into parts:
    $P_1P_2 \ldots P_\alpha$ that are of equal size n;
  Initialize the key K with an empty list;
Parameter:=ith 0; /*Parameter of final left shift */
/* Treatment of the first part P_1*/
Cut out T into blocks of size t;
While there is no marked blocks in P_1 do
Begin
  If there is one no marked block then
  Begin
Choose randomly two integers $1 \le p \le q \le t$;
 Mutation (B, p, q); Left shift (B, q - p);
 Mark B; Parameter:= parameter + (q – p) ;
 Add [-1, index of B, p, q] to K;
  End Else /*more than one no marked block */
Begin
  Choose randomly the operation to apply;
  Choose randomly two integers $1 \le p \le q \le t$;
  If the operation is Mutation then
  Begin
   Choose randomly a no marked block B ;
   Mutation (B, p, q); Left shift (B, q - p) ;
   Mark B; Parameter := parameter + (q – p) ;
   Add [-1, index of B, p, q] to K ;
   End else /* Crossover is chosen */
  Begin
Choose randomly two no marked blocks B_1 and B_2
Crossover (B_1, B_2, p, q) ;
   Left shift (B_1, q - p) ; Left shift (B_2, q – p) ;
   Mark B_1 and B_2;
   Parameter := parameter + (q – p) ;
   Add [index of B_1, index of B_2, p, q] to K ;
   End
 End
End /* While*/
Left shift (P_1 , Parameter) ;
The part P'_1 obtained is the ciphering of P_1 ;
K is the ciphering key;
/* treatment of the part P_2, P_3, …P_α */
For s := 2 to α do
Begin
Cut out P_s into blocks of size t;
For each quadruplet (i, j, p, q) in K do
Begin
  If i = -1 then /* Mutation */
  Begin
    Let B the j^{th} block in P_s ;
    Mutation (B, p, q); Left shift (B, q - p) ;
  End
  Else /* Crossover */
  Begin
  Let B1 the i^{th} block and B2 the j^{th} block in P_s
     Crossover (B1, B2, p, q);
     Left shift (B1, q – p); Left shift (B2, q – p);
   End ;
 End;
 Left shift (P_s , Parameter) ;
The part P'_s obtained is the ciphering of P_s ;
End;
The concatenation $P'_1P'_2 \ldots P'_\alpha$ is the ciphertext T'.

3.1.5. Algorithm 2. (Encryption)

Input: T the plaintext of size N, t the size blocks
Output: a cipher text T' and a ciphering key K
Method :
Begin
  Cut out T into blocks of size t;
  Initialize the key K with an empty list;
  Initialize parameter with 0; /*Parameter of left shift will be applied at the end*/
/* Treatment */

```
While there is no marked blocks in T do
Begin
 If there is one no marked block then
 Begin
Choose randomly two integers 1 ≤ p ≤ q ≤ t;
 Mutation (B, p, q); Left shift (B, q - p);
 Mark B; Parameter:= parameter + (q – p) ;
 Add [-1, index of B, p, q] to K;
 End Else /*more than one no marked block */
 Begin
  Choose randomly the operation to apply;
  Choose randomly two integers 1 ≤ p ≤ q ≤ t ;
  If the operation is Mutation then
  Begin
   Choose randomly a no marked block B ;
   Mutation (B, p, q); Left shift (B, q - p) ;
   Mark B; Parameter := parameter + (q – p) ;
   Add [-1, index of B, p, q] to K ;
  End else /* Crossover is chosen */
  Begin
Choose randomly two no marked blocks B1 and B2
Crossover (B1, B2, p, q) ;
   Left shift (B1, q - p) ; Left shift (B2, q – p) ;
   Mark B1 and B2;
   Parameter := parameter + (q – p) ;
   Add [index of B1, index of B2, p, q] to K ;
   End
 End
End /* While*/
Left shift (T , Parameter) ;
The text T' obtained is the ciphering of T ;
K is the ciphering key with maximal length;
End;
```

3.2. Decryption algorithm

The process of deciphering in ICIGA is simple. It is composed of right shifts and genetic operations. In fact, the inverse of a left shift is a right shift and the reciprocal operations of mutation and crossover are equal to themselves. The latter operations are involutive. We note that the order of the deciphering operations is the reverse of the ciphering order. Indeed, the last coded operation should be deciphered firstly and the deciphering process continues in a LIFO manner.

3.2.1. Right shift.

```
Input: a block X of bits, an integer n; 1 ≤ n ≤ |X|.
Output: a block Y obtained after shifting the n end bits in X to the beginning.
Method :
For i := 1 to n do Y[i] :=X[|X|-n+i]
For i := n+1 to |X| do Y[i] := X[i - n] ;
```

3.2.2. Algorithm of deciphering.

```
Input: a ciphertext T' of size N and a key K.
Output: The plaintext T correspond to T'.
Method :
Let K = [(i1, j1,p1,q1) ; (i2, j2,p2,q2);...; (im, jm, pm,qm)]
Browse K for calculate:
  the size blocks t, chosen during the ciphering;
  the size of the first part P'1 and
  the number of bits to right shift P'1 .
val := (q1-p1)+(q2-p2)+…+(qk-pk)
Divide T' into parts P'1P'2…P'α which have same size;
For s := 1 to α do
Begin
Right shift (P's , val) ;
Divide P's into blocks of size t ;
For each quadruplet (i, j, p, q) in K do
Begin
  If i = -1 then /*it correspond to a mutation*/
  Begin
    Let B the ith block in P's ;
    Right shift (B, q - p) ;
    Mutation (B, p, q) ;
  End ;
    Else /*it correspond to a crossover */
  Begin
    Let B1 The ith block and B2 the jth block P's;
    Right shift (B1, q - p);
    Right shift (B2, q - p);
    Crossover (B1, B2, p, q);
  End ;
 End;
Thus we obtain the part in clear Ps ;
End ;
Concatenate the resultant parts P1 P2 … Pα ;
The text T obtained is the plaintext ;
```

4. Evaluations of our system ICIGA

4.1 The complexity of ICIGA

We note that the principal operations used in ciphering are just substitutions and permutations of the bits of the plaintext.

Furthermore, we can show that the complexity of our system ICIGA is o(n) by modeling the basic operations of ciphering by finite and pushdown automata [7].

Indeed, the mutation is modeling by deterministic finite automata, the crossover by double pushdown automata and the shift by pushdown automata, all are without loop.

4.1.1. Modeling of Mutation operator by deterministic finite automata (DFA). Let t be the size block, p and q the positions of Mutation.

Formally, a DFA with output is the quintuplet (Q, $\underset{\sim}{\Sigma}$, Δ, δ , e_0, F) where Q is the set of states, $\underset{\sim}{\Sigma}$ is the input alphabet , Δ is the output alphabet , e_0 is the start state, F is the set of final state, it can be empty and δ is the transition function defined by :

Q x Σ--------> Q x Δ

(e , a) --------> (e', c)

Q={ e_0, e_1, e_2, …, e_{n-1}, e_t}

$\underset{\sim}{\Sigma}$= Δ ={0, 1}, F=$\varnothing$ and δ defined by :

δ (e_i,a)=(e_{i+1},a) for i=0, …, p-1

δ (e_i,a)=(e_{i+1},1-a) for i=p, …, q

δ (e_i,a)=(e_{i+1},a) for i =q+1, …, t-1.

For example: for t=8, p= 3 and q =6, we have

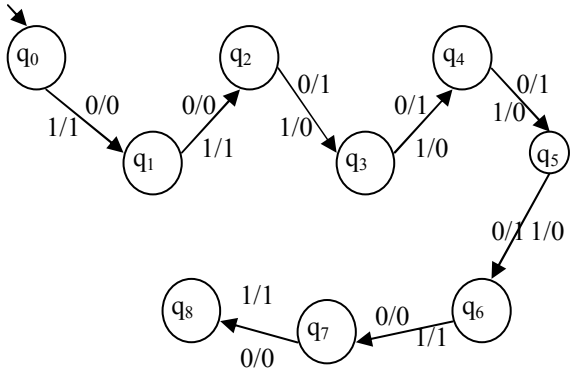

Figure 1 . Diagram of DFA with output

If the input is the block B = 10110001 then Mutation (B, 3, 6)=10001101 is the output

4.1.2. Modeling of Crossover operator by Pushdown automata. Let t be the size block, p and q the positions of Crossover.

A Two-Pushdown automata with output "2PDA-O" is a pushdown automata with two input tapes, two output tapes and two stacks. Formally, it is a system (Q , Σ, Δ, Γ , δ , e_0 , Z_1, Z_2, F) Where:

- Q is a finite set of states;
- Σ is an alphabet called the input alphabet;
- Δ is an alphabet called the output alphabet;
- Γ is an alphabet, called the stack alphabet;
- e_0 in Q is the initial state;
- Z_1 and Z_2 in Γ are particular stack symbols called the start symbols, Z_1 correspond to first stack and Z_2 correspond to second stack;
- F a subset of Q is the set of final states;
- δ is a mapping from Qx(ΣU{ε}x(ΣU{ε}) x (ΓxΓ) to set Qx (ΔU{ε}x(ΔU{ε} x (Γx Γ)

The interpretation of

δ (e ; (a,b) ; (X_1,X_2)) = (e' ; (c,d) ; (α_1,α_2)) where e and e' are states, a, b, c and d are in Σ, X_1 and X_2 are stack symbols and α_1,α_2 are in Γ^* , is that the 2PDA-O in state e, with input symbols α_1 in first tape input and α_2 in second tape input; X_1 and X_2 are respectively the top symbols on the first stack and the second stack, can enter state e', replace X_1 by string α_1, X_2 by string α_2, put c in the first output tape and d in the second output tape (See fig. 2).

In our case, we take Q={ e_0 , e_1 , ….. , e_t}, $\underset{\sim}{\Sigma}$= $\underset{\sim}{\Delta}$={ 0,1}, Γ={Z_1, Z_2, X, Y }, F= $\varnothing$ and δ defined by :

δ (ei ;(a,b) ;(Z1,Z2))=(ei+1 ; (a,b) ; (Z1,Z2))
For i = 0, ... , p-1 and all symbols a, b
δ (e_i ; (0,0) ; (A,B)) = (e_{i+1} ; ($\underset{\sim}{\varepsilon}$,$\varepsilon$) ; (AX,BX))
δ(e_i ;(0,1) ;(A,B)) = (e_{i+1} ; ($\underset{\sim}{\varepsilon}$,$\varepsilon$) ; (AX,BY))
δ(e_i ;(1,0) ;(A,B)) = (e_{i+1} ; ($\underset{\sim}{\varepsilon}$,$\varepsilon$) ; (AY,BX))
δ (e_i ;(1,1) ;(A,B))= (e_{i+1} ; ($\underset{\sim}{\varepsilon}$,$\varepsilon$) ; (AY,BY))
For i = p, p+1, … q And all A, B in Γ
δ (e_{q+1} ; (ε,ε) ; (X,X)) = (e_{q+1} ; ($\tilde{0}$,$\tilde{0}$) ; ($\underset{\sim}{\varepsilon}$,$\varepsilon$)) .
δ (e_{q+1} ; (ε,ε) ; (X,Y)) = (e_{q+1} ; (1,$\tilde{0}$) ; ($\underset{\sim}{\varepsilon}$,$\varepsilon$))
δ (e_{q+1} ; (ε,ε) ; (Y,X)) = (e_{q+1} ; ($\tilde{0}$,1) ; ($\underset{\sim}{\varepsilon}$,$\varepsilon$))
δ (e_{q+1} ; (ε,ε) ; (Y,Y)) = (e_{q+1} ; (1,1) ; ($\underset{\sim}{\varepsilon}$,$\varepsilon$))

We can easily deduce that for two blocks B_1 and B_2 of size t, the Crossover (B_1, B_2, p, q) is the two output of the mentioned 2PDA-O.

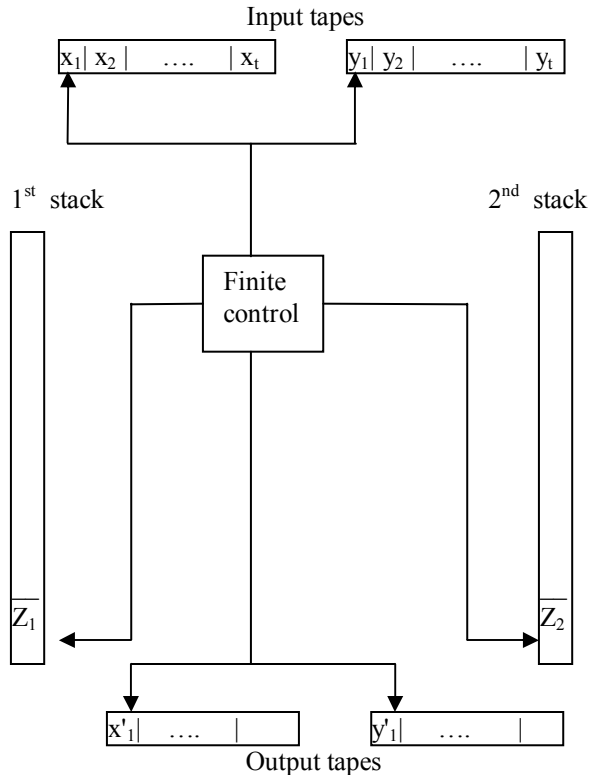

Figure 2 . A 2PDA with output

4.2 Comparison with some cryptosystem

In this section, we compare our algorithm with DES, TDES, IDEA and AES, the most well known symmetrical algorithms. The DES has a key of 64 bits but only 56 bits are used. This short key encourages attackers to decode a message by an exhaustive search using powerful computers. In our algorithm, the size of key depends of the number of genetic operations mapped during the ciphering; this reinforces the resistance against such attacks. Furthermore, our key is a session key generated by the algorithm, while the DES, IDEA and AES keys are not. This allows again our system to be more resistant.

With respect to IDEA, the time complexity of the main operations: XOR, addition modulo 2^{16} and multiplication modulo $2^{16}+1$ is very high, while our operations: crossover, mutation and shift are not expensive as proved in 4.1, thus our system is more powerful.

Our system further allows the user to define the length key that according to the degree of security that he prefers. ICIGA uses a combination of the two modes of ciphering by blocks: the Electronic Code Book mode (ECB) and the Cipher Block Chaining mode (CBC); indeed the ECB corresponds to the mutation operations while the CBC corresponds at the crossover operations.

The table below is a comparative summary of the algorithms.

Table 1. Comparison of keys and size blocks

	block	key	Type of key
DES	64	56	Fixed at Beginning
TDES	64	112 /168	Fixed at Beginning
IDEA	64	128	Fixed at Beginning
AES	128	128/192/256	Fixed at Beginning
GIC	Multiple of 8	Variable	Generated randomly during Ciphering
ICIGA	An integer > 1	Defined by user	Generated randomly during ciphering

Table 2. Comparison of stages and operations

	stages for a block	Operations used
DES	16	Substitution and permutation using boxes
TDES	48	Substitution and permutation using boxes
IDEA	8	Exclusive Or, Add mod 2^{16} & multiply mod $2^{16}+1$
AES	10/12/14	Substitution, Shift & addition
GIC	One stage	Crossover & mutation (genetic operators)
ICIGA	One stage	Crossover, mutation, left shift & right shift.

4.3. Experimental results

After the implementation of our system ICIGA in Java, we have obtained following results.

For the plaintext:

CRYPTOGRAPHY BASED ON RANDOM NUMBERS
Jason R. Kauffman, a sophomore at the University of Dayton majoring in mechanical engineering, has developed a new encryption technology based on random-number generation. Kauffman first thought of the idea while working on a science-fair project to improve computer animation. He extended a mathematical technique used in Disney's "The Hunchback of Notre Dame," which assigned pseudo-random numbers to body movements for a crowd scene in the film. While studying number generators, he found references to theories that the technique could be used in encryption technology, but no details. He then thought of a unique way to use random numbers in a math equation to encrypt data. He and his father, Robert Kauffman, formed a partnership with the University of Dayton to patent the idea.
Chronicle of Higher Education, 3 July 2002

The application of ICIGA with 53bits as size blocks and 5 as length of key, gives the ciphertext:

z_ÕaI_A!d¤_4TE„âœ@¤,¦ ^"Ò_âT"'U%0Ð¥¦$Õê‰ q\Ü1-…Õ™™´_â Â__6 ÞàÐ ÞÞÐ¦R__C_Z†R_Vâ—2;ŠCÉ™CA½¯¦ _…çFöâ_®_"ÞäÒÜ Þ @–àÐ¦Ö V6†_æ–6_±".‹{_f••É¥'œ,_†_3æFÊîÊ ØÞ ÎB___æV²_Væ 7'›R 4í_œ_í•¦ ¡'½±öw'_!—6ÊÈ__ßÜG&_æF 6ÐÖçVÖ'×_ÒÇ"^-5 Ñ¥½¸€ ¶_Vfx Ö"Ü@ÌÒàæÂ_F†ö_z‡B_öv_³RL£)__D¦ Ý¡¥±˜Ð§v÷^¶Ü"ÛÎ@ÞÌ@__66" V¶6RÖf6—[ó,",^b¦ Ð¦ Ñ¼¦ ¶×_&ö†T@ÆÞÚà³ÍW"__ú–__F–÷"çñµí+È_í Ñ•¹'•¦ ___Ö_Æ¦ ÊÚÂèÓÆÖÂ_FV&€æ—_VV_s^AOq\Þ±_¥Í'•äw2_%X †_@êÜ ÃÐ¦_6²_6pÐ¤æ÷[&_{¦)aHH¦ Ý¡¥¢__76îv_ÊÊ@àöËVF ò×#_¶F öÒ_·WsÓ¬ƒ«Xæ ¼¦ ‰ ½'ä†Ö÷fW6PÜèæ@ÌžàÐ¦__#'7vB_7vWQ¶d_›˜½¡"¦ ™¥±Òâ_v¦ _ÔÊ@ æ èëÈò–ær_7TÖ&W"_ösòÌÇ{[ž¹Ì°¦ ¡",f÷Væ^Ð'äÊÌÊà Êæ6W2_Fò_F †V ÷&'³®ƒ YC_¦ Ñ¡"¦ ÞV6†æï_*Ê@ÆÞúØB_&R_W6VB_–ò_› ð¦ _©[æ¥ ½¸4)Ñ–6†æ÷&òÎòX@Âªê_æò_zV__–Ç3_ '·.+R_´¦ Ñ¡½Õ¦ ‡B_öe†_@êÜÒãêÒ_v_'‡Bò_W6PP³ñm_iœ Ÿ±'Õµ ‰• É2_– â__0ÚÂèÐDÊ—V_F—vú_Fò_Zæ_3ã Ü¦ EÑ„,¦ !š__æB~†'æ @ÌÃøÐW"ÀÐ &¶&W'BT·Ñð„x]'°¦ ™½Éµ–B___æ_äèÜÊä¦Ö—__vƒF__F † SEU35¬[› ©Ñä¦ ½¯¦ F_—Fñb èÞ__áÂæVçB_6,R_–FQ_ñ3 NO{ƒEl¥ ¦ ±"¦ ¼b_†–h†:ä@ ŠÈîÆ—F–öã__2_§VÛ'_'Ž

And the key: [-1 0 13 31] [5 4 40 43] [-1 3 41 41] [-1 1 1 2] [-1 2 39 42]

A second session with same size blocks and length key gives another ciphertext and the following key:

[2 5 12 17] [4 1 32 46] [-1 3 37 42] [-1 0 19 40], since the key changes in each session.

On the other hand, the figure1 allows us to place our system among the others symmetrical cryptosystems.

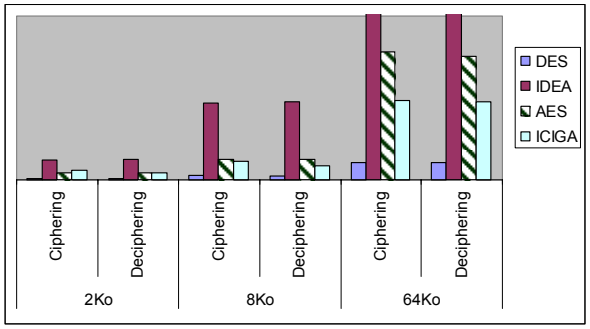

Figure 3. Graphical representation of execution time

The figure above is obtained from the following results:

Table 3. Time of ciphering and deciphering (second)

Size of plaintext		DES	IDEA	AES	ICIGA
2Ko	Cipher	0.045	O.721	0.268	0.343
	Decipher	0.038	0.735	0.268	0.270
8Ko	Cipher	O.148	2.809	0.752	0.661
	Decipher	0.118	2.864	0..752	0.501
64Ko	Cipher	0.621	22.142	4.68	2.907
	Decipher	0.619	22.823	4.59	2.844

5. References

[1] Menzes A. J., Paul, C., Van Dorschot, V., Vanstone, S. A., *Handbook of Applied Cryptography,* CRS Press fifth Printing 2001.

[2] Douglas, R. Stinson, *Cryptography- Theory and Practice*, CRC Press, 1995.

[3] Wenbo Mao, *Modern Cryptography: Theory and Practice*, Publisher: Prentice Hall PTR, Copyright: Hewlett Packard, 2004

[4] A. Tragha, F. Omary, A. Kriouile, "Genetic Algorithms Inspired Cryptography" *A.M.S.E Association for the Advancement of Modelling & Simulation Techniques in Entreprises, Series D : Computer Science and Statistics,* Novembre 2005.

[5] D.E. Goldberg, *Genetic algorithms in search optimisation & Machine Learning*, Addison-Wesley. Publishing Company, Inc 1989.

[6] C. Caux , H. Pierreval and M.C. Portmann "Les algorithmes génétiques et leur application aux problèmes d'ordonnancement" *APII*, Volume 29-N° 4-5, 1995, pp. 409 - 443.

[7] J. E. Hopcroft, J. D. Ullman, *Automata Theory Language, and Computation*, Addison Wesley,1979.

[8] Claude Shannon**,** "Communication Theory of Secrecy Systems" *Bell Systems Technical Journal*, 1949.