JAVA IMPLEMENTED ENCRYPTION ALGORITHM

Luminița SCRIPCARIU[1], Andrei ALISTAR[2], Mircea Daniel FRUNZĂ[3]
Technical University "Gh.ASACHI" of Iaşi, ROMANIA
Bd. Copou 11, RO – 700506 Iasi
[1]luminita.scripcariu@gmail.com

Abstract. *Secure communications involves the use of different cryptography methods, with digital keys longer than the transmitted message [1]. Many applications dedicated to text transmission request special encryption algorithms. A secure encryption algorithm, with a great diversity of the key space, is proposed in this paper, based on polynomial invertible functions defined on Galois Fields [2].*
Keywords: *communication security, encryption, algorithm, Galois field.*

Invertible Functions Defined on Galois Fields

A class of 3-parameters polynomial invertible functions defined on Galois Fields, with m-bits symbols, can be used for cryptography:

$$E_{\bar{k}}(\bar{a}) = \bar{c} = k_1 + k_2 \bar{a}^{k_3}, k_2 \neq 0, k_3 \neq 0$$

$$k_3 \neq 2^m - 1, \bar{k} = [k_1, k_2, k_3] \quad . \qquad (1)$$

The data vector $\bar{a} = \overline{a_1 a_2 ... a_m}$ is an element of the definition field GF (2^m) and the m bits a_1, a_2 … a_m are obtained by decimal-to-binary conversion of this element.
The inversed functions, defined on the same GF, are:

$$E_{\bar{k}}^{-1}(\bar{c}) = \left[k_1^{-1}(\bar{c} + k_0)\right]^q \qquad (2)$$

The inverse key component q verifies:

$$(\bar{a}^{k_2})^q = \bar{a} \qquad (3)$$

$$(k_2 \cdot q) \bmod (2^m - 1) = 1 \qquad (4)$$

The maximum number of all 3-parameters polynomial invertible functions defined on a GF (2^m) is given by:

$$M = 2^m \cdot (2^m - 1) \cdot (2^m - 2) \qquad (5)$$

It can be approximated exponentially as:

$$M \cong ct \cdot 10^N, 1 \leq ct < 10 \qquad (6)$$

The existence of q for all k_2 values is guaranteed only if m and 2^m-1 are simultaneously prime numbers, when all the elements of the GF have the same order equal to 2^m-1 and the maximum number of simple encryption functions defined on a GF (2^m) is obtained (Table 1).

For example, on GF (8) there are two symmetric couples (k_2, q), (1 - 1) and (6 - 6), and two asymmetric couples: (2 - 4) with (4 - 2), (3 - 5) with (5 - 3).
These functions are applied on each data block of m bits, with a special set of parameters, randomly generated with the initial state ($\bar{s}$) defined by the public key. All the arithmetic operations (addition, multiplication) are defined on the GF.

Table1. Optimum GF Dimensions

m	2^m-1	N
2	3	1
3	7	2
5	31	4
7	127	6
13	8 191	11
17	131 071	15
19	524 287	17
31	2 147 483 647	27

Chaotic Number Generators

The key-vector $\bar{k}$ contains the encryption function parameters and it is used as the key sequence for encryption. This is given by a chaotic number generator based on a dynamic system described by the logistic function:

$$x_{k+1} = Rx_k \cdot (1 - x_k),\ x_k \in (0, 1), R \in (0, 4). \qquad (7)$$

The initial value is deduced based on the user password by character–to-number conversion.
The chaotic generated number stream is sampled and quantized in order to obtain only those values included in the GF used for encryption.
The non-zero values are excluded for two components of the encryption key and the maximum value of the GF is also forbidden for the exponent value. Special cases regarding the encryption key-vector should be analyzed and solved in the Java implemented application.
The public key represents only the non-zero initial state of a random number generator on a GF, so it is not a very long sequence and the number of choices is increased by m.
The secret key used for encryption is the generated random number sequence itself and it is much longer than the public key.

Encryption Principle

The serial data stream is partitioned into blocks of m bits which are converted binary-to-decimal to obtain an element of the GF (Figure 1). For each value $\bar{a}$, the encryption function uses another set of parameters.
The same value could be different encoded, based on the memory of the pseudo-random number generator. The public key is also randomly modified to maximize the length of the encryption key.
The public key could be changed for every new transmitted frame or on-demand. In this way, the encryption method avoids being a redundant code without any memory. The encryption key length is greater or comparable to the message length.

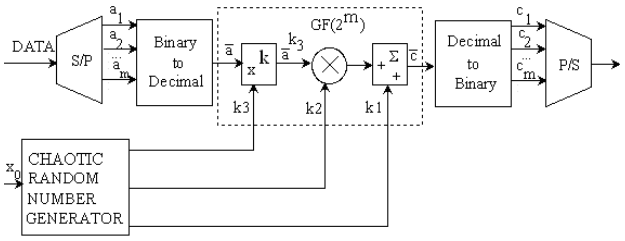

Figure1. Encryption Method Principle

The transmitted data is the output of a discrete information source, in particular a character generator, encoded for example by the ASCII code (American Standard Code for Information Interchange), with or without parity check bit.
For values greater than 2, the prime number m is an odd number so the encryption algorithm is applied on some interleaved bits which could originate in different characters. Therefore the same character will be encoded in a different way, depending on the partitioning process, its position into the input stream and the corresponding key symbols. Figure 2 illustrates the partitioning of a binary data stream for encryption on GF (8).

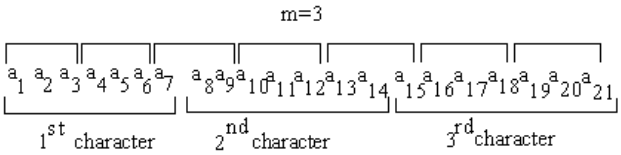

Figure2. Partitioning Process for Encryption on GF (8) of the 7-bit ASCII Code.

The data encryption and the pseudo-random key generation are parallel processes.
A great dimension of the GF could be used to increase the algorithm robustness.
If m>8, than the encryption function is applied on groups of bits loaded from more than one character.
In this way, the proposed encryption algorithm modifies the probabilistic distribution of the initial character set. Near-uniform distributed characters in the encrypted frame result.
The decryption is made similar, using the same chaotic number generator, the same password with the inverse polynomial function given by (2). The encryption system is symmetric because the same password is used to extract the plain text. The coding rate is 1:1, so the transmission data rate is not decreased by encryption.

Encryption/Decryption Algorithms

The encryption algorithm has the following steps:

STEP 1 READ THE PASSWORD
STEP 2 DEDUCE THE INITIAL STATE OF THE CHAOTIC NUMBER GENERATOR
STEP 3 READ THE DATA BIT SEQUENCE OF ONE CHARACTER, SYMBOL OR

	BLOCK OF THE INFORMATION SOURCE
STEP 4	GENERATE A SET OF THREE COMPONENTS OF THE ENCRYPTION KEY
STEP 5	ENCRYPT THE DATA BLOCK
STEP 6	RUN STEP 2 UNTIL THE END OF THE FILE IS REACHED.

The decryption algorithm is designed in the same way only the key components are changed and the GF function used to process the data.
Encryption algorithm implementation in a programming language such as Java or C/C++ has some difficulties given by special unprintable characters resulted from the encoding process (Appendix A).
The control character, with ASCII codes less than 0x.20 could give uncontrollable behavior of the receiver, for example the BREAK condition could be obtained so the communication process is stopped and the text is truncated with information loss.

Therefore we have to use a "character stuffing" method. Each time, a control character results from the encryption code, the ESCAPE code (0x.1B) is introduced first, in order to ignore the significance of that character.

In the same way we have to proceed with the code 0x.34 used for the quotation mark character (") which is used for string delimitation.

Conclusions

The proposed encryption algorithm modifies the probabilistic distribution of the initial character set. Near-uniform distributed characters in the encrypted frame result.

Two JAVA algorithms are implemented for encryption and decryption of text files, with different classes which execute GF arithmetic operations, character-to-ASCII conversion, the encoding and the decoding functions.

The performances of these algorithms are very good. The processing time is reduced by parallel processing, the robustness is ensured by a very long key, the encrypted file probabilistic distribution is near-uniform so the statistic attacks chances are minimized.

The proposed JAVA text encryption algorithm is more efficient than the well-known DES or IDEA algorithms because the encryption key length is theoretically infinite, longer than the message itself, without any periodicity or redundancy.

References

[1] Schneier B. (1996), "*Applied cryptography", second edition*, NY: John Wiley & Sons, Inc.
[2] Scripcariu L. and Duma P. (2004), *"Analysis of Simple Invertible Functions Defined on Galois Fields for Cryptography Use*", Trans. on Electronics and Communications, Vol. 49 (63), Fasc. 2, 2004, ISSN 1583-3380, Ed. Politehnica Timisoara (Romania), pp.55-59.
[3] Luca A. and Vlad A. (2005), *"Generating Identically and Independently Distributed Samples Starting from Chaotic Signals*", In Proc. of the Intern. Symposium on Signals, Circuits and Systems ISSCS 2005, July 14-15, 2005, Vol.1, Iasi, Romania, pp. 227 - 230.
[4] Scripcariu L. and Frunza M.D.(2005), "*A New Image Encryption Algorithm Based on Invertible Functions Defined on Galois Fields*", In Proc. of the Intern. Symposium on Signals, Circuits and Systems ISSCS 2005, July 14-15, 2005, Vol.1, Iasi, Romania, pp. 243 - 246.

Appendix A

The JAVA encryption algorithm is given below. Different GF (8) functions are used. This is a selective encryption algorithm for text files.

Criptcharchaos.java

```
//  Copyright (c) 2005-2010 Faculty of
Electronics and Telecommunications -
Iasi

import java.io.*;

public class Criptcharchaos {
  int joker[]=new int[8];
  int joker3[]=new int[3];
  int joker_e;
  int aaaa[];
```

```
  int num_key[]=new int[1000];
  int num_a[]=new int[1000];
  int b[]=new int[1000];
  int sir[]=new int[1000];
  double x[]= new  double [1000];
  int bia[]=new int[1000];
  int bia3[]=new int[3];
  int e;
  int v[]=new int[7];
  int v7[]=new int[7];
  int joker_v[]=new int[8];
  int joker_v7[]=new int[8];
  int c[]=new int[1000];
  int joker_c;
  Byte aa[]=new Byte[1000];
  Byte by[]=new Byte[1000];
  Integer in[]=new Integer[1000];
static int lungime_key;


public  static void main(String args[])
{
Criptcharchaos         ccc=         new
Criptcharchaos();
String nume_fisier="";
String key="";
try
{
InputStream is= System.in;
DataInputStream        dis1=        new
DataInputStream(is);
System.out.println("\n   Copyright  (c)
2005-2010  Faculty  of  Electronics  and
Telecommunications - Iasi\n");
System.out.println("\t\t\t\t*****");
System.out.println("\nIntroduceti
numele   fisierului   text   (maxim   500
caractere)   pe   care   doriti   sa-l
criptati: ");
nume_fisier = dis1.readLine();
if
(nume_fisier.substring(nume_fisier.leng
th()-4,
nume_fisier.length()).equals(".txt"))
{
}
else
{
System.out.println("Fisierul  nu  e  in
format .txt !" );
System.exit(1);
}
DataInputStream        dis2=        new
DataInputStream(is);
System.out.println("Introduceti   cheia
secreta : ");
key = dis2.readLine();
lungime_key=key.length();
if (key.length() >256)
{
System.out.println("Lungimea      cheii
secrete este prea mare !");
System.exit(1);
      }
      }
catch(IOException e){}
ccc.criptare(nume_fisier,key);
System.out.println("Criptarea   a   avut
loc cu succes !");
}

private void criptare(String nume_fis,
String key)
{
aaaa=criptcharchaos(FileIO.fileIO(nume_
fis),key);
try
{
FileWriter       fos       =       new
FileWriter(nume_fis.substring(0,nume_fi
s.length()-4)+"_criptat.txt");
BufferedWriter      dos      =      new
BufferedWriter(fos);
int i=0;
while (aaaa[i]!=0)
{
dos.write(aaaa[i]);
i++;
}
dos.close();
}
catch(IOException e)
{
System.out.println("Eroare      scriere
fiser "+e);
}
}



public  int[]  criptcharchaos(String  a,
String key)
{
int k=0;
int L=a.length();
int m=3;
int tp=300;
double r=3.9;
int nr_es=tp*L;
int n1min=0,n1max=7;
int n2min=1,n2max=7;
int n3min=1,n3max=6;
by=string2ascii(key);
for (int i=0;i<key.length();i++)
{
num_key[i]=by[i].intValue();
}
for (int i=0;i<lungime_key;i++)
{
in=binarInteger(num_key[i],7);
for(int z=0;z<7;z++)
{
      b[z]=in[z].intValue();
}
for (int j=0;j<7;j++)
{
```

```
k=7*(i)+j;
sir[k]=b[j];
}
}
double      s0=0;
for (int ii=1;ii<=7*lungime_key;ii++)
{
s0=s0+(Math.pow(2,(-ii)))*sir[ii-1];
}
aa=string2ascii(a);
for (int j=0;j<a.length();j++)
num_a[j]=aa[j].intValue();
joker =binar(233,8);
for(int o=0;o<3;o++)
joker3[o]=joker[o];
joker_e=bin2zec(joker3);
joker_e=power3(joker_e,1);
joker_v=binar(sgf3(1,pgf3(1,joker_e)),3
);
for(int s=3;s<8;s++)
{
joker_v[s]=joker[s];
}
for(int s=0;s<8;s++)
{
joker_v7[s]=joker_v[s];
}
joker_c=bin2zec(joker_v7);
for(int i=0;i<L;i++)
{
x[0]=s0;
for(int q=1;q<=tp+1;q++)
{
x[q]=r*x[q-1]*(1-x[q-1]);
}
s0=x[tp+1];
int   k1=(int)Math.round(x[100]*(n1max-
n1min)+n1min);
int   k2=(int)Math.round(x[200]*(n2max-
n2min)+n2min);
int   k3=(int)Math.round(x[300]*(n3max-
n3min)+n3min);
bia=binar(num_a[i],7);
for(int w=0;w<3;w++)
bia3[w]=bia[w];
e=bin2zec(bia3);
e=power3(e,k3);
v=binar(sgf3(k1,pgf3(k2,e)),3);
for(int s=3;s<7;s++)
{
v[s]=bia[s];
}
for(int s=0;s<7;s++)
{
v7[s]=v[s];
}
c[i]=bin2zec(v7);
if (c[i]==0)
c[i]=joker_c;
}
return c;
}
public int bin2zec(int a[]) {
int z=0,doi=2;
int l=a.length;
if(l==0) z=-1;
if(l>0) {
for(int i=0;i<l;i++){
if(a[i] >=1)
a[i]=1;
if(a[i] < 1)
a[i]=0;
z=z+(int)Math.pow(doi,(l-i-1))*a[i];
}
}
return z;
}

public int[] binar(int a, int k)
{
double doi=2;
int
b[]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
a = a%(int)(Math.pow(doi,k)) ;
for(int i=0;i<k+1;i++) {
b[i]=(int)Math.floor(a/(Math.pow(doi,k-
i-1)));
a=a-(int)(Math.pow(doi,k-i-1))*b[i];
}
return b;
}

public Integer[] binarInteger(int a,int
k)
{
double doi=2;
Integer b[]=new Integer[10000];
a = a%(int)(Math.pow(doi,k)) ;
for(int i=0;i<k+1;i++)
{
b[i]=new
Integer((int)Math.floor(a/(Math.pow(doi
,k-i-1))));
a=a-(int)(Math.pow(doi,k-i-
1))*b[i].intValue();
}
return b;
}

public int complem3(int c){
int q=0;
int complem[]={0,1,4,5,2,3,6};
q=complem[c];
return q;
}

public int invgf8(int x){
int y[]={0,0,0,0,0,0,0,0};
y[1]=1; y[2]=5; y[3]=6; y[4]=7; y[5]=2;
y[6]=3; y[7]=4;
int k=0;
if (Math.abs(x)>0)
{
x=x-8*(int)Math.floor(x/8);
k=y[x];
```

```
}
if(x==0)
k=-1;
return k;
}

public int pgf3(int a, int b) {
int y=0;
int pow2to3[][]=
{{0,0,0,0,0,0,0,0},{0,1,2,3,4,5,6,7},{0
,2,4,6,3,1,7,5},{0,3,6,5,7,4,1,2},
{0,4,3,7,6,2,5,1},{0,5,1,4,2,7,3,6},{0,
6,7,1,5,3,2,4},{0,7,5,2,1,6,4,3}};
a=a-8*(int)Math.floor(a/8);
b=b-8*(int)Math.floor(b/8);
y=pow2to3[a][b];
return y;
}

public int power3(int a, int k) {
int power[][]=
{{0,0,0,0,0,0,0,0},{1,1,1,1,1,1,1,1},{1
,2,4,3,6,7,5,1},{1,3,5,4,7,2,6,1},
{1,4,6,5,2,3,7,1},{1,5,7,6,3,4,2,1},{1,
6,2,7,4,5,3,1},{1,7,3,2,5,6,4,1}};
int y=power[a][k];
return y;
}

public int sgf3(int a, int b) {
int sgf[][]=
{{0,1,2,3,4,5,6,7},{1,0,3,2,5,4,7,6},{2
,3,0,1,6,7,4,5},{3,2,1,0,7,6,5,4},
{4,5,6,7,0,1,2,3},{5,4,7,6,1,0,3,2},{6,
7,4,5,2,3,0,1},{7,6,5,4,3,2,1,0}};
int s=sgf[a][b];
return s;
}

public Byte[] string2ascii(String str)
{
      byte b[]=str.getBytes();
      Byte bye[]=new Byte[1000];
      for(int i=0;i<str.length();i++)
      bye[i]= new Byte(b[i]);
      return bye;
}

public Integer[] zec2bin(int a[], int
m)
{
      int l=a.length;
      Integer vi[] = new Integer[l*m];
      Integer bi[] = new Integer[l*m];
      for (int i=0;i<l;i++)
       {
        vi=binarInteger(a[i],m);
for(int j=0;j<m;j++)
{
bi[(i)*m+j]=vi[j];
}
}
return bi;
}


public String conversie(int a[])
{
String str="";
char ch[]=new char[a.length];
for(int i=0;i<a.length;i++)
ch[i]=(char)a[i];
str=new String(ch);
return str;
}
}
```