ANALYSIS OF A CRYPTOSYSTEM BASED ON A MODIFIED RINDJAEL ALGORITHM IMPLEMENTATION

Marian CREȚU[1]

Rijndael, algoritmul de criptare / decriptare ales de NIST (National Institute of Standards and Technology), ca AES (Advanced Encryption Standard), în octombrie 2001, a demonstrat o puternică rezistență la metodele de criptanaliză liniară şi diferențială. În ciuda faptului că este un algoritm simetric de criptare, structura sa este extrem de complexă şi se bazează pe elemente matematice care utilizează câmpuri finite şi polinoame ireductibile peste GF(2^m). Implementarea unei soluții care poate utiliza o bază de date securizată, în care pot fi stocate mai multe perechi de tabele d esubstituție, parametri pentru diferiți paşi ai algoritmului şi un număr mare de chei de criptare, poate oferi o bună soluție de securitate, prin modificarea factorului de difuzie.

The Rijndael encryption / decryption algorithm elected by NIST (National Institute of Standards and Technology) as AES (Advanced Encryption Standard) in October 2001, provided a strong resistance to linear and differential cryptanalysis methods. Despite the fact that it is a symmetric encryption algorithm, his structure is extremely complex and based on mathematic elements using finite fields and irreducilble polynomials over GF(2^m). Implementing a solution that can use a secure database where can be stored multiple pairs of substitution tables, parameters for different algorithm steps and a large number of encryption keys, can provide a good security solution modifying the diffusion factor.

Keywords: Rijndael, AES, S-Box, substitution tables, finite field element.

1. Introduction

In October 2001, NIST (National Institute of Standards and Technology) established the new AES (Advanced Encryption Standard) in order to provide a standard encryption algorithm, strong enough to replace the old DES (Data Encryption Standard) [1]. Used for more than twenty years, DES has proven insecure on some cryptanalytic attacks, and it was replaced by 3 DES (Triple DES) in 1999, published in FIPS PUB 46/3 [2]. Rijndael, the algorithm of two Belgian cryptographers (Joan Daemen from "Proton World" and Vincent Rijmen from "COSIC"), was selected from the 5 finalists of the NIST contest (Table 1) started in 2000, and published as the new AES in FIPS PUB 197 [3], being the

[1] Ph.D. Student, Faculty of Electronics, Telecommunications and Information Technology, POLITEHNICA University of Bucharest, Romania, e-mail: marian.cretu77@yahoo.com

best software implementation for data encryption in order to respect FIPS 140-2 [4]. The resistance against differential and linear cryptanalysis techniques (two of the most powerful cryptanalytic tools) was decisive in its election as AES [5].

Table 1

AES finalists

Nr.	Algorithm	Designer
1	**MARS**	IBM
2	**Twofish**	Bruce Schneier and friends
3	**Rijndael**	Joan Daemen and Vincent Rijmen
4	**RC6**	RSA
5	**Serpent**	Ross Anderson, Eli Biham and Lars Knudsen

The main purpose of new AES was to replace the 3DES algorithm for government use, secure transactions over the internet and other encryption needs. A wide variety of Rijndael algorithm implementations showed up to satisfy different types of application.

There were implementations who seeks to maximize the throughput [2], [5], [6], to minimize the power needed for computation [7] or to minimize the hardware circuitry [8], [9], [10], [11].

2. The Rijndael/AES algorithm

Like many other block ciphers, Rijndael can be used in several modes, such as ECB (Electronic Codebook), CBC (Cipher Block Chaining), or CFB (Cipher Feedback). AES is a symmetric encryption algorithm meaning that encryption and decryption are performed by essentially the same steps, in reverse order. Despite the fact that Rijndael algorithm has a variable block length, using keys with on 128, 192 or 256 bits length, the standard refers only to the variant of 128 bits block length. The corresponding number of rounds for each key length is 10, 12, or 14 rounds, respectively. For each round, starting from the original key, a different "round key" is computed. In this paper, we will study the case of a 128-bit key length on 10 rounds.

The four steps in each round of encryption are *SubBytes*, *ShiftRows*, *MixColumns*, and *AddRoundKey*. The input block is processed by *AddRoundKey* step*,* before the first round; the last round skips the *MixColumns* step. All rounds are identical, except each one uses a different round key, and the input of one round is the output of the previous round (Fig. 1).

The last three steps are linear (*ShiftRows*, *MixColumns*, and *AddRoundKey*), meaning that the output 128-bit block for one of these steps is the linear combination (bitwise, modulo 2) of the outputs for each separate input bit.

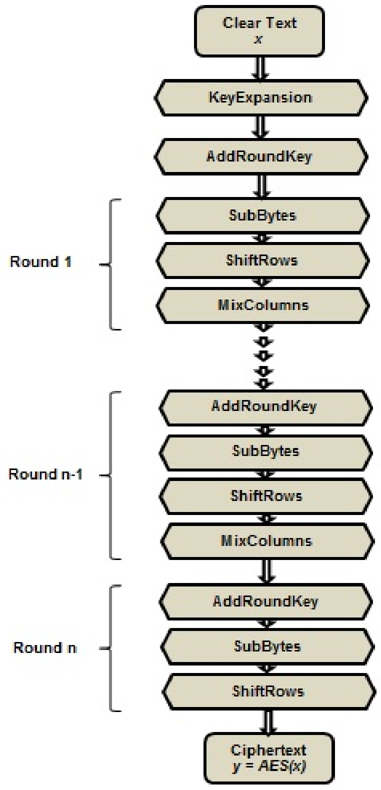


Fig. 1. The AES symmetric key cryptosystem

According to [12], the four steps correspond to the round key addition step (*AddRoundKey*), the non-linear step (*SubBytes*), the dispersion step (*ShiftRows*) and the diffusion step (*MixColumns*). They are described as follows:

AddRoundKey is the first step, where a round key is added to the State matrix using a simple bitwise XOR operation which in the field GF(2^8) is a sum. Each round key is obtained from the key schedule.

SubBytes is a non-linear byte substitution that operates independent on each byte of the state matrix. It uses a substitution table (S-Box) which is invertible and is constructed by composing two transformations in GF(2^8), an inversion and an affine function:

1. Inversion in the GF(2^8) field, modulo the irreducible polynomial:

$$q(x) = x^8 + x^4 + x^3 + x + 1 \tag{1}$$

2. Affine transformation defined by the relation:

$$S = AX^{-1} + c \tag{2}$$

where A is a 8x8 fixed matrix and *c* is a 8x1 vector-matrix [5].

$$\begin{pmatrix} s_7 \\ s_6 \\ s_5 \\ s_4 \\ s_3 \\ s_2 \\ s_1 \\ s_0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_7 \\ c_6 \\ c_5 \\ c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

The 7th bit is the most significant within all bit operations modulo 2.

ShiftRows is the third step where all bytes in the last three rows of the state matrix are shifted with different numbers of bytes. The first row, row 0, is not rotated; row 1 is rotated to the left by 1 byte; row 2 is rotated to the left by 2 bytes and row 3 is rotated to the left by 3 bytes.

MixColumns is a transformation that operates in a column-by-column mode on the State matrix, treating each column as a four-degree polynomial on GF(2^8). These polynomials are multiplied *mod* $(x^4 + 1)$ with a fixed polynomial $f_a(x)$ as follows:

$$f_a(x) = 03 * x^3 + 01 * x^2 + 01 * x + 02 \tag{3}$$

The transformation is invertible because this polynomial is coprime to $(x^4 + 1)$. This affine transformation can be written as a matrix multiplication.

We presume that:

$$z'_c(x) = f_a(x) \otimes z_c(x); \quad 0 \le c \le 3 \tag{4}$$

for all the columns in the state matrix. As a result this multiplication is the 4 bytes in the *c* column that are replaced by the following values (for *c* = 0, 1, 2 and 3):

$$\begin{aligned} z_{0,c} &= 02 * z_{0,c} \otimes 03 * z_{1,c} \otimes z_{2,c} \otimes z_{3,c} \\ z_{1,c} &= z_{0,c} \otimes 02 * z_{1,c} \otimes 03 * z_{2,c} \otimes z_{3,c} \\ z_{2,c} &= z_{0,c} \otimes z_{1,c} \otimes 02 * z_{2,c} \otimes 03 * z_{3,c} \\ z_{3,c} &= 03 * z_{0,c} \otimes z_{1,c} \otimes z_{2,c} \otimes 02 * z_{3,c} \end{aligned} \tag{5}$$

Applying to the state matrix these four transformations in this order, we obtain a complete round:

round = {SubBytes, ShiftRows, MixColumns, AddRoundKey}

The final round is obtained by performing only three of four transformations to the state matrix, in the following order:

final round = {SubBytes, ShiftRows, AddRoundKey}

The Rijndael encryption / decryption algorithm consists in an initial application of the *AddRoundKey* operation, followed by 9, 11, 13 rounds and concluded with a final round. Decryption in Rijndael algorithm is performed by applying the inverse of all the transformations in reverse order, to obtain the plaintext.

The AES algorithm uses a particular version of Galois field of 8-bit bytes. The bits are coefficients of a polynomial and multiplication is modulo the irreducible polynomial q(x) with the addition of coefficients modulo 2.

2.1 *Finite field addition*

For A as root of $q(x)$, the standard polynomial basis is [$A^7, A^6, A^5, A^4, A^3, A^2, A^1, A^0$]. Addition and subtraction are equivalent to an exclusive-or operation for the bytes where the field elements are represented. The symbol "$\otimes$" stands for the addition operation for finite field elements. The following three equations (6, 7, and 8), are equivalent:

1. in polynomial notation

$$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) \equiv x^7 + x^6 + x^4 + x^2 \quad (6)$$

2. in binary notation

$$\{01010111\} \otimes \{10000011\} \equiv \{11010100\} \quad (7)$$

3. in hexadecimal notation

$$\{57\} \otimes \{83\} \equiv \{d4\} \quad (8)$$

2.2 *Finite field multiplication*

This operation is more complex than addition and is gained by multiplying the polynomials of the elements concerned, achieving in the result the powers of x in a complete equation. Because each polynomial can have powers of x from 0 up to 7, the result can have the x powered up to 14 a single byte will no longer be sufficient for its representation.

In Rijndael algorithm, Rijmen and Daemen solved this problem by replacing the multiplication result with the remainder polynomial after division by $q(x)$ (an eighth order irreducible polynomial) [1]. Since the $q(x)$ polynomial have powers up to 8 it cannot be represented on a single byte and the solution is to write it in binary or hexadecimal representation, as 1{00011011} or 1{1b}. The product {57} • {83} ≡ {c1} illustrate the finite field multiplication process (where "•" is the finite field multiplication operator) [12]:

$$(x^6+x^4+x^2+x+1)\bullet(x^7+\mathrm{x}+1)=$$
$$=x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+1 \qquad (9)$$

This result is divided by $q(x)$ and subtracted by $q(x) \bullet x^5$ to obtain the intermediate reminder:

$$(x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+1)-(\mathrm{q}(x)\bullet x^5)=$$
$$=x^{11}+x^4+x^3+1 \qquad (10)$$

The operation above is repeated with intermediate reminder, in order to obtain the final reminder:

$$(x^{11}+x^4+x^3+1)-(\mathrm{q}(x)\bullet x^3)=x^7+x^6+1 \qquad (11)$$

Multiplication is an associative operation and, by definition, there is a neutral element {01}. The extended Euclidean algorithm can be used to compute polynomials $f_a(x)$ and $f_c(x)$ for any binary polynomial $f_b(x)$ of degree less than 8, such that:

$$f_b(x)\bullet f_a(x)\otimes m(x)\bullet f_c(x)=1 \qquad (12)$$

$$f_a(x)\bullet f_b(x) \bmod q(x)=1 \qquad (13)$$

This demonstrates that the polynomials $f_a(x)$ and $f_b(x)$ are inverse one for the other. Even more, the associativity property is proved by the relation:

$$f_a(x)\bullet(f_b(x)\otimes f_c(x))=f_a(x)\bullet f_b(x)\otimes f_a(x)\bullet f_c(x) \qquad (14)$$

The set of 256 byte values, following the demonstration above, with the XOR as addition and multiplication, has the structure of the finite field GF(2^8) [12].

2.3 *Finite field multiplication using substitution tables*

When the finite field elements are multiplied to produce the list of their powers (g^p), all 255 elements in the field different from "0" element will be generated. For p = 256, the original field element recurs, indicating that g^{255}={01} [5]. Using the p values as logarithms the algorithm changes multiplication into addition. For two elements in the field $x = g^\delta$ and $y = g^\theta$, the product $x \bullet y = g^\delta + g^\theta$. The log table implemented in Rijndael algorithm is listing the power of the generator for each finite field element. The reverse table is used to look up the product element. The two initial elements can reach values up to 255 and their sum will be greater than 255 but this problem is solved by subtraction of 255 due to equality g^{255} = {01}.

2.4 *Irreducible polynomials*

The main application of irreducible polynomials is in elliptic curve and pairing-based cryptography [13], [14]. In a communication protocol where there are shared elements between users, it is important for all participants to know the irreducible polynomial. The inclusion of the defined parameters for the irreducible polynomial as part of the information that each participant needs is not difficult. A participant with low computational power might be in advantage by the irreducible polynomial in its need of an optimal choice, in other way, a change of the polynomial representation being necessary [6].

Applications such as pseudorandom number generators with feedback shift registers, arithmetic of finite fields or discrete logarithm uses irreducible polynomials [15].

An irreducible polynomial always has a reciprocal irreducible polynomial. A primitive polynomial always has a reciprocal primitive polynomial. For *a* denoted as root the entry following *j* is the minimum polynomial of a^j. The polynomial exponent can be determined with the condition of *a* as a primitive element of $GF(2^m)$ and the formula:

$$e = (2^{m-1})/GCD(2^{m-1}, j) \tag{15}$$

where *e* is the exponent to which the minimum function of a^j belongs.

A procedure to determine if a function f(x) of degree m is primitive or not, follows the 3 steps:

1. $1, X, X^2, X^4, \ldots, X^{2^\wedge(m-1)}$ are residues of *f(x)* and they are obtained by modulo *f(X)*.
2. In order to form the residue of X^{2m-1}, these residues are multiplied and reduced modulo *f(X)*. The polynomial is rejected for a result ≠ 1. The test is continued if result = 1 is obtained.
3. X^r (where *r* takes values between 0 and 2^{m-1}) is formed by multiplying an appropriate combination of the residues obtained in step 1. The polynomial is primitive if none of these has 1 as result [16].

3. Implementation of the modified versions of Rijndael algorithm

There are many implementations of Rijndael algorithm in order to be integrated in other applications. Some of them use the replacement of the constants values implemented, the coefficients used in the *ShiftRows* step or *MixColumns* step [17], the affine transformation in the S-Box, the S-Boxes switching [18] or the replacement of the irreducible polynomial. According to De Wang, Shi-Liang Sun and Zhuo-Hui Xian in [19, 20] there are 30 irreducible polynomials in $GF(2^8)$ that can be implemented in Rijndael to construct suitable S-Boxes.

In this study, I analyze the correlation factor between encrypted texts where the algorithm keeps the original S-Boxes but changes both the *ShiftRows* step and the *MixColumns* step parameters.

ShiftRows step provides diffusion by mixing data within rows. In the modified Rijndael implementation (MRI) row zero of the state is shifted 3 bytes, row 1 is shifted 2 bytes, row 2 is shifted 1 byte, and row 3 is unchanged. It is actually the reverse order of the shifting process from the original Rijndael implementation (ORI).

MixColumns step also provides diffusion by mixing data within columns. To calculate the MixColumns transformation, the columns of the current state are polynomials over $GF(2^8)$. The coefficients of the polynomial are elements of $GF(2^8)$. In the original Rijndael implementation, each column (each polynomial) is multiplied by the polynomial $a(x)$ mod (x^4+1):

$$a(x) = 02x^3+03x^2+01x+01 \tag{16}$$

The inverse of this polynomial, used in decryption process is:

$$a^{-1}(x) = 14x^3+11x^2+13x+9 \tag{17}$$

In MRI, the polynomial coefficients for both *MixColumns* and *InvMixColumns* steps changes as follow:

$$b(x) = 02x^3+01x^2+01x+03 \tag{18}$$

The inverse of this polynomial, used in decryption process is:

$$b^{-1}(\mathrm{x}) = 14x^3+9x^2+13x+11 \tag{19}$$

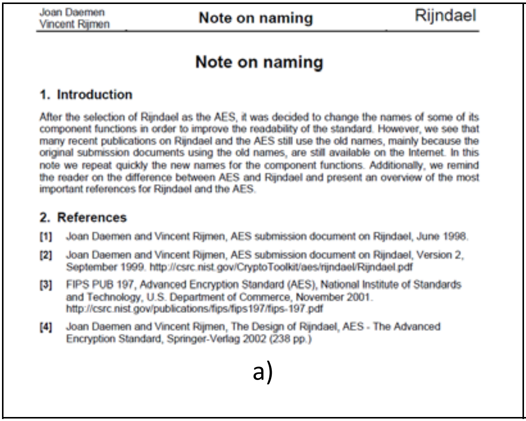
With the proposed polynomial implementations I analyzed the correlation between the plaintext (PT) and cyphertext obtained running ORI, PT and the cyphertexts obtained running variants of MRI and cyphertext obtained running ORI and cyphertexts obtained running variants of MRI as in *Table 2.*

Table 2

Rijndael implementations

ORI	Original *ShiftRows* step implementation Original *MixColumns* step implementation
MRI 1	Original *ShiftRows* step implementation Modified *MixColumns* step implementation
MRI 2	Modified *ShiftRows* step implementation Original *MixColumns* step implementation
MRI 3	Modified *ShiftRows* step implementation Modified *MixColumns* step implementation

Running the Rijndael algorithm implementations described above and using the same plaintext and the same encryption key we can get different results for the cyphertext, as we can see in Fig.2-6 (a): A better view of the character frequency is given by the histograms realized for PT, ORI and MRI 1-3 files as in Fig.2-6 (b).



Joan Daemen
Vincent Rijmen

Note on naming

Rijndael

Note on naming

1. Introduction

After the selection of Rijndael as the AES, it was decided to change the names of some of its component functions in order to improve the readability of the standard. However, we see that many recent publications on Rijndael and the AES still use the old names, mainly because the original submission documents using the old names, are still available on the Internet. In this note we repeat quickly the new names for the component functions. Additionally, we remind the reader on the difference between AES and Rijndael and present an overview of the most important references for Rijndael and the AES.

2. References

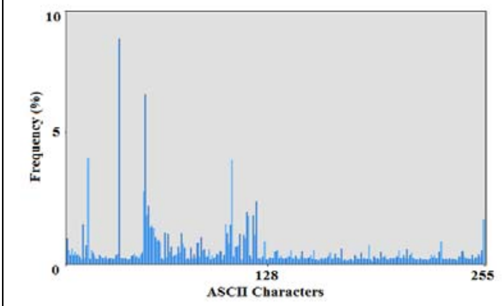
[1] Joan Daemen and Vincent Rijmen, AES submission document on Rijndael, June 1998.

[2] Joan Daemen and Vincent Rijmen, AES submission document on Rijndael, Version 2, September 1999. http://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael.pdf

[3] FIPS PUB 197, Advanced Encryption Standard (AES), National Institute of Standards and Technology, U.S. Department of Commerce, November 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[4] Joan Daemen and Vincent Rijmen, The Design of Rijndael, AES - The Advanced Encryption Standard, Springer-Verlag 2002 (238 pp.)

a)

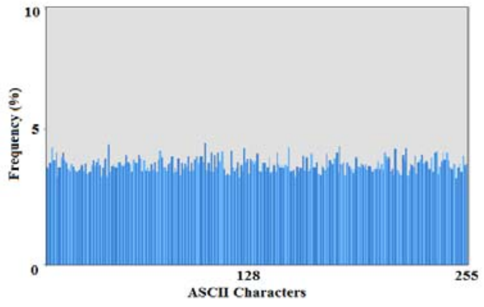


b)

Fig.2. PT (a) and PT histogram (b)

```
M3IÇ¹áý4 †«þ†|v +„¶Ê¤ÝÙìħý²j  TÑó-á®←åõmQþ®‹ ÿwÅé┐
V 1Åßh#½¿¨ôÄÈóG²-ÿßaÐ¢UàŒZ█®Å»vcÅ‰,mÄS-à01¢fÈ[ˆi:yÁ¹eRr‼—œ-
u← ÇBá5 ¾i×ü utž.+Ø,átã>Å‹"P+Ê
B ÌØ,átã>Å‹"P+ÊB ÌØ┼óÞ¹S Ê<E
P/PÅÛ+Z)1CzvKã+6 MÆ o®|#å‹ŽÎ1ÊbhÁêv˜ûOpè‼Í»›Õ´„] F°#"Ê1BwÅ'wö
\®ÜZ£5úýû ó¶‡†Ù[ã»…ãæ]Îêa- ┬Ò,ìóCx·´UÐß¼!
ã2)z"«"©ßÌ┬/¡ãÞ‼Åú┐ÅÕaO¬Û¤Ô|¦¢q ‼"m°1(+)'wYóZ%2@¥Á?HÞ ì+▯Ì>< @Ý
┘Ÿy: @":¡└Å‼ÅR»┘¯®p¤(K•«[µ•G¨;À°IÙ Å‡Nç¯KŽ?Åë@R┼™Á UúW † Ÿ
Î§SÛµ¼¾¬,HKÌ e3qÎœ ³ó¤ßD!"Õ.åc×Êb„‼_¾æ†ì NÌ¡J—
«ýDØŒ ÑÉB,Æ°¹ÞE @öÅ-¬mY…™ @‡²Ù¯┘!m,÷7" ì,
Wã·+Nûì þó-¡ëó8OX"! —¬P®└ Œ┴▯Å^ÞW
ûY,e|Ó¼æ:H* ` ±óó:Ÿ• ÈÚ¼éú1┐©ì¶Êæk!³ ,rõ(Ñ˜o®³›‹█K
Ð[Is-ö[=ý5¶1'nC¤ Ô¼█ß      iWV‡(█"P®‡†œMI$¤´„—ë†¾Ê¤ XŒ[
mÆ³ `E¶X,; p¿"££ÕŽÕ†PI‹,5>°;ò ĵýMh¼¤O¿     ÊV›*Ú‼,j ├";©r*
£„ëäH±g ò…·†ãä@ÅI£ÅÕ┤>i7-;┼z´
wU$ `¢¯s-È‼„²ó┘òè|ó'ó,¶W˜´;}%xx `¶s —└'uL= \óB"‼     T¯-Ê%$"
ÅYb—¹Å ┘ì[Ûy^X┴5Avã¾?ÝeÈnÖ¦┼cå· ´ÙÅMuRÌÚ¹_¼´¨ç«'cž‡—åÞVGev›
Œ9Dái",4,¯Ê œ9‹]!Å?q´SW4 £¹áe%³¹³|›óM‼QDNÊ$O$ñÆÇ'r-åúc÷eFã%
|_úO zbÐÛ"Bx±|f¼ñqyµe O§IùÊV¨® YÓÅ§Ñ[┘O÷¶  `@£%
"4æìÜ ÅÌ1 'L¶¯kŒ)-O£V÷Ìcög&Gr-·XE»|ì˜"@#
¼_,im þ¦ì¢|f‡ ‼ k(Å±÷µ 49a´ßB átüæ€˜"]\æò¢ë┼…W°Å•ìEQ' <¤¼b┬dÌÅÈF
€Q®_®´ßs…°RìÕGVã§Ûk¨åUYó²?†ÇŸ é
rÅ┼ìÜÐÓÐ) |'ã/°q┬Œãõ¢¯Ê┴>¼²Ma"k·1ß¢¹V¥┬   (ßfÅÄO[•7‼%
```

a)



b)

Fig.3. ORI cyphertext (a) and ORI histogram (b)

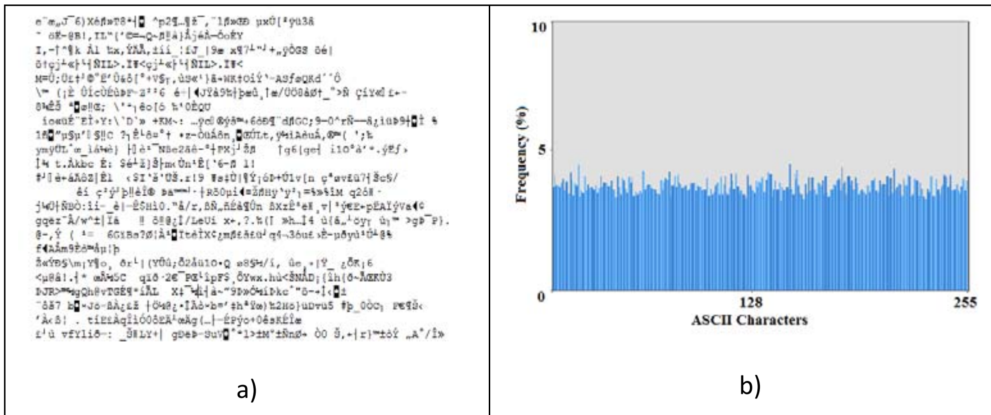


Fig.4. MRI 1 cyphertext (a) MRI 1 histogram (b)

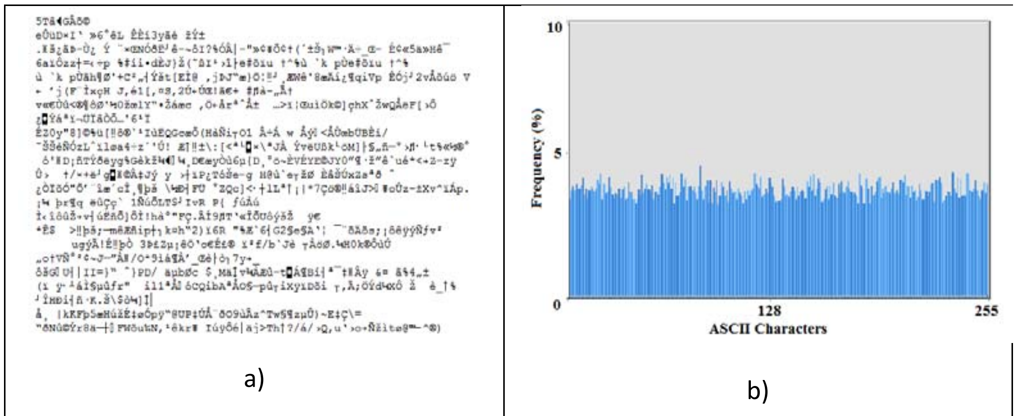


Fig.5. MRI 2 cyphertext (a) MRI 2 histogram (b)

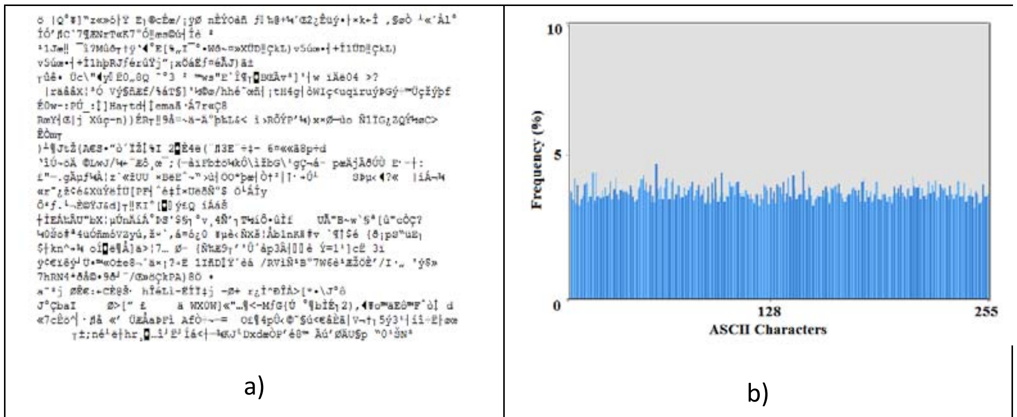


Fig.6. MRI 3 cyphertext (a) MRI 3 histogram (b)

Their histograms illustrate the uniformity of character occurrence after the encryption processes. The frequency peaks from plaintext histogram disappear in the cyphertexts histograms.

3.1 *ASCII characters frequency in cyphertexts*

The encryption process changes the characters frequency in cyphertexts using complex transformations. The characters with greatest frequency for the plaintext (PT), for the cyphertext obtained using the original Rijndael implementation (ORI) and for the modified Rijndael implementation variants (MRI 1-3) are presented in *Table 3*.

Only the most relevant ten percent from ASCII and extended ASCII characters from PT, ORI and MRI is listed in *Table 3*.

Table 3

ASCII characters frequency

PT			**ORI**			**MRI 1**			**MRI 2**			**MRI 3**		
Char	***Count***	***Freq. (%)***	***Char***	***Count***	***Freq. (%)***	***Char***	***Count***	***Freq. (%)***	***Char***	***Count***	***Freq. (%)***	***Char***	***Count***	***Freq. (%)***
' '	70543	6.73	'`'	5007	0.48	'’'	4941	0.47	**'P'**	5019	0.48	**'4'**	5197	0.50
'/'	22843	2.18	'“'	4802	0.46	**'Ò'**	4752	0.45	**'ó'**	4718	0.45	**'Æ'**	4821	0.46
't'	19604	1.87	**'Ú'**	4747	0.45	'—'	4740	0.45	**'8'**	4716	0.45	'ˆ'	4731	0.45
'2'	18324	1.75	**'x'**	4741	0.45	**'7'**	4671	0.45	'±'	4708	0.45	','	4729	0.45
'n'	16062	1.53	**'Ô'**	4729	0.45	**'H'**	4663	0.45	'•'	4695	0.45	**'Ñ'**	4710	0.45
'o'	15015	1.43	**'k'**	4650	0.44	'•'	4588	0.44	','	4648	0.44	';'	4668	0.45
'ÿ'	14095	1.34	**'í'**	4628	0.44	**'q'**	4571	0.44	'='	4640	0.44	**'X'**	4660	0.44
'a'	12312	1.17	**'h'**	4603	0.44	'†'	4548	0.43	'“'	4617	0.44	':'	4631	0.44
'd'	12190	1.16	**'ë'**	4568	0.44	**'ö'**	4540	0.43	**'ø'**	4594	0.44	'>'	4623	0.44
'4'	11833	1.13	**'Í'**	4564	0.44	**'Đ'**	4531	0.43	**'í'**	4587	0.44	'+'	4607	0.44
'5'	11208	1.07	**'ñ'**	4559	0.44	**'Û'**	4527	0.43	**'â'**	4576	0.44	**'ß'**	4546	0.43
'<'	9779	0.93	**'d'**	4550	0.43	**'ò'**	4518	0.43	**'u'**	4572	0.44	**'í'**	4532	0.43
'b'	9649	0.92	**'ó'**	4548	0.43	**'M'**	4511	0.43	'½'	4566	0.44	**'Û'**	4527	0.43
'j'	9279	0.89	'°'	4543	0.43	'¶'	4504	0.43	'?'	4540	0.43	'»'	4519	0.43
'>'	9239	0.88	**'ì'**	4542	0.43	**'K'**	4497	0.43	'¦'	4536	0.43	**'a'**	4512	0.43
's'	9099	0.87	'ª'	4533	0.43	**'Ü'**	4494	0.43	**'G'**	4522	0.43	**'f'**	4506	0.43
'l'	8905	0.85	**'€'**	4533	0.43	'('	4477	0.43	','	4516	0.43	'='	4494	0.43
'R'	8534	0.81	'¡'	4532	0.43	**'Ç'**	4473	0.43	**'ï'**	4498	0.43	**'J'**	4488	0.43
'6'	8300	0.79	**'ä'**	4502	0.43	**'j'**	4472	0.43	**'µ'**	4494	0.43	','	4477	0.43
'm'	8069	0.77	**'0'**	4484	0.43	**'é'**	4466	0.43	'˜'	4492	0.43	'²'	4476	0.43
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
'Ý'	1113	0.11	'—'	3588	0.34	'''	3636	0.35	'('	3545	0.34	**'y'**	3630	0.35
'¬'	1030	0.10	**'u'**	3562	0.34	'›'	3575	0.34	**'j'**	3540	0.34	**'s'**	3580	0.34
'»'	882	0.08	**'ø'**	3528	0.34	**'Œ'**	3495	0.33	'['	3524	0.34	'÷'	3520	0.34

As we can see in *Table 3*, for different characters of original Rijndael implementation and for the modified Rijndael implementations, the character frequencies are very close.

3.2 *Cyphertexts correlations*

The cyphertexts resulted using our implemented version of the Rijndael algorithm have a correlation degree. The correlation coefficient is a quantity revealing the quality of a least square fitting with the original data. To determine a correlation coefficient r of a set of q data points (x_i, y_i) we consider the formula:

$$r = \frac{n\sum x_i y_i - (\sum x_i \sum y_i)}{\sqrt{(n\sum {x_i}^2 - (\sum x_i)^2)\,(n\sum {y_i}^2 - (\sum y_i)^2)}} \tag{20}$$

For cyphertexts resulted in our simulation with the modified versions, x_i and y_i takes the count values partially described in *Table 3* while n is the number of characters in the ASCII and extended ASCII code. The correlation factor values calculated for PT, ORI and MRI 1-3 are listed in *Table 4*.

Table 4

Correlation factors

File	PT	ORI	MRI 1	MRI 2	MRI 3
PT	ac	0.010084	0.017858	-0.00849	0.027894
ORI	-	ac	0.010852	-0.02976	-0.04059
MRI 1	-	-	ac	-0.03063	0.021702
MRI 2	-	-	-	ac	0.038238
MRI 3	-	-	-	-	ac

Diagonal represents autocorrelation (ac) and is equal to 1. Changing the order of the elements (switching x_i and y_i) does not change the resulted correlation coefficients. The correlation coefficient may take on any value between plus and minus one. The positive correlation coefficient means that as the value of one variable increases, the value of the other variable increases; as one decreases the other decreases. A negative correlation coefficient indicates that as one variable increases, the other decreases, and vice-versa.

4. Implementation of a database for substitution tables, encryption / decryption keys and other parameters

The usual implementation of Rijndael algorithm in symmetric ciphers uses only one secret key for encryption and decryption processes, and one pair of substitution tables (the direct S-Box and the related inverse of it).

The possibility to extend the original Rijndael algorithm using a large number of substitution tables pairs stored in a local database, can provide for each pair (S-Box and inverse S-Box), a new variant of the algorithm. Another table in the database, containing secret keys, can be stored in the same way as the substitution tables are stored. Their combination can increase the number of encrypted texts for the same plaintext.

In the proposed cryptosystem, I implemented four pseudorandom number generators (PRNGs): one for the secret keys table, one for the substitution tables, another one for the *ShiftRows* step parameters and the last one for the *MixColumns* step parameters. The algorithm will obtain the identification number of key (ID_k), the identification number of substitution table pair (ID_s), the identification number of *ShiftRows* step (ID_{sr}) and the identification number of *MixColumns* step (ID_{mc}) from the database, to be used by the algorithm engine (AE) in the encryption process. For *n* secret keys, *n* pairs of substitution tables, two variants of *ShiftRows* step and two variants of *MixColumns* step there are a number of $4n^2$ cyphertext variants that can be created for the same clear text.

The encryption process (Fig. 7) is realized by the algorithm engine in 5 steps:

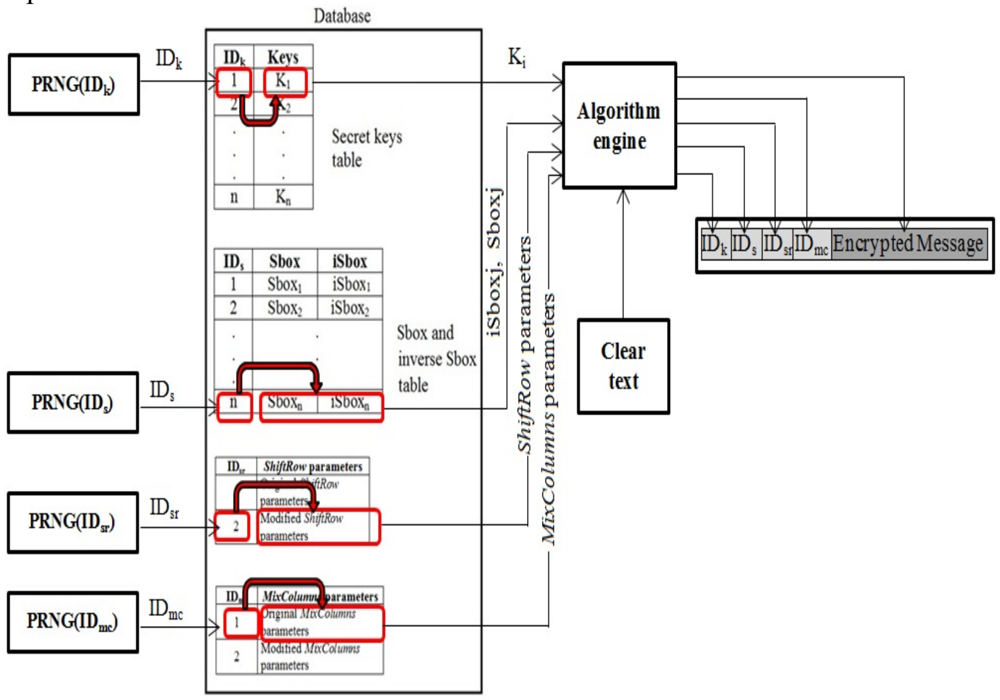


Fig. 7. Encryption process

1) the four PRNGs will generate the random numbers identified as ID_k, ID_s, ID_{sr} and ID_{mc} indexes in the local database;
2) AE read the encryption key with the related identification number ID_k, from the local database;
3) AE read the substitution table pair with the related identification number ID_s, from the local database;

4) AE read the *ShiftRows* step variant with related identification number ID_{sr}, from the local database;
5) AE read the *MixClumns* step variant with related identification number ID_{mc}, from the local database;
6) AE use the key, substitution table pair, *ShiftRows* step variant, *MixColumns* step variant to encrypt the plaintext;
7) AE add ID_k, ID_s ID_{sr} and ID_{mc} and encrypted message in a data package to be sent over the network.

ID_k, $ID_{s,}$ ID_{sr} and ID_{mc} selected from the database are necessary in the decryption process and must be embedded in the data package sent over the network from one entity to another (Fig. 8).

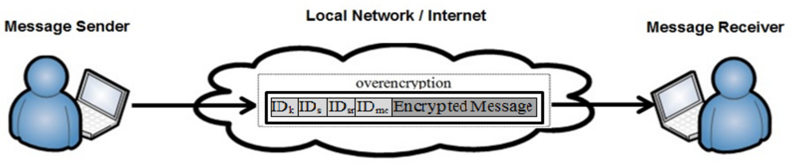


Fig. 8. Data package transmission

For an increased security, data packages may be over-encrypted using a predefined key and a predefined pair of S-Boxes from the local database, or another trusted algorithm (e.g. a public key encryption algorithm). The main condition using this system is that all entities implied in the message exchange process must have the same local database and encryption/decryption engine.

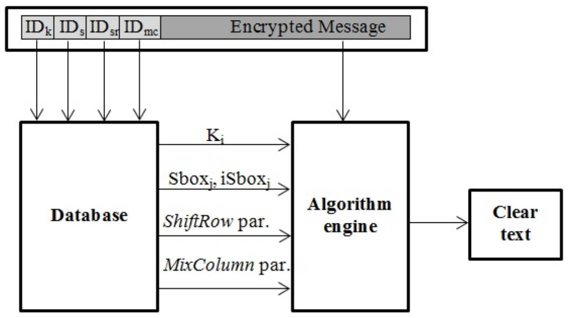


Fig. 9. Decryption process.

The decryption process (Fig. 9) is made in the reverse order of the encryption process, except for the ID_k, ID_s, ID_{sr} and ID_{mc} generation and is realized by the algorithm engine in 6 steps:

1. AE read the ID_k, ID_s, ID_{sr} and ID_{mc} indexes from data packages sent over the network;
2. AE read the decryption key with the related identification number ID_k, from the local database;

3. AE read the substitution table pair with the related identification number ID_s, from the local database;
4. AE read the *ShiftRows* step variant with the related identification number ID_{sr}, from the local database;
5. AE read the *MixColumns* step variant with the related identification number ID_{mc}, from the local database;
6. AE use the key, substitution table pair, the *ShiftRows* step variant and the *MixColumns* step variant to decrypt the cyphertext.

5. Conclusion

The resistance provided by the Rijndael algorithm to linear and differential cryptanalysis was decisive for its election as the new AES, back in 2001.

Once the NIST election was announced, many significant cryptanalysts developed very complex attack methods on AES. A related key attack on AES-256 with complexity 2^{119} is described in [21], following other successful attacks with a complexity of 2^{96} for one out of every 2^{35} keys, described in 2009 by Biryukov, Khovratovich and Nikolic [22].

The use of a modified Rijndael algorithm engine, that can change the encryption key, the pairs of substitution tables, the *ShiftRows* step variant and the *MixColumns* step variant for every new encryption process, has the role to change the diffusion factor for the implemented algorithm and to obtain four different cyphertexts, using the same plaintext and encryption key. For a large number of keys and pairs of substitution tables stored in the local database, and because of the four pseudorandom number generators implemented, the probability to use twice the same combination of key, substitution table pair, *ShiftRows* step variant and *MixColumns* step variant is extremely low.

A notable advantage of the proposed cryptosystem is the security of data stored in database (e.g. MySQL, Postgres, Oracle) on the local machine. The database access can be made using predefined user and password and data can be stored in a secure manner, using the internal mechanism of database, making them available only for the authorized applications.

Another advantage is that encryption / decryption key no longer needs a separate transmission channel on every process. A secure solution for data exchange is necessary only when the entire database is shared to legitimate users.

R E F E R E N C ES

[1] NIST Specification for the Advanced Encryption Standard (AES). Technical Report FIPS PUB 197, November 2001;

[2] FIPS PUB 46-3 (Federal Information Processing Standard Publication). Available online at: http://csrc.nist.gov/publications/fips /fips46-3/fips46-3.pdf;

[3] FIPS-197: Advanced Encryption Standard, November 2001. Available online at: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf;

[4] FIPS PUB 140-2: Security Requirements For Cryptographic Modules. Available online at: http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf;

[5] *Daemen, J., Rijmen, V.*: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer Verlag (2002);

[6] IEEE Computer Society, New York, USA. IEEE Standard Specifications for Public-Key Cryptography. IEEE Std. 1363: 2000. Available online at: http://grouper.ieee.org /groups/1363;

[7] *Sumio M., Akashi S.*, An optimized S-box circuit architecture for low power AES design. In CHES2002, LNCS 2523, pages 172—186, 2003;

[8] *Pawel C., Kris G.*, Very compact FPGA implementation of the AES algorithm. In C.D. Walter et al., editor, CHES 2003, LNCS 2779, pages 319—333, 2003;

[9] *Atri R., Pradeep K. D., Charanjit S. J., Vijay K., Josyula R. R., and Pankaj R.*, Efficient Rijndael encryption implementation with composite field arithmetic. In CHES2001, LNCS 2162, pages 171—184, 2001;

[10] *Satoh A., Morioka S., Takano K., and Seiji Munetoh.*, A compact Rijndael hardware architecture with S-box optimization. In Advances in Cryptology - ASIACRYPT 2001, LNCS 2248, pages 239—254, 2001;

[11] *Johannes Wolkerstorfer, Elisabeth Oswald, and Mario Lamberger,* An ASIC implementation of the AES S-boxes. In CT-RSA 2002, LNCS 2271, pages 67—78, 2002;

[12] *Daemen, J. and V. Rijmen*, (2002). The design of Rijndael: AES–The Advanced Encryption Standard. Springer-Verlag.FIPS Pub. 197: Specification for the AES (2001);

[13] *Hankerson D., Menezes A., and Vanstone S.*, Guide to Elliptic Curves Cryptography. Springer, 2004;

[14] *Barreto, S. Galbraith, C. and Colm O'hE., Scott M.*, Efficient pairing computation on supersingular abelian varieties. Designs, Codes and Cryptography, 2007. Available online at: http://eprint.iacr.org/2004/375;

[15] *Copersmith, D.*, Fast evaluation of logarithms in fields of characteristic two. IEEE Trans. Info. Theory 30 (1984), 587-594;

[16] *Peterson W. W.*, Error-Correcting Codes, MIT Press (1970), pages 251 -155;

[17] *Pimpale P., Rayarikar R., Upadhyay S.*, Modifications to AES Algorithm for Complex Encryption. IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.10, October, 2011.

[18] *Cretu, M., Apostol, C-G.*, A modified version of Rijndael algorithm implemented to analyze the cyphertexts correlation for switched S-Boxes. 9th International Conference on Communication COMM 2012, Conference Proceedings, pages 331-335, 2012;

[19] *Wang D., Sun S-L.*, "Replacement and structure of S-Boxes in Rijndael," Computer Science and Software Engineering, 2008 International Converence, pp. 782-784, December 2008;

[20] *Xian Z-H., Sun S-L.*, "Study on test for structure of S-Boxes in Rijndael," Education Technology and Software Engineering (ETCS), 2008 Second International Workshop, pp. 84-86, March 2010;

[21] *Biryukov, A., Khovratovich, D.*, Related-key Cryptanalysis of the Full AES-192 and AES-256, IACR ePrint report 2009/317, 2009. Available online at: http://eprint.iacr.org/2009/317;

[22] *Biryukov, A., Khovratovich, D., Nikolic, I.*, Distinguisher and related-key attack on the full AES-256. In CRYPTO'09, LNCS. Springer Verlag.