Modified Advanced Encryption Standard

Luminița SCRIPCARIU and Mircea-Daniel FRUNZĂ
Technical University "Gheorghe Asachi" of Iasi, Romania
11 Carol I Blvd, RO – 700506 Iasi
luminita.scripcariu@gmail.com,mircea.frunza@gmail.com

***Abstract* — Information security becomes an important issue of the communication networks. Different cryptographic algorithms are used to ensure the confidentiality of the transmitted information: public-key algorithms, such as RSA (Rivest, Shamir, Adleman), IDEA (International Data Encryption Algorithm) or El-Gamal, and secret-key algorithms, such as DES (Data Encryption Standard), TDES (Triple DES) and AES (Advanced Encryption Standard) [1]. Some of them works binary (DES, TDES), RSA uses decimal numbers; others are defined on finite algebraic fields. The most powerful is considered to be AES, which uses 128, 192 and 256-bit encryption keys. The known attacks against AES are not yet successful but it is a matter of time till breaking AES will be possible. Computing technology evolves quickly and all known encryption algorithms are intensively studied by the cryptanalysts in order to develop efficiently attack methods. So, it is necessary to make AES more robust and one way to do this is to increase the encryption key length to 384, 512, 768 and 1024 bits. This involves larger data structures and also larger algebraic fields. We propose and present the modified AES algorithm, defined on GF (256), on data matrices of 8*6, 8*8, 8*12 and 8*16 bytes.**

***Index Terms* — Data security, Data structures, Cryptography**

I. INTRODUCTION

Confidentiality is an important security service in any communication system.

The private character of the information transmitted on a communication channel is ensured using different cryptographic techniques.

A powerful cryptography system should resist to all types of attacks:

- **Differential cryptanalysis,** based on the correlation properties of the coded sequence (discovered in 1990).
- **Linear cryptanalysis,** searching for some correlation between the coded signal and the original.
- **Key cryptanalysis,** looking for the encryption key from a large set of possible key-sequences.

Public key cryptography and secret-key cryptography compete for the most robust algorithms, but it seems that the algorithms based on secret encryption keys are the most powerful.

Brute-force attack is trying to find the encryption key analyzing a set of possible values. Its success depends on the key space dimension and it is inefficient for very long key sequences. Nowadays, 256-bit keys are long enough to avoid this type of attack.

The number of iterations made by an encryption algorithm is also important to ensure its robustness.

DES (Data Encryption Standard) is a binary symmetric block-code, with a 64-bit secret encryption key and it has 16 identical rounds (ANSI X3.92).

TDES (Triple Data Encryption Standard) applies three times the DES encoding and decoding algorithms, with a larger key, of 128 bits and it is more robust than its predecessor.

AES (Advanced Encryption Standard) operates on octets, with secret encryption keys of 128, 192 up to 256 bits, and it makes 10, 12 or 14 iterations, depending on the length of the encryption keys. But successful attacks were developed only for a reduced number of iterations. Therefore it is widely used on computer networks, to protect stored information and transmitted data. It is adopted for different governmental and military applications but it is successfully used in other domains.

RSA (Rivest, Shamir, Adleman) operates on the decimal value of the message and its robustness is based on the impossibility to factorize very large numbers (of at least 300 digits). It can use public encryption keys up to 2048 bits to prevent the brute force attack.

Many encryption algorithms can be defined on Galois Fields (GF). The binary alphabet is, in fact, a GF with only two elements.

A GF is an algebraic finite field with a number of elements equal to a power of 2. It is denoted by GF (2^m) [2].

The Galois Field is very convenient for encoding algorithms, because computation is simple and there are dedicated circuits and fast algorithms for GF algebra. There is no difference in complexity comparing to binary algebra.

An element a from the Galois field GF (2^m), can be expressed:

- in the decimal system,
- as a binary sequence of m bits:
$$a = [a_0\ a_1\ a_2\ \ldots\ a_{m-1}] \quad (1)$$
- as the polynomial associated to the binary sequence:
$$a(x) = \sum_{i=0}^{m-1} a_i x^i \quad (2)$$
- exponentially, using a primitive element b of the field (if the element itself is not null):
$$a = b^e,\ a \neq 0. \quad (3)$$

For example, GF (256) works with bytes, having integer values from 0 to 255:

GF (256) = {0, 1, 2 … 255} (4)

Each element can be written as an 8-bit sequence, as a polynomial, having the coefficients given by the corresponding binary sequence and also, with an exponent of 3 (if it is not null), which is a primitive element of this field.

For example, the number 23 is expressed as the octet [0 0

0 1 0 1 1 1] and it has the associated polynomial x^4+x^2+x+1. It could be written in exponential form, as 3^{239} in GF (256).

Addition and multiplication are inner operations of the algebraic finite field.

On GF (256), addition of two numbers, *a* and *b*, is made modulo-2, bit-by-bit, using the binary expression of the elements:

$$s = a + b \Leftrightarrow s_i = a_i \oplus b_i, \quad for\ i = 0\ \ to\ \ 7 \tag{5}$$

Subtraction is similar to addition. It implies the opposite element of the subtractive, but using modulo-2 operations, the opposite element is the element itself, because $b + b = 0$:

$$c = a - b = a + (-b) = a + b \tag{6}$$

On GF (256), multiplication of two elements is made multiplying their polynomials and finally, a primitive polynomial p(x), is used to reduce modulo-p(x) the result.

The multiplication of two elements is made as it follows:

$$c(x) = \{a(x) \cdot b(x)\} \bmod p(x) \tag{7}$$

where $a(x) = \sum_{i=0}^{m-1} a_i x^i$, $b(x) = \sum_{i=0}^{m-1} b_i x^i$, $c(x) = \sum_{i=0}^{m-1} c_i x^i$ are the polynomials of the elements *a*, *b* and *c* of the GF.

For example, on GF (256), we may use the primitive polynomial:

$$p(x) = x^8 + x^4 + x^3 + x^2 + 1 \tag{8}$$

It is called the polynomial number 285, according to the decimal value which corresponds to its set of binary coefficients: [1 0 0 0 1 1 1 0 1].

Other primitive polynomials may also be used, but it must be specified each time some arithmetic operations on GF are made.

Division is defined based on the inverse element which exists for all not-null elements of the field. In fact, division of an element *a* by *b* is computed as the multiplication of *a* with the inverse element b^{-1}:

$$c = a / b = a \cdot b^{-1}, \text{ with } b \neq 0 \text{ and } b \cdot b^{-1} = 1. \tag{9}$$

All algebraic operations are used by the encoding and decoding algorithms defined on Galois Fields, including exponentiation, logarithm computation, matrix operations and so on.

II. CLASSIC AES ALGORITHM

The classic AES algorithm is a block code, defined on 128, 192 or 256-bit words, according to the length of the encryption key and denoted as AES-128, AES-192, and AES-256.

AES operates on the Galois Field with 256 elements, denoted GF (256).

Each data word is written as a sequence of 16, 24 or 32 bytes, which become the elements of the data-matrix, defined on the 8-bit Galois Field GF (256). So, AES is applied on a 4*4, 4*6 or 4*8 data matrix.

The serial input data stream is partitioned into blocks of 8 bits, which represent elements of the finite algebraic Galois Field GF (256). These elements are grouped into state-matrices, with 4 rows and Nc columns (Nc = 4, 6 or 8), depending on the chosen length of the encryption key (128, 192 or 256 bits) (Figure 1).

First, each i^{th} block of 8 bits [$b_i\ b_{i+1}\ b_{i+2}\ b_{i+3}\ b_{i+4}\ b_{i+5}\ b_{i+6}\ b_{i+7}$] is non-linearly processed bit-by-bit, as modulo-2 sums of transmitted bits and some constant values c_i read from the constant byte 0110.0011 = 0x.63:

$$b'_i = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i \tag{10}$$

This step is a substitution process and therefore it is called the SubBytes stage.

S_{24} =

s_{00}	s_{01}	s_{02}	s_{03}	s_{04}	s_{05}
s_{10}	s_{11}	s_{12}	s_{13}	s_{14}	s_{15}
s_{20}	s_{21}	s_{22}	s_{23}	s_{24}	s_{25}
s_{30}	s_{31}	s_{32}	s_{33}	s_{34}	s_{35}

Fig. 1 State-matrix for AES-192

Next step permutes the elements on each row of the state-matrix, with a different shift value: 0 for the first row, 1 for the second one, 2 for the third and 3 for the last.

For AES-128, the permutation is made as it follows:

$$S_{16}' = \begin{bmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ s_{13} & s_{10} & s_{11} & s_{12} \\ s_{22} & s_{23} & s_{20} & s_{21} \\ s_{31} & s_{32} & s_{33} & s_{30} \end{bmatrix} \tag{11}$$

This permutation corresponds to the cryptographic technique of transposition, in the simple manner of right-hand shifting.

Then, AES processes each column of the matrix, using an invertible polynomial *a(x)*, for column multiplication:

$$a(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 \tag{12}$$

Rjindael propose a particular reducible polynomial a(x) for column multiplication [3, 1, 1, 2], but other polynomials of third degree may also be used.

Each column of the state matrix is associated to a polynomial *s(x)* which is multiplied by the encoding polynomial *a(x)*.

The elements of the state matrix are octets, so all the arithmetic operations are made on GF (256).

The polynomial multiplication is made modulo-p(x), where p(x) is the polynomial number 17:

$$p(x) = x^4 + 1 \tag{13}$$

The coefficients of the resulting polynomial are written back into the state matrix.

For AES-128, instead of multiplying the polynomial of each column with *p(x)*, we can compute the multiplication between the invertible matrix *A,* associated to the polynomial *a(x),* and each column vector of the state-matrix S:

$$A = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ a_3 & a_0 & a_1 & a_2 \\ a_2 & a_3 & a_0 & a_1 \\ a_1 & a_2 & a_3 & a_0 \end{bmatrix} \tag{14}$$

$$\overline{c_i} = A \cdot \overline{s_i},\ i = \overline{0, N_c - 1} \tag{15}$$

and $\overline{s_i} = \begin{bmatrix} s_{i,0} \\ s_{i,1} \\ s_{i,2} \\ s_{i,3} \end{bmatrix}$ represents the i[th] column of the state matrix.

In the particular case of AES-128, this step corresponds to the matricial multiplication between *A* and *S:*

$$C = A \otimes S \tag{16}$$

All arithmetic operations are made on GF (256).

We prefer using the matricial way instead of the polynomial mode, in order to easily implement the algorithm.

The fourth step of the algorithm applies the encryption key to the data matrix.

The encryption key is written as a key-matrix *K*, having the same dimensions as the state-matrix (*S*), and then the sum of the matrices is computed on GF (256), according to the following relation:

$$E = K \oplus C \tag{17}$$

Decryption involves the same steps, in reverted order, and uses the inverse matrix A^{-1}:

$$C = E \oplus K;\ D = A^{-1} \otimes C$$

$$\text{or } \overline{d_i} = A^{-1} \cdot \overline{c_i},\ i = \overline{0, N_c - 1} \tag{18}$$

After all these operations are made, the inverse permutation of the elements on each row is applied and finally the original bits of each byte are deduced.

III. MODIFIED AES ALGORITHM

To increase the robustness of the AES algorithm, we have to use longer encryption keys and larger data matrix [3]. To keep the processing time at low values, we have to maintain unchanged the complexity of the AES algorithm.

The modified AES algorithm (MAES) will work on data matrices with exactly 8 rows and a variable number of columns (N_c): 6, 8, 12 and 16 (Figure 2). Same steps are made. The encryption key will have an equivalent length of about 384, 512, 768 and 1024 bits, and the modified algorithm is denoted according to it: MAES-384, MAES-512, MAES-768 and MAES-1024.

$$S_{64} = \begin{bmatrix} s_{00} & s_{10} & s_{20} & s_{30} & s_{40} & s_{50} & s_{60} & s_{70} \\ s_{01} & s_{11} & s_{21} & s_{31} & s_{41} & s_{51} & s_{61} & s_{71} \\ s_{02} & s_{12} & s_{22} & s_{32} & s_{42} & s_{52} & s_{62} & s_{72} \\ s_{03} & s_{13} & s_{23} & s_{33} & s_{43} & s_{53} & s_{63} & s_{73} \\ s_{04} & s_{14} & s_{24} & s_{34} & s_{44} & s_{54} & s_{64} & s_{74} \\ s_{05} & s_{15} & s_{25} & s_{35} & s_{45} & s_{55} & s_{65} & s_{75} \\ s_{06} & s_{16} & s_{26} & s_{36} & s_{46} & s_{56} & s_{66} & s_{76} \\ s_{07} & s_{17} & s_{27} & s_{37} & s_{47} & s_{57} & s_{67} & s_{77} \end{bmatrix}$$

Fig. 2 State-matrix for MAES-512

Input data will be processed as data blocks of 48, 64, 96 or 128 bytes, associated with the input polynomial:

$$s(x) = \sum_{i=0}^{7} \sum_{j=0}^{N_c - 1} s_{ij} x^{ij} \tag{19}$$

The first and the second steps of AES are similarly done by MAES, excepting the shift value which is increased, row-by-row, from 0 to 7.

For the third step of MAES, we propose the following invertible polynomial:

$$a(x) = x^7 + 2x^6 + 3x^5 + 4x^4 + 5x^3 + 6x^2 + 7x + 8 \tag{20}$$

The encryption polynomial has a higher order in comparison to the classic AES algorithm, considering the increased number of rows, from 4 to 8:

$$a(x) = \sum_{i=0}^{7} a_i x^i \tag{21}$$

For column multiplication step, it may be used the encryption matrix *A,* given below:

$$A = \begin{bmatrix} 13 & 7 & 5 & 4 & 4 & 3 & 2 & 1 \\ 2 & 13 & 7 & 5 & 4 & 4 & 3 & 2 \\ 7 & 2 & 13 & 7 & 5 & 4 & 4 & 3 \\ 5 & 7 & 2 & 13 & 7 & 5 & 4 & 4 \\ 6 & 2 & 2 & 6 & 9 & 4 & 7 & 5 \\ 3 & 1 & 7 & 6 & 2 & 10 & 6 & 6 \\ 5 & 4 & 4 & 3 & 2 & 1 & 8 & 7 \\ 10 & 6 & 7 & 4 & 3 & 2 & 1 & 8 \end{bmatrix} \tag{22}$$

It is an 8*8 matrix, deduced based on the invertible polynomial *a(x)*, given by the equation (20).

Its determinant, computed on GF (256) using some Matlab functions [4], is equal to 242, so it is an invertible matrix.

It is more efficient to use a secret encryption polynomial, with a set of coefficients deduced based on a password of the user or on the encryption key. If the resulting polynomial is not-invertible, a unit will be added to it and the problem is solved. Even if this procedure is more secure, it needs more time to compute the inverse matrix for the decoder.

The multiplication of the two polynomials on GF (256) is made modulo-*p(x)*, where *p(x)* can be x^8+1 or another 8-degree primitive polynomial, obtained based on the primitive element of GF (256) equal to 3 and its minimal polynomial:

$$p(x) = M_3(x) = x^8 + x^4 + x^3 + x^2 + 1 \tag{23}$$

We recommend using the minimal polynomial for multiplication.

The encoded polynomial is computed on GF (256), as:

$$c(x) = \{a(x) \cdot s(x)\} \bmod - p(x) = \sum_{i=0}^{7} c_i x^i \tag{24}$$

Its coefficients are derived and equation (15) is written using the invertible encryption matrix *A* of 8*8 elements and two column-vectors of 8 coefficients each, one is a column of the state-matrix *S* and the other is the output encoded column-vector:

$$\overline{c} = A \cdot \overline{s} \tag{25}$$

Relation (17) is finally applied for MAES, as we use it in AES.

Same steps are used by MAES in comparison to AES, but the dimensions of all the arrays are increased.

The decryption algorithm makes the inverse operation:

$$\bar{s} = A^{-1} \cdot \bar{c} \tag{26}$$

A Matlab algorithm for matrix inversion, on GF (256), is applied to compute the decoding matrix A^{-1}:

$$A^{-1} = \begin{bmatrix} 4 & 9 & 177 & 145 & 195 & 84 & 19 & 151 \\ 107 & 44 & 216 & 46 & 123 & 60 & 62 & 78 \\ 47 & 195 & 27 & 54 & 206 & 183 & 173 & 104 \\ 182 & 202 & 228 & 72 & 96 & 10 & 121 & 86 \\ 129 & 95 & 137 & 7 & 11 & 19 & 35 & 174 \\ 51 & 169 & 162 & 180 & 34 & 15 & 76 & 68 \\ 143 & 183 & 49 & 204 & 206 & 50 & 237 & 51 \\ 32 & 180 & 197 & 64 & 87 & 34 & 52 & 108 \end{bmatrix} \tag{27}$$

The modified algorithm makes for each input data block an increased number of arithmetical operations in comparison to AES, but the number of operations-per-symbol (ops) remains the same since larger data structures are processed by MAES.

IV. CONCLUSION

The modified AES algorithm, called MAES, is more complex and more robust than the AES algorithm and the encoding time is mostly the same, because no more data are processed and the arithmetic operations are all made in the same Galois field. Only the dimension of the data block is changed but fewer blocks are processed for the same volume of the transmitted data.

The existence of the inverse matrix on GF (256) should be tested each time if different encoding polynomials are used, but special cases could be solved in a simple manner, to obtain a non-zero determinant of the encoding matrix.

The proposed MAES algorithm could be implemented in different programming languages, such as Java [5], in order to be used on different communication devices, like computers, mobile phones or i-pads, with different operating systems and requirements.

Data confidentiality is essential on wireless communication systems, and more robust encryption algorithms are needed. The proposed MAES algorithm meets all these goals, its complexity is comparable to the classic AES algorithm but it is more robust in comparison to it, using longer encryption keys and higher matrix dimensions.

Some improvements of MAES, for larger Galois fields, will be considered for future works.

REFERENCES

[1] B. Schneier, “Applied cryptography”, second edition, NY: John Wiley & Sons, Inc., 1996

[2] J.G. Proakis, D.G. Manolakis, "Introduction to Digital Signal Processing", MacMillan Publishing Company, 1988

[3] L. Scripcariu, S. Ciornei, “Improving the Encryption Algorithms Using Multidimensional Data Structures”, Proceedings of the Third European Conference on the Use of Modern Information and Communication Technologies, ECUMICT 2008, Gent (Belgium), pp. 375 – 384, Mar. 2008.

[4] B.D. Hahn, D.T. Valentine, “Essential MATLAB for Engineers and Scientists, 4e”, Academic Press, 2010

[5] L. Scripcariu, A. Alistar, M.D. Frunza, “JAVA Implemented Encryption Algorithm”, Proceedings of the 8th Int. Conference "Development and Application Systems", Suceava, DAS 2006, pp. 424-429, May 2006.