VARIATIONS TO THE CRYPTOGRAPHICS ALGORITHMS AES AND TWOFISH

P. Freyre*[1], N. Díaz * and O. Cuellar †[2]

*Faculty of Mathematics and Computer Science, University of Havana, Cuba.

†Faculty of Mathematics, Physical and Computer Science, Central University of Las Villas, Cuba.

[1]e-mail: pfreyre@matcom.uh.cu

[2]e-mail: oristela@uclv.edu.cu

Abstract

The Cryptographics Algorithms AES and Twofish guarantee a high diffusion with the use of fixed MDS matrices of size 4×4. In this article variations to the Cryptographics Algorithms AES and Twofish are made. They allow that the process of cipher - decipher come true with MDS matrices selected randomly from an algorithm that obtaining an MDS matrix of set of all the MDS matrices possible. A new Schedule of key with a high diffusion is designed for the Algorithm Cryptographic AES. Besides it is proposed a new S - box that he varies in function of the key.

Key words: Block Cipher, AES, Twofish, S - boxes, MDS matrix.

1 Introduction.

The Cryptographics Algorithms of Block Cipher Rijndael [DR99] and [DR02] and Twofish [SKWHF98] and [GBS13] were finalists in the world contest to select the Advanced Encryption Standard (AES) convened by the National Institute of Standards and Technology from the United States (NIST). The contest finished in October 2000 with the selection of the Cryptographics Algorithms Rijndael as the AES, which was proposed by Joan Daemen and Vincent Rijmen, from Belgium.

In order to reach a high diffusion, the Cryptographics Algorithms AES and Twofish, use a MDS (Maximal Distance Separable) matrix, selected a priori. In this paper we will explain the variations of these Cryptographics Algorithms of Block Cipher, where the MDS matrix is selected randomly in function of the secret key, as part of the Schedule of Keys. In addition, a new Schedule of Keys for the Cryptographic Algorithm AES is proposed, which guarantee a high diffusion and where a new S-box as function key is obtained. Proposals of variation of the Cryptographic Algorithm AES can be found in [AE13], [AHK13], [ERDM09], [IGKAE12], [MEEZ13], [MJ11] and [MKAF11].

The algorithm for the random generation of MDS matrix $A = \{a_{i,j}\}_{4\times 4}$, where for all i and j, $a_{i,j} \in GF(2^8)$ (GF - Galois Fields), it only needs a random matrix $M = \{m_{i,j}\}_{4\times 4}$, where for all i and j, $a_{i,j} \in GF(2^8)$, which has as restriction that for none i, $i \geq 2$, that is fulfilled $m_{i,i} = m_{i,i+1} = \cdots m_{i,4} = 0$ and $m_{1,2}, m_{1,3}$ and $m_{1,4} \neq 0$. This algorithm has the advantage, in relation to the other ones, that the selection of a random MDS matrix is achieved from the set of all MDS matrices, see epigraph 2.3 and [FDDP14]. The attainment of MDS matrices can be seen in [AF14], [AF13],[DMMF15], [DMMMP14], [GR13], [GR13a], [GR14], [KM14], [JV04], [LF04], [MI08], [MI11], [NA09], [RRYB15], [SDMO12] and [SKOP15].

This paper also contains a proposal of variation, in function of the key, of the S - box (S_{RD}) from the Cryptographic Algorithm AES. S_{RD} is transformed into in $S'_{RD} = \lambda_2 S_{RD} \lambda_1(x)$, where λ_1 y λ_2 are Boolean invertible matrices of 8×8, which are generated in the Schedule of Key for the algorithm that is described in epigraph 2.3 of the present paper. Proposals of variation of the S-box from AES can be found in [Ke97], [FSESH05], [KM08], [MRE09], [KK09] and [JMV15].

This article begins with the summarized description of the Cryptographic Algorithm AES, followed by variation proposals and afterwards a similarly summarized description of the Cryptographic Algorithm Twofish, followed by the explanation of its variations. The work finishes with two algorithms which allow the variation in function of the key of the S - box from AES and the MDS matrices of the Cryptographics Algorithms AES and Twofish.

2 Development.

2.1 Cryptographic Algorithm of Block Cipher AES.

The input and output blocks of the Cryptographic Algorithm AES are described in matrices forms of bytes in 4 rows per $N_b = 4$ columns. The input matrix is formed from the succession of bytes of clear text $p_0p_1p_2p_3 \ldots p_{4N_b-1}$ in the following way: $a_{i,j} = p_{i+4j}\ 0 \leq i < 4,\ 0 \leq j < N_b$, where p_0 is the initial byte and p_{4N_b-1} is the final byte. The output matrix is transformed into bytes of ciphered texts $c_0c_1c_2c_3 \ldots c_{4N_b-1}$ in the following way: $c_{i+4j} = a_{i,j}\ 0 \leq i < 4,\ 0 \leq j < N_b$. The transformations in each round operate on the matrix $a_{i,j}$, $0 \leq i < 4$, $0 \leq j < N_b$, which it is denoted matrix of state [DR99] and [DR02].

The key is a one-dimensional arrangement of bytes which is written as a matrix of bytes of 4 rows per $N_k = 4$, 6 o 8 columns. The number of round N_r in the Cryptographic Algorithm AES is function of N_b and N_k [DR99] and [DR02].

The Cryptographic Algorithm AES uses the following funtions: **SubBytes**, which we denote as S_{RD}, **ShiftRows** and **MixColumns**. The schedule of key for $N_k \leq 6$ and $N_k > 6$ is presented in pseudo code in [DR99] and [DR02].

2.1.1 Variations of the Cryptographic Algorithm of Block Cipher AES.

Variation of Schedule of Keys.

The Cryptographic Algorithm AES has a Schedule of Keys with a low diffusion, which it has made possible the success of differential Cryptanalysis with related keys (see [JD04], [BKN09] and [BK09]), in order to avoid this cryptanalytic attack in [DR12] and [CZKHP11] it is proposed to redesign a new Schedule of Keys.

The first variation to the Cryptographic Algorithm AES exhibited in this paper is the substitution of its Schedule of Keys by the one that follows, which uses in its base Rijndael Cryptographic Algorithm with $N_b = N_k = 8$ and $N_r = 10$ when the key is 256 bits, $N_b = N_k = 6$ and $N_r = 8$ when the key is 192 bits and $N_b = N_k = 4$ and $N_r = 8$ when the key is 128 bits [DR99] and [DR02].

To generate the keys in each round with this new Schedule of Keys, the following algorithms are used:

MDSMatrixGeneration:
Input:
1. Primitive polynomials $g_1(x),\ g_2(x)$ y $g_3(x) \in GF(2^8)[x]$ (they are a priori selected and $gr(g_1(x)) = 4,\ gr(g_2(x)) = 3$ and $gr(g_3(x)) = 2$)
2. 16 bytes that are transformed in a $M[4][4]$ matrix of bytes.

Output:
1. MDS matrix $A[4][4]$ of bytes used in the **MixColumns** function.

This algorithm of generation of random MDS matrix can be seen in epigraph 2.3 and [FDDP14]. If in the matrix $M[4][4]$ comes true that $(m_{k,0}, m_{k,1}, \ldots, m_{k,4-k}) = 0$ then $(m_{k,0}, m_{k,1}, \ldots, m_{k,4-k}) = (0, 0, \ldots, 0, 2^8 - 1)$, $k = 2..4$.

BooleanMatrixGeneration:
Input:
1. Primitive polynomials $g_1(x),\ g_2(x), \ldots,\ g_7(x) \in GF(2)[x]$(They are a priori selected and $gr(g_1(x)) = 8,\ gr(g_2(x)) = 7, \ldots,\ gr(g_7(x)) = 2$).
2. 8 bytes that are transformed in a Boolean matrix $M[8][8]$.

Output:
1. Invertible Boolean matrix $A[8][8]$.

This algorithm of generation of random invertible Boolean matrices can be seen in epigraph 2.3 and [FDM09]. If in the matrix $M[8][8]$ comes true that $(m_{k,0}, m_{k,1}, \ldots, m_{k,8-k}) = 0$ then $(m_{k,0}, m_{k,1}, \ldots, m_{k,8-k}) = (0, 0, \ldots, 0, 1),\ k = 1..8$.

The **BooleanMatrixGeneration** algorithm is used to create the new **SubBytes** function where S_{RD} is substituted by $S'_{RD} = \lambda_2 S_{RD} \lambda_1$ and the matrices λ_2 and λ_1 are obtained by the **BooleanMatrixGeneration** algorithm. Anothers proposal of variation of the S - box from AES can be found in [MRE09], [KK09] and [JMV15].

RoundKeyGeneration:
Input:
1. The key of 256, 192 or 128 bits identified as Key.
2. A primitive element $\alpha \in GF(2^8)$.
3. The S - Box of the AES identified as S_{RD}.

Output:
1. 17 keys of 16 bytes identified as RoundKey[17][16].

Note: RoundKey[0][16], RoundKey[1][16],..., RoundKey[14][16] are the keys of rounds, RoundKey[15][16] is used in the **MDSMatrixGeneration** algorithm, RoundKey[16][16] is used to obtain two Boolean matrices with the **BooleanMatrixGeneration** algorithm.

RoundKeyGeneration(byte Key[$Nkey$],byte Keyr[Nr][$Nkey$], byte α, byte S_{RD}[256],byte RoundKey[17][16]){
switch (Nkey){
case 32:
for(t=0;t $<$ 16;t++){
RoundKey[0][t]=Key[t];
RoundKey[1][t]=Key[16+t];
}
MDSMatrixGeneration(S_{RD}[Key[0]],S_{RD}[Key[1]],. . . ,S_{RD}[Key[15]]);
λ_1=BooleanMatrixGeneration(S_{RD}[Key[16]],S_{RD}[Key[17]],. . . ,S_{RD}[Key[23]]);
λ_2=BooleanMatrixGeneration(S_{RD}[Key[24]],S_{RD}[Key[25]],. . . ,S_{RD}[Key[31]]);
for(i=0;i $<$ 256;i++)
S'_{RD}[i] $= \lambda_2 S_{RD} \lambda_1$[i];
for(i=0;i $<$ Nr;i++)
for(j=0;j $<$ Nkey;j++)
Keyr[i][j]= $S'_{RD}\left[\alpha^{(i \, * \, Nkey \, + \, j) \; mod \; 255}\right]$;
for(j=0;j $<$ 8;j++){
for(i=0;i $<$ 10;i++){
SubBytes(Key);
ShiftRows(Key);
MixColumns(Key);
AddRoundKey(Key,Keyr[i]);
}
If(j $\neq$ 7)
for(t = 0 ; t ¡ 16 ; t++) {
RoundKey[j*2+2][t]=Key[t];
RoundKey[j*2+3][t]=Key[16+t];
}
else
for(t=0;t $<$ 16;t++)
RoundKey[j*2+2][t]=Key[t];
Keyr[j] $<<$ 8;
}
break
case 24:
for(t=0;t $<$ 8;t++){
RoundKey[0][t]=Key[t];
RoundKey[0][t+8]=Key[t+8];
RoundKey[1][t]=Key[t+16];
}
MDSMatrixGeneration(S_{RD}[Key[0]],S_{RD}[Key[1]],. . . ,S_{RD}[Key[15]]);
for(i=0;i $<$ Nr;i++)
for(j=0;j $<$ Nkey;j++)
Keyr[i][j]=$\alpha^{(i \, * \, NKey \, + \, j)}$;

```
for(i=0;i < 8;i++){
   SubBytes(Key);
   ShiftRows(Key);
   MixColumns(Key);
   AddRoundKey(Key,Keyr[i]);
}
λ1=BooleanMatrixGeneration(S_RD[Key[0]],S_RD[Key[1]],...,S_RD[Key[7]]);
λ2=BooleanMatrixGeneration(S_RD[Key[8]],S_RD[Key[9]],...,S_RD[Key[15]]);
for(i=0;i < 8;i++)
   RoundKey[1][t+8]=Key[t+16];
for(i=0;i < 256;i++)
   S'_RD[i]= λ2 S_RD λ1[i];
for(i=0;i < Nr;i++)
   for(j=0;j < Nkey;j++)
      Keyr[i][j]= S'_RD [α^(i * Nkey + j)] ;
for(j=1;j < 11;j++){
   for(i=0;i < 8;i++){
      SubBytes(Key);
      ShiftRows(Key);
      MixColumns(Key);
      AddRoundKey(Key,Keyr[i]);
   }
if(j mod 2)
   for(t=0;t < 8;t++){
      RoundKey[j+1][t]=Key[t];
      RoundKey[j+1][t+8]=Key[t+8];
      RoundKey[j+2][t]=Key[t+16];
   }
else
   for(t=0;t < 8;t++){
      RoundKey[j+2][t]=Key[t];
      RoundKey[j+2][t+8]=Key[t+8];
      RoundKey[j+1][t]=Key[t+16];
   }
Keyr[(j  1) mod 8] <<  8;
}
break
Case 16:
for(t=0;t < 16 ; t++)      RoundKey[0][t]=Key[t];
MDSMatrixGeneration(SRD[Key[0]],S_RD[Key[1]],...,SRD[Key[15]]);
for(i=0;i < Nr;i++)
   for(j=0;j < Nkey;j++)
      Keyr[i][j]=α^(i * NKey + j);
for(i=0;i < 8;i++){
```

```
    SubBytes(Key);
    ShiftRows(Key);
    MixColumns(Key);
    AddRoundKey(Key,Keyr[i]);
  }
  λ1=BooleanMatrixGeneration(S_RD[Key[0]],S_RD[Key[1]],...,S_RD[Key[7]]);
  λ2=BooleanMatrixGeneration(S_RD[Key[8]],S_RD[Key[9]],...,S_RD[Key[15]]);
  for(i=0;i < 256;i++)
    S'_RD[i]=λ2 S_RD λ1[i];
  for(i=0;i < Nr;i++)
    for(j=0;j < Nkey;j++)
      Keyr[i][j]= S'_RD [α^(i * NKey + j)];
  for(j=0;j < 16;j++){
    for(i=0;i < 8;i++){
      SubBytes(Key);
      ShiftRows(Key);
      MixColumns(Key);
      AddRoundKey(Key,Keyr[i]);
    }
  for(t=0;t < 16;t++)
    RoundKey[j+1][t]=Key[t];
  Keyr[j mod 8]  << 8;
  }
}
MDSMatrixGeneration(RoundKey[15][0],RoundKey[15][1],...,RoundKey[15][15]);
λ1=BooleanMatrixGeneration(RoundKey[16][0],RoundKey[16][1],...,RoundKey[16][7]);
λ2=BooleanMatrixGeneration(RoundKey[16][8],RoundKey[16][9],...,RoundKey[16][15]);
for(i=0;i < 256;i++)
S'_RD[i] = λ2 S_RD λ1[i];
```

Variation of Rijndael Algorithm in Text Ciphering.

Ciphering a clear text block is done in the following way:

```
Rijndael(State ,CipherKey???){
  switch(???){
    case 256:
      Nr = 10;
      Nkey = 32;
    Break;
    case 192:
      Nr = 8;
      Nkey = 24;
    Break;
```

```
    case 128:
      Nr = 8;
      Nkey = 16;
  }
  RoundKeyGeneration(Key[Nkey],Keyr[Nr][Nkey],α,SRD[256],RoundKey[17][16]){
  AddRoundKey (State ,RoundKey[0]);
  for(i=1;i < 14;i++)
    Round(State,RoundKey[i]);
  FinalRound(State,RoundKey[14]);
  }
  Round(State,RoundKey[i]){
    SubBytes(State);
    ShiftRows(State);
    MixColumns(State);
    AddRoundKey(State,RoundKey[i]);
  }
  FinalRound(State,RoundKey[14]){
    SubBytes(State);
    ShiftRows(State);
    AddRoundKey(State,ExpandedKey[14]);
  }
}
```

2.2 Cryptographic Algorithm of Block Cipher Twofish.

The Cryptographic Algorithm of Block Cipher Twofish [SKWHF98] and [GBS13] has 128 bits of input - output block and it accepts keys of variable longitude up to 256 bits, it is a Feistel Cipher with an F function conformed by S - boxes of 8 bits and a MDS fixed matriz $A = \{a_{i,j}\}_{4\times4}$, where for all i and j, $a_{i,j} \in GF(2^8)$, it has a Schedule of keys that uses the same primitives than the ones from the Algorithm body and it was carefully designed to resist the differential cryptoanalysis with related key.

In the Cryptographic Algorithm Twofish when the longitude of the key is $N = 128$, 192 or 256 bits, then the key divided into bytes is $m_0, \ldots, m_{8K-1}$, where $K = N/64$. the bytes are converted into $2K$ words of 32 bits through the following expression:

$$M_i = \sum_{j=0}^{3} m_{4i+j} \; 2^{8j}$$

Where $M_e = (M_0, M_2, \ldots, M_{2k-2})$ and $M_0 = (M_1, M_3, \ldots, M_{2k-1})$.

The Schedule of keys extends the key in 40 words of 32 bits $K_0,\ K_1, \ldots,\ K_{39}$ in the following way:

$\rho = 2^{24}\ +\ 2^{16}\ +\ 2^{8}\ +\ 2^{0}$
$A_i = h(2i\rho, M_e)$
$B_i = ROL(h((2i+1)\rho, M_0), 8)$
$K_{2i} = (A_i\ +\ B_i)\ mod\ 2^{32}$
$K_{2i+1} = ROL((A_i\ +\ 2B_i)\ mod\ 2^{32}, 9)$
Where: $i = 0, \ldots, 19$

2.2.1 Variation of Cryptographic Algorithm of Block Cipher Twofish.

The creators of the Cryptographic Algorithm Twofish designed a cipher function where the S-Boxes depend on the key. They thought in the possibility of a MDS matrix to be formed during the Schedule of keys similarly as function of the key, but finally they discarded this variant due to the amount of additional work that they had to incorporate to the Schedule of keys [SKWHF98] and [GBS13].

The modification proposed in this paper to the Cryptographic Algorithm of Block Cipher Twofish is precisely to make a variable of the MDS matrix and that it to be generated in the Schedule of key as a function of the key, to do that, we transform the Schedule of key in the following way:

The Schedule of keys expands the key in 44 words of 32 bits $K_0,\ K_1, \ldots,$ K_{43} in a similar way as it is previously done:

$\rho = 2^{24}\ +\ 2^{16}\ +\ 2^{8}\ +\ 2^{0}$
$A_i = h(2i\rho, M_e)$
$B_i = ROL(h((2i+1)\rho, M_0), 8)$
$K_{2i} = (A_i\ +\ B_i)\ mod\ 2^{32}$
$K_{2i+1} = ROL((A_i\ +\ 2B_i)\ mod\ 2^{32}, 9)$

Where: $i = 0, \ldots, 21$

The words of 32 bits $K_0,\ K_1, \ldots, K_{39}$ are used in the Cryptographic Algorithm as it has been established and the 4 words of 32 bits $K_{40},\ K_{41},$ $K_{42},\ K_{43}$ are used to conform a MDS matrix through the algorithm that is described in epigraph 2.3 and [FDDP14].

First of all, another variant to have into consideration is that the MDS matrix used in the Schedule of Key be obtained first, taking the 128 bits of $M_e = (M_0,\, M_2, \ldots, M_{2k-2})$ and $M_0 = (M_1,\, M_3, \ldots, M_{2k-1})$ but choosing the values from right to left as it's shown $M_{2k-2},\ M_{2k-1},\ \ldots,\ M_2,\ M_3,\ M_0$, M_1, and later with this MDS matrix and the Schedule of key to obtain the 4 words of 32 bits K_{40},K_{41}, K_{42}, K_{43} to conform the MDS matrix that will be used in the cipher and decipher of texts.

2.3 Algorithms for the random generation of matrices.

2.3.1 Algorithm for the random generation of a Boolean invertible matrix $\mathbf{A} = \{\mathbf{a_{i,j}}\}_{\mathbf{8\times 8}}$, where for all i and j, $\mathbf{a_{i,j}} \in \mathbf{GF(2)}$

The algorithm for the generation of a Boolean invertible matrix $A = \{a_{i,j}\}_{8\times 8}$, where for all i and j, $a_{i,j} \in GF(2)$ is presented here, it will be used in the generation of Boolean invertible matrices λ_1 and λ_2 to transform the S–box of the Cryptographic Algorithm Rijndael. The explanation and analysis of this algorithm of generation of random invertible matrices for the general case in which the elements of the matrix belong to an finite arbitrary field can be seen in [FDM09].

BooleanMatrixGeneration
Input:
- Primitive polynomials $g_1(x),\, g_2(x), \ldots, g_7(x) \in GF(2)[x]$(They are selected a priori and $gr(g_1(x)) = 8,\ g_r(g_2(x)) = 7, \ldots, gr(g_7(x)) = 2)$.
- Random matrix M.

$$M = \begin{pmatrix} b_{1,0} & b_{1,1} & b_{1,2} & \ldots & b_{1,7} \\ c_{2,0} & b_{2,0} & b_{2,1} & \ldots & b_{2,6} \\ c_{3,0} & c_{3,1} & b_{3,0} & \ldots & b_{3,5} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ c_{7,0} & \ldots & c_{7,5} & b_{7,0} & b_{7,1} \\ c_{8,0} & c_{8,1} & \ldots & c_{8,6} & b_{8,0} \end{pmatrix}$$

Where: $c_{i,j}$ and $b_{k,t} \in GF(2)$, $(b_{k,0},\, b_{k,1}, \ldots, b_{k,8-k}) \neq 0, k = 1..8,\ t = 0..7$, $i = 2..8$ and $j = 0..6$.

Begin
Calculation of the first row of A.
Step 1:
Input: $(a_0,\, a_1,\, a_2, \ldots, a_7) = (1,\, 0,\, 0, \ldots, 0)$
$\hat{a}_0 \,+\, \hat{a}_1 x \,+ \ldots +\, \hat{a}_7 x^7 =$
$(a_0 \,+\, a_1 x \,+ \ldots +\, a_7 x^7)(b_{1,0} \,+\, b_{1,1} x \,+ \ldots +\, b_{1,7} x^7) \, mod \, g_1(x)$
Output: $Row_1 = (\hat{a}_0,\, \hat{a}_1, \ldots, \hat{a}_7)$
Calculation of the row j of A, $\mathbf{2 \leq j \leq 8}$.
Steps from 1 to j-1:
Input: $(a_0,\, a_1, \ldots, a_7) = (0,\, 0, \ldots, 1, \ldots, 0)$. (the canonical vector $(0,\, 0, \ldots, 1, \ldots, 0)$ has a one in the *j-th* position)
For $i = j$ downto 2 do
Begin
$\hat{a}_0 = a_0 + c_{i,0} a_{i-1},\, \hat{a}_1 = a_1 + c_{i,1} a_{i-1}, \ldots, \hat{a}_{i-2} = a_1 + c_{i,1} a_{i-1}$
$\hat{a}_{i-1} + \hat{a}_i x + \ldots + \hat{a}_7 x^{8-i} =$
$(a_{i-1} + a_i x + \ldots + a_7 x^{8-i})(b_{i,0} + b_{i,1} x + \ldots + b_{i,8-i}) \, mod \, g_i(x)$
$(a_0,\, a_1, \ldots, a_7) = (\hat{a}_0,\, \hat{a}_1, \ldots, \hat{a}_7)$
End
Step j:
Input: $(a_0,\, a_1, \ldots, a_7)$
$\hat{a}_0 + \hat{a}_1 x + \ldots + \hat{a}_7 x^7 = (a_0 + a_1 x + \ldots + a_7 x^7)(b_{1,0} + b_{1,1} x + \ldots + b_{1,7} x^7) \, mod \, g_1(x)$
Output: $Row_j = (\hat{a}_0,\, \hat{a}_1, \ldots, \hat{a}_7)$
End

Output: Matrix $A = \begin{pmatrix} Row_1 \\ Row_2 \\ \ldots \\ Row_8 \end{pmatrix}$

2.3.2 Algorithm for the random generation of MDS matrices.

The algorithm for the random generation of a MDS matrix $A = \{a_{i,j}\}_{4\times 4}$, where for all i and j, $a_{i,j} \in GF(2^8)$, part of the algorithm for the random generation of an invertible matrix $A = \{a_{i,j}\}_{4\times 4}$, where for all i and j, $a_{i,j} \in GF(2^8)$ see [FDDP14], that is similar in its structure to the algorithm of the previous epigraph.

Input:
- Primitive polynomials $g_1(x)$, $g_2(x)$ y $g_3(x) \in GF(2^8)[x]$ (They are selected a priori and $gr(g_1(x)) = 4$, $gr(g_2(x)) = 3$ and $gr(g_3(x)) = 2$).
- Random matrix M.

where: $c_{i,j}$ and $b_{k,t} \in GF(2^8)$, $(b_{k,0},\, b_{k,1}, \ldots, bk, 4-k) \neq 0$, $k = 1..4$, $t = 0..3$, $i = 2..4$ and $j = 0..2$.

Begin

Calculation of the first row of matrix A.

Step1:

Input: $(a_0, a_1, a_2, a_3) = (1, 0, 0, 0)$.

$\hat{a}_0 + \hat{a}_1 x + \hat{a}_2 x^2 + \hat{a}_3 x^3 =$

$(a_0 + a_1 x + a_2 x^2 + a_3 x^3)(b_{1,0} + b_{1,1} x + b_{1,2} x^2 + b_{1,3} x^3) \, mod \, g_1(x)$

Output: $Row_1 = (\hat{a}_0, \hat{a}_1, \hat{a}_2, \hat{a}_3)$

Calculation of the row j of A, $\mathbf{2 \leq j \leq 4}$.

Steps from 1 to j-1:

Input: $(a_0, a_1, \ldots, a_4) = (0, \ldots, 1, \ldots, 0)$. (the canonical vector $(0, \ldots, 1, \ldots, 0)$ has a one in the *j-th* position)

For $i = j$ downto 2 do

Begin

$\hat{a}_0 = a_0 + c_{i,0} a_{i-1}, \hat{a}_1 = a_1 + c_{i,1} a_{i-1}, \ldots, \hat{a}_{i-2} = a_1 + c_{i,1} a_{i-1}$

$\hat{a}_{i-1} + \hat{a}_i x + \ldots + \hat{a}_3 x^{4-i} =$

$(a_{i-1} + a_i x + \ldots + a_3 x^{4-i})(b_{i,0} + b_{i,1} x + \ldots + b_{i,4-i} x^{4-i}) \, mod \, g_i(x)$

$(a_0, a_1, \ldots, a_3) = (\hat{a}_0, \hat{a}_1, \ldots, \hat{a}_3)$

End

Step j:

Input: $(a_0, a_1, \ldots, a_3)$

$\hat{a}_0 + \hat{a}_1 x + \ldots + \hat{a}_3 x^3 = (a_0 + a_1 x + \ldots + a_3 x^3)(b_{1,0} + b_{1,1} x + \ldots + b_{1,3} x^3) \, mod \, g_1(x)$

Output: $Row_j = (\hat{a}_0, \hat{a}_1, \ldots, \hat{a}_3)$

End

Output: Matrix $A = \begin{pmatrix} Row_1 \\ Row_2 \\ Row_3 \\ Row_4 \end{pmatrix}$

Notice that in the algorithm previously described in the *i-th* row, $i \geq 2$, the output values in the last vector $(\hat{a}_0, \hat{a}_1, \hat{a}_2, \hat{a}_3)$ can be expressed in the following way: $\hat{a}_j = a_j b_{1,0} \oplus r_j \, j \in \{0..3\}$.

Now, let's see the algorithm for the random generation of a MDS matrix $A_{4\times 4} = \{a_{i,j}\}_{4\times 4}$, where for all i and j, $a_{i,j} \in GF(2^8)$. In this algorithm, it is used the fact that an A matrix is MDS iff all its squared sub-matrices are not singular.

MDSMatrixGeneration
Input:

- Primitive polynomials $g_1(x)$, $g_2(x)$ and $g_3(x) \in GF(2^8)[x]$ (They are selected a priori and $gr(g_1(x)) = 4$, $gr(g_2(x)) = 3$ and $gr(g_3(x)) = 2$)
- Random matriz M

$$M = \begin{pmatrix} - & b_{1,1} & b_{1,2} & b_{1,3} \\ c_{2,0} & b_{2,0} & b_{2,1} & b_{2,2} \\ c_{3,0} & c_{3,1} & b_{3,0} & b_{3,1} \\ c_{4,0} & c_{4,1} & c_{4,2} & b_{4,0} \end{pmatrix}$$

where: $c_{i,j}$ and $b_{k,t} \in GF(2^8)$, $(b_{k,0}, b_{k,1}, \ldots, b_{k,4-k}) \neq 0$, $k = 2..4$, $t = 0..3$, $i = 2..4$, $j = 0..2$ and $b_{1,1}$, $b_{1,2}$ and $b_{1,3} \neq 0$.

Begin
1.- First row of matrix A:

The first row of matrix A is formed by $b_{1,0}$, $b_{1,1}$, $b_{1,2}$ and $b_{1,3}$. Values $b_{1,1}$, $b_{1,2}$ and $b_{1,3}$ are taken from matrix M then $a_{1,1} = b_{1,1}$, $a_{1,2} = b_{1,2}$, $a_{1,3} = b_{1,3}$. The value $b_{1,0}$ will be determined in step 3 of the present algorithm.

2.- *i-th* row of matrix A, $\mathbf{2 \leq i \leq 4}$:

From the values from the first up to the i-th row, matrix M and the previous algorithm, values a_j and r_j de $\hat{a}_{i,j} = \hat{a}_j = a_j b_{1,0} \oplus r_j$, $j \in \{0..3\}$ are calculated, leaving the i-th row of matrix A in the following way:

$$A = \begin{pmatrix} - & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ - & - & - & - \\ \hat{a}_{i,0} & \hat{a}_{i,1} & \hat{a}_{i,2} & \hat{a}_{i,3} \end{pmatrix}$$

- Values $\hat{a}_{i,j}$, $j \in \{0..3\}$, are done equal to zero and lineal equations with $b_{1,0}$ as variables are formed.
- The determinants of all sub-matrices of 2×2, which were not obtained in the previous steps are calculated. The determinants are equalled to zero and the lineal and quadriatic equations are formed with $b_{1,0}$ as variant.
- The determinants of all sub-matrices of 3×3, which were not obtained in the previous steps are calculated. The determinants are equalled to zero and the previous quadriatic and cubic equations are formed with $b_{1,0}$ as variant.
- The values of $b_{1,0}$ which do not satisfy the previous equations are stored.

3.- Random generation of matrix A.

From the values of $b_{1,0}$ which do not satisfy the previous equations, one should be selected at random, leaving matrix M full, as it will be shown now. For the Schedule of keys which is described above, the selection of $b_{1,0}$ is done by taking the necessary bits that allow to conform a number that is the biggest integer minor or equal to the cardinal of the set of the posible $b_{1,0}$.

$$M = \begin{pmatrix} - & b_{1,1} & b_{1,2} & b_{1,3} \\ c_{2,0} & b_{2,0} & b_{2,1} & b_{2,2} \\ c_{3,0} & c_{3,1} & b_{3,0} & b_{3,1} \\ c_{4,0} & c_{4,1} & c_{4,2} & b_{4,0} \end{pmatrix}$$

Observations:
If for any of the equations formed in the *i-th* row, $i \geq 2$, all the cofficients are equal to zero, a new value $c_{i,0}$ is randomly selected for matrix M, then, the values a_j and r_j, $j \in \{0..3\}$ are again calculated and the whole process is repeated, subsequently, we continue with the algorithm for the remaining rows.

If we take matrix $M = \begin{pmatrix} b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ c_{2,0} & b_{2,0} & b_{2,1} & b_{2,2} \\ c_{3,0} & c_{3,1} & b_{3,0} & b_{3,1} \\ c_{4,0} & c_{4,1} & c_{4,2} & b_{4,0} \end{pmatrix}$ in the following way $M = \begin{pmatrix} b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ then, the multiplication of a vector (a_0, a_1, a_2, a_3) by the MDS matrix A obtained by the previous algorithm is:

Input: (a_0, a_1, a_2, a_3)

$$a_0 = a_0 + c_{2,0}a_1 + c_{3,0}a_2 + c_{4,0}a_3$$

$[\hat{a}_0, \hat{a}_1, \hat{a}_2, \hat{a}_3] = (a_0+a_1x+a_2x^2+a_3x^3)(b_{1,0}+b_{2,0}x+b_{2,1}x^2+b_{2,2}x^3)\,mod\,g_1(x)$

Output: $(\hat{a}_0; \hat{a}_1; \hat{a}_2; \hat{a}_3)$

In addition to multiplying the vector (a_0, a_1, a_2, a_3) by the inverse matrix A^{-1} from MDS matrix A, see [FDM09], is done in the following way:

Input: (a_0, a_1, a_2, a_3)

$[\hat{a}_0, \hat{a}_1, \hat{a}_2, \hat{a}_3] = (a_0+a_1x+a_2x^2+a_3x^3)(b_{1,0}+b_{2,0}x+b_{2,1}x^2+b_{2,2}x^3)\,mod\,g_1(x)$

$$\hat{a}_0 = \hat{a}_0 + c2, 0a_1 + c_{3,0}a_2 + c_{4,0}a_3$$

Output:$(\hat{a}_0; \hat{a}_1; \hat{a}_2; \hat{a}_3)$

Now we will show an example of determinating a MDS matrix A through the previous algorithm.

Input:

- Primitive polynomials:

$g_1(x) = x^4 \oplus z^9x^3 \oplus z^2x \oplus z^{13}$

$g_2(x) = x^3 \oplus z^{16}x \oplus z^{53}$

$g_3(x) = x^2 \oplus z^{67}x \oplus z^{14}$

$z = 0X03$ it is primitive element of the $GF(2^8)$.

- Random matrix $M = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 7C & 9F & EA & 1A \\ 52 & 74 & B2 & 08 \\ 5E & D1 & OF & 2F \end{pmatrix}$

The $GF(2^8)$ was constructed from the irreductible polynomial $1 \oplus x^2 \oplus x^3 \oplus x^4 \oplus x^8$ (read in hexadecimal notation as B8).

The posible values for $b_{1,0}$ are:

```
02, 03, 04, 05, 06, 07, 08, 09, 0A, 0B, 0D, 0E, 0F, 10,
12, 13, 14, 15, 16, 17, 18, 1C, 1D, 1E, 1F, 20, 21, 22,
23, 25, 26, 27, 29, 2A, 2C, 2D, 2E, 30, 31, 33, 35, 36,
38, 39, 3A, 3B, 3C, 3D, 3E, 40, 41, 42, 43, 44, 45, 46,
47, 49, 4A, 4F, 50, 51, 52, 54, 55, 56, 58, 59, 5B, 5C,
5D, 5E, 5F, 60, 61, 62, 63, 65, 66, 67, 69, 6A, 6B, 6D,
6E, 70, 71, 72, 75, 76, 77, 79, 7A, 7B, 7C, 7E, 80, 81,
82, 83, 84, 85, 86, 88, 89, 8A, 8B, 8C, 8E, 8F, 90, 91,
92, 94, 95, 96, 97, 98, 99, 9A, 9B, 9C, 9F, A0, A2, A4,
A5, A6, A7, AA, AB, AD, AE, B0, B1, B2, B3, B5, B6, B8,
B9, BA, BB, BC, BD, BE, BF, C0, C1, C2, C4, C5, C7, C8,
C9, CB, CC, CD, CE, CF, D1, D2, D3, D4, D5, D8, D9, DA,
DB, DC, DD, DF, E0, E1, E3, E5, E7, E8, E9, EA, EC, ED,
EE, EF, F0, F1, F3, F4, F5, F9, FA, FB, FC, FD, FE, FF,
```

Taking $b_{1,0} = 2A$ then, the matrix M will be: $\begin{pmatrix} 2A & 03 & 01 & 01 \\ 7C & 9F & EA & 1A \\ 52 & 74 & B2 & 08 \\ 5E & D1 & OF & 2F \end{pmatrix}$ and

the matrix $A = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}$

Notice that the matrix obtained is MDS matrix which is used in the Cryptographic Algorithm AES [DR99] and [DR02].

3 Conclusions:

The variations carried out in this paper to the Cryptographic Algorithms AES [DR99] and [DR02] and Twofish [SKWHF98] and [GBS13] differ to others which are reported in the especializad literature. We have proposed that the MDS matrices from both algorithms vary in the function of the key, and that in the AES the S - box also varies as function of the key and we propose a new Schedule of the key in order to avoid the differential cryptanalysis of the key related as it is suggested in [DR12]. It is advisable to change the S - box of the AES, S_{RD}, by the one used in [DD13].

The porposal to vary in the function of the key the MDS matrices of the Cryptographic Algorithms AES and Twofish can be extended to any Cryptographic Algorithm that uses matrices MDS $A = \{a_{i,j}\}_{n \times n}$, where for all i, j, $a_{i,j} \in GF(2^8)$ and $n \leq 5$ [FDDP14].

References

[AE13] Ahmed F. and Elkamchouchi D. Strongest AES with S-Boxes bank and dynamic key MDS matrix (SDK-AES). International Journal of Computer and Communication Engineering, Vol. 2, No. 4, July 2013.

[AHK13] Arrag S., Hamdoun A., Tragha A., and Khamlich S. E. Implementation of stronger AES by using dynamic S-box dependent of master key Journal of Theoretical and Applied Information Technology. Marruecos. 2013.

[AF14] Augot D. and Finiasz M. Direct construction of recursive MDS diffusion layers using shortened BCH code. Lix. INRIA Saday - Île de France. 2014.

[AF13] Augot D. and Finiasz M. Exhaustive search for small dimension recursive MDS diffusion layers for block ciphers and hash functions. In information Theory Proceedings (ISIT), 2013. IEEE International Symposium. pp. 1551-1555. 2013.

[BKN09] Biryukov A., Khovratovich D. and Nikolić I. Distinguisher and related-key attack on the full AES-256. In CRYPTO'09, volume 5677 of LNCS, pp. 231-249. Springer. 2009.

[BK09] Biryukov A. and Khovratovich D. Related-key cryptanalysis of the full AES-192 and AES-256 . Lecture Notes in Computer Science, 6477, pp. 1-19. 2009.

[CZKHP11] Choy J., Zhang A., Khoo K., Henricksen M. and Poschmann A. AES variants secure against related-key differential and boomerang attacks. Workshop on Information Security Theory and Practice (WISTP). 2011.

[DD13] Dolmatov V. and Degtyarev A. GOST R 34.11-2012: Hash Function. Independent Submission, Ed. Request for Comments: Updates: 5831 Cryptocom, Ltd. Category: Informational. ISSN: 2070-1721. August. 2013.

[DMMF15] Dehnavi S. M., Mirzaee M. R., Mahmoodi A. and Fekri Y. Efficient MDS Diffusion Layers Through Decomposition of Matrices. IACR Cryptology. ePrint Archive. 2015.

[DMMMP14] Dehnavi S. M., Mahmoodi A., Mirzaee M. R., Maimani H. and Pasha E. Construction of new families of MDS diffusion layers. IACR Cryptology ePrint. 2014.

[DR99] Daemen J. and Rijmen V. The Rijndael block cipher. AES proposal. http/www.daimi.ai.dk/$\sim$ iran/rijndael.pdf. 1999.

[DR02] Daemen J. andRijmenV. The design of Rijndael: AES - The Advanced Encryption Standard. Springer-Verlag. 2002.

[DR12] Daemen J. and Rijmen V. On the related-key attacks agains AES. Proceedings of Romanian Acadamy, Serie A, Vol. 13. Nro. 4. pp. 395-400. 2012.

[ERDM09] ElGhafar A., Rohiem A., Diaa A. and Mohammed F. Generation of AES key dependent S-Boxes using RC4 Algorithm. Military Technical College, Kobry Elkobbah, Cairo, Egypt. 2009.

[FDDP14] Freyre P.,Díaz N., Díaz R. and Pérez C. Random generation of matrices MDS. Proceeding CD. CTCrypt2014. Moscow. 2014.

[FDM09] Freyre P., Díaz N. and Morgado E. R. Fast algorithm for the multiplication of a row vector by a randomly selected matrix A. Journal of Discrete Mathematical Sciences & Cryptography Vol. 12 No. 5, pp. 533-549. India. 2009.

[FSESH05] Fahmy A., Shaarawy M., El-Hadad K., Salama G. and Hassanain K. A Proposal For A Key-Dependent AES. 3^{rd}. International Conference: Sciences of Electronic, Technologies of Information and Telecommunications. March 27-31, 2005.

[GBS13] Gehlot P., Biradar S. R. and Singh B. P. Implementation of Modified Twofish Algorithm using 128 and 192-bit keys on VHDL. International Journal of Computer Applications. Volume 70 No.13, May 2013.

[GR13] Gupta K. C. and Ray I. G. On Constructions of Involutory MDS Matrices. In AFRICA CRYPT2013, pp 43-60. 2013.

[GR13a] Gupta K. C. and Ray I. G. On constructions of MDS matrices from companion matrices for lightweight cryptography. In CD-ARES. 2013 Workshop: MOCrySEn, pp. 29-43, Springer. 2013.

[GR14] Gupta K. C. and Ray I. G. On constructions of MDS matrices from circulant-like matrices for lightweight cryptography. Applied Statistics Unit, Indian Statistical Institute. Calcuta. India. 2014.

[IGKAE12] IsmailI A., Galal-Edeen G. H., Khattab S., AbdElhamid M. and El Bahtity M. Performance examination of AES encryption algorithms with constant and dynamic rotation, Faculty of Computer and Information, Cairo University, Egypt. 2012.

[JD04] Jakimoski G. and Desmedt Y. Related-Key differential cryptanalysis of 192 bit key AES variants. SAC 2003, LNCS, 3006, pp. 208-221. Springer-Verlag. 2004.

[JMV15] Jacob G., Murugan A. and Viola I. Towards the Generation of a Dynamic Key-Dependent S-Box to Enhance Security. IACR Cryptology ePrint Archive. 2015.

[JV04] Junod P. and Vaudenay S. Perfect diffusion primitives for Block Ciphers. Building efficient MDS matrices. In: Selected Areas in Cryptography. 2004.

[KK09] Kazlauskas K. and Kazlauskas J. Key-Dependent S-Box Generation in AES Block Cipher System. INFORMATICA. *Institute of Mathematics and Informatics*. Vilnius. Vol. 20, No. 1. 2009.

[Ke97] Keliher L. A New Substitution-Permutation Network CipherUsing Key-Dependent S-Boxes. Thesis for the degree of Master of Science. Queens University. Canada. September. 1997.

[KM08] Krishnamurthy G. N. and Ramaswamy V. Making AES Stronger: AES with Key Dependent S-Box. IJCSNS International Journal of

Computer Science and Network Security, VOL.8 No.9, September. 2008.

[KM14] Kolay S. and Mukhopadhyay D. Lightweight diffusion layer from k^{th} root of the MDS matrix. IACR Cryptology ePrint. 2014.

[LF04] Lacan J. and Fimes J. Systematic MDS erasure codes based on vandermonde matrices. IEEE Trans. Com-mun. Lett. 8(9), 570-572 CrossRef. 2004.

[MEEZ13] Mahmoud E. M., ElHafez A. A., Elgarf T. A. and Zekry A. Dynamic AES-128 with key-dependent S-box. International Journal of Engineering Research and Applications. Egypt. 2013.

[MJ11] Malik M. Y. and Jong-Seon. Dynamic MDS matrices for substantial cryptographic strength. Seoul National University. 2011.

[MI08] Murtaza G. and Ikram N. New methods of generating MDS matrices. Proceedings of International Cryptology Workshop and Conference. 2008.

[MI11] Murtaza G. and Ikram N. Direct exponent and scalar multiplication classes of an MDS matrix. IACR Cryptology. ePrint Archive. 2011.

[MKAF11] Murtaza G., Khan A. A., Alam S. W. and Farooq A. Fortification of AES with dynamic mix-column Transformation. National University of Science and Technology. Islamabad. Pakistan. 2011.

[MRE09] Mohammad F. Y., Rohiem A. E. and Elbayoumy A. D. A Novel S-box of AES Algorithm Using Variable Mapping Technique. 13^{th}. International Conference on AEROSPACE SCIENCES & AVIATION TECHNOLOGY. ASAT- 13. 2009.

[NA09] Nakahara J. Jr. and Abrahao E. A New involutory MDS matrix for the AES. International Journal of Network Security, 9(2), pp.109-116. 2009.

[RRYB15] Ruoxin Z., Rui Z., Yongqiang L. and Baofeng W. On Constructions of a Sort of MDS Block Diffusion Matrices for Block Ciphers and Hash Functions. IACR Cryptology. ePrint Archive. 2015.

[SDMO12] Sajadieh M., Dakhilalian M., Mala H. and Omoomi B. On construction of involutory MDS matrices from Vandermonde matrices in $GF(2^q)$. Des. Codes Cryptography. Vol.64, No.3, pp. 287-308. 2012.

[SKOP15] Sim S. M., Khoo K., Oggier F. and Peyrin T. Lightweight MDS Involution Matrices (Full version). IACR Cryptology. ePrint Archive. 2015.

[SKWHF98] Schneier B., Kelsey J., Whiting D., Wagner D., Hall C. and Ferguson N. Twofish: A 128-Bit Block Cipher. Available from NIST's AES homepage http://www.nist.gav/aes. 1998.