Block Ciphers with Matrices Operating Alternately over Columns and Rows

Pablo Freyre, Oristela Cuellar, Adrián Alfonso, Nelson Díaz

***Abstract*—In this paper, we present the dynamic cryptographic algorithms for long states named ACDEL-2D and ACDEL-3D. The first one was inspired by Rijndael and the second one was inspired by 3D, a three-dimensional block cipher. In both proposals, MDS matrices are used alternately on rows and columns of the state and all transformations used in the encryption process are randomly selected depending on pseudorandom sequences. In the block cipher ACDEL-3D the state takes the form of a rectangular parallelepiped or cuboid.**

***Tóm tắt*—Trong bài báo này, chúng tôi trình bày các thuật toán mật mã động cho các trạng thái dài, có tên là ACDEL-2D và ACDEL-3D. Thuật toán đầu tiên bắt nguồn từ thuật toán Rijndael và thuật toán thứ hai bắt nguồn từ thuật toán 3D, một thuật toán với khối dữ liệu được biểu diễn ở dạng ba chiều. Cả hai đề xuất đều sử dụng xen kẽ ma trận MDS trong các hàng và cột của trạng thái và tất cả các phép biến đổi được sử dụng trong quá trình mã hóa được chọn ngẫu nhiên tùy thuộc vào chuỗi giả ngẫu nhiên. Trong mật mã khối ACDEL-3D, trạng thái có dạng hình chữ nhật song song hoặc hình khối.**

Keywords—block cipher; Rijndael; random transformations; three-dimensional state.

Từ khóa—mật mã khối; Rijndael; biến đổi ngẫu nhiên; trạng thái ba chiều.

I. INTRODUCTION

Rijndael is a cryptographic algorithm designed by the Belgian Joan Daemen and Vincent Rijmen and submitted to the AES competition in 1997 [1]. Announced as a winner in 2001, Rijndael was adopted as the standard AES [2] with some specifications in terms of block and key sizes.

Rijndael uses the transformations SubBytes, ShiftRows, MixColumns and AddRoundKey, all of them are fixed and selected a priori [3], acting on a two-dimensional state. Since the appearance of Rijndael several cryptographic algorithms based on its design have surged, some of them with random transformations [4]-[10].

Another variant of Rijndael is the block cipher 3D, using fixed transformations on a three-dimensional state [11]. Similar constructions are 3D-AES [12], HiSea [13], DOZEN [14], the hash function Keccak [15] and Non-Alternate 3D [16].

Our contributions: In this paper, two fully dynamic cryptographic algorithms for long states named ACDEL-2D and ACDEL-3D (from Spanish **A**lgoritmo **C**riptográfico **D**inámico para **E**stados **L**argos) are presented, inspired by the design of the block ciphers Rijndael and 3D. In both proposals, all transformations in the encryption process are chosen depending on pseudorandom sequences and MDS matrices operating alternately on rows and columns of the state are used. The state in ACDEL-2D is a two-dimensional matrix and in ACDEL-3D is a three-dimensional cuboid; in both cases, message blocks longer than 512 bits can be encrypted. Additionally, a new method for the generation of 4x4 MDS matrices in $GF(2^8)$ is proposed.

This paper begins with a brief description of the cryptographic algorithms Rijndael and 3D in *Section II*. In *Section III* and *Section IV*, the dynamic block ciphers ACDEL-2D and ACDEL-3D are presented as well as the random transformations used during the encryption processes. The necessary algorithms for the random generation of these transformations are shown in *Section V* and further comments on design strategy and security are discussed in *Section VI*. The paper is finished in *Section VII* with the conclusion.

II. RIJNDAEL AND 3D

The operations of the cryptographic algorithm Rijndael are performed in the Galois field $GF(2^8)$, so the input block and the output block are arrays of $4N_b$ bytes for each one, where $4 \leq N_b \leq 8$.

This manuscript is received on May 22, 2020. It is commented on July 15, 2020 and is accepted on July 15, 2020 by the first reviewer. It is commented on June 23, 2020 and is accepted on December 08, 2020 by the second reviewer.

The bytes of the input block $p_0 p_1 \dots p_{N_b-1}$ are located inside a matrix with 4 rows and N_b columns named state

$$S = \begin{pmatrix} s_{0,0} & s_{0,1} & \cdots & s_{0,N_b-1} \\ s_{1,0} & s_{1,1} & \cdots & s_{1,N_b-1} \\ s_{2,0} & s_{2,1} & \cdots & s_{2,N_b-1} \\ s_{3,0} & s_{3,1} & \cdots & s_{3,N_b-1} \end{pmatrix}$$

so that $s_{i,j} = p_{i+4j}$ for $0 \le i < 4$ and $0 \le j < N_b$.

The secret key is another array of bytes located inside a matrix with 4 rows and N_k columns, where $4 \le N_k \le 8$, which splits through the key schedule in a matrix with 4 rows and N_b columns for every round of the encryption process. The number of rounds N_r depends on the size of the block and the key and it is computed as $N_r = 6 + \max\{N_b, N_k\}$.

In each round, the following transformations act on the state matrix offering confusion and diffusion:

- **SubBytes**, acting like S-box on every state byte.
- **ShiftRows**, performing cyclic rotations on the rows of the state.
- **MixColumns**, multiplying every column of the state by one MDS matrix.
- **AddRoundKey**, a bitwise XOR of the state with the round key.

The encryption process and the key schedule of Rijndael can be seen in [3]. Here we present the encryption process of Rijndael (from two rounds in two rounds) in pseudocode in view to simplify the presentation of our dynamic proposals.

Encryption process of Rijndael

```
Rijndael(State, CipherKey)
{
KeyExpansion(CipherKey, ExpandedKey)
AddRoundKey(State, ExpandedKey[0])
for (i = 1; i < Nr − 1; i + 2)
{
SubBytes(State)
ShiftRows(State)
MixColumns(State)
AddRoundKey(State, ExpandedKey[i])
SubBytes(State)
ShiftRows(State)
MixColumns(State)
AddRoundKey(State, ExpandedKey[i + 1])
}
SubBytes(State)
ShiftRows(State)
MixColumns(State)
AddRoundKey(State, ExpandedKey[Nr − 1])
SubBytes(State)
ShiftRows(State)
AddRoundKey(State, ExpandedKey[Nr])
}
```

On the other hand, the state in the block cipher 3D is a three-dimensional cube with 4 rows, 4 columns and 4 lanes. 3D uses a fixed S-box like SubBytes, a fixed MDS matrix like MixColumns and two fixed transformations like ShiftRows acting in alternate rounds.

Let γ be the S-box, π the MDS matrix, θ_1 the rotations on slices, θ_2 the rotations on sheets, and k_i the XOR of the state with the i-th round sub-key for $0 \le i \le N_r$, then the encryption process of 3D (from two rounds in two rounds) in pseudocode is presented next.

Encryption process of 3D

```
3D(State, CipherKey)
{
KeyExpansion(CipherKey, ExpandedKey)
for (i = 1; i < Nr − 2; i + 2)
{
k_i(State, ExpandedKey[i])
γ(State)
θ1(State)
π(State)
k_{i+1}(State, ExpandedKey[i + 1])
γ(State)
θ2(State)
π(State)
}
k_{Nr−2}(State, ExpandedKey[Nr − 2])
```

γ(State)

θ_1(State)

π(State)

k_{N_r-1}(State, ExpandedKey[$N_r - 1$])

γ(State)

θ_2(State)

k_{N_r}(State, ExpandedKey[N_r])

}

In the case of Rijndael, full diffusion is reached in two rounds if $N_b = 4$ or three rounds in other cases due to the properties of the diffusion layer, which is ensured because ShiftRows is a diffusion optimal permutation.

In the case of 3D, full diffusion is reached in three rounds since the transformations θ_1 and θ_2 are both diffusion optimal permutations and the number of rounds is $N_r = 22$. The main advantage of 3D is to encrypt message blocks of 512 bits, while Rijndael encrypts message blocks of 256 bits at most.

III. ACDEL-2D

In this section, we propose the dynamic block cipher ACDEL-2D where the state is a two-dimensional matrix with 4 rows and N_b columns, for $N_b \geq 4$ is any natural number. The pseudocode of the encryption process is presented below.

Encryption process of ACDEL-2D

ACDEL − 2D(State, CipherKey)

{

KeyExpansion(CipherKey, ExpandedKey)

RandomSubBytes(sequence$_1$, SubBytes)

RandomMixColumns(sequence$_2$, MixColumns)

RandomMixRows(sequence$_3$, MixRows)

RandomAffineKey(sequence$_4$, AffineKey)

AffineKey(State, ExpandedKey[0])

for ($i = 1; i < N_r - 1; i + 2$)

{

SubBytes(State)

MixColumns(State)

AffineKey(State, ExpandedKey[i])

SubBytes(State)

MixRows(State)

AffineKey(State, ExpandedKey[$i + 1$])

}

SubBytes(State)

MixColumns(State)

AffineKey(State, ExpandedKey[$N_r - 1$])

SubBytes(State)

AffineKey(State, ExpandedKey[N_r])

}

Here sequence$_i$ is a pseudorandom sequence for all $1 \leq i \leq 4$ obtained through the key schedule or any pseudorandom number generator.

A. RandomSubBytes

In Rijndael, the transformation SubBytes acts on every byte of the state like a S-box denoted as S_{DR}, constructed through a non-affine transformation and an affine transformation [3]. S_{DR} was selected by the Rijndael's designers taking in mind a complex algebraic expression; however, we consider to use a random S-box so that its algebraic expression will be unknown.

In this paper, we present two possibilities for the random transformation RandomSubBytes.

The first possibility is to generate a random invertible matrix in the general linear group $GL_{8x8}(GF(2))$ used to construct a random affine transformation. It is combined with the non-affine transformation of SubBytes, so RandomSubBytes acts like a random S-box $RS_{DR}[x]$.

The second possibility is to construct a random S-box independent form S_{DR} through a random permutation Π of the symmetric group S_{256}, this way RandomSubBytes acts like the random S-box $RS_{DR}[x] = \Pi[x]$.

B. RandomMixColumns and RandomMixRows

In Rijndael, the transformation MixColumns acts on the columns of the state multiplying the same ones by a MDS matrix and the transformation ShiftRows acts on the rows of the state cyclically rotating their bytes to the left [3]. In ACDEL-2D, the random transformations RandomMixColumns and RandomMixRows act in alternate rounds on the state.

RandomMixColumns is a random MDS matrix in $GL_{4x4}(GF(2^8))$ providing high diffusion into the columns of the state, and RandomMixRows is a random MDS matrix in $GL_{N_b x N_b}(GF(2^8))$ providing high diffusion into the rows of the state.

The value of N_b can increase as much as desired since the random MDS matrix in RandomMixRows can be constructed through a fixed MDS matrix from a Reed-Solomon code, by swapping the rows depending on a random permutation in S_{N_b} and multiplying each row by a different non-zero element in $GF(2^8)$.

C. RandomAffineKey

In this paper, three ways to introduce the round key in the round function are proposed. The first way consists of a bitwise XOR like in Rijndael.

The second way consists of a random affine transformation on every byte of the state of the form:

$$L \cdot s_{i,j} \oplus \text{ExpandedKey}[t]_{i,j}$$

where L is a random invertible matrix in $GL_{8x8}(GF(2))$ for all $0 \le i < 4$, $0 \le j < N_b$ and $0 \le t \le N_r$.

The third way, also proposed in [17] for the block cipher SHARK, consists of two random affine transformations AffineKey_1 and AffineKey_2 on the state of the form:

$$\text{AffineKey}_1(S) = S \cdot M_1 \oplus \text{ExpandedKey}[t]$$

where S is the state, M_1 is a random invertible matrix in $GL_{N_b x N_b}(GF(2^8))$, and t is odd.

$$\text{AffineKey}_2(S) = M_2 \cdot S \oplus \text{ExpandedKey}[t]$$

where S is the state, M_2 is a random invertible matrix in $GL_{4x4}(GF(2^8))$, and t is even.

Remark 1: ACDEL-2D can encrypt long message blocks and full diffusion is always achieved in 2 rounds. For example, in ACDEL-2D with $N_b = 16$ we can encrypt 512 bits and still full diffusion is reached in 2 rounds instead of 3D.

Remark 2: Although we present a state matrix with 4 rows, this number can increase to 5, since the first method presented in section V for the generation of random MDS matrices allows this increment.

Remark 3: In the third way of RandomAffineKey, two different transformations act in alternate rounds as shown above. Furthermore, the second way of RandomAffineKey is used only in combination with the second way of RandomSubBytes.

D. The key expansion and the number of rounds

ACDEL-2D was designed to support any cipher key multiple of 32 bits (length of a column) from a minimum size of 256 bits.

Note that each state matrix is formed by N_b columns, then the key expansion procedure splits the cipher key into the $N_r + 1$ round keys needed in the encryption process.

Let N_k be the number of columns of the cipher key, then columns W_i for all $N_k \le i < N_b(N_r + 1)$ are derived from the given columns of the cipher key W_j for $0 \le j < N_k$ as shown below. We clarify that this procedure to derive the round keys from the cipher key is the same as presented in [3] for Rijndael, where SW is the transformation SubBytes acting on each byte of the column and RW is a cyclic rotation from the column up.

if $(i \equiv N_k \bmod N_k)$

$W_i = W_{i-N_k} \oplus SW\big(RW(W_{i-1})\big) \oplus RC[i/N_k]$

else if $(i \equiv 4 \bmod N_k)$

$W_i = W_{i-N_k} \oplus SW(W_{i-1})$

else

$W_i = W_{i-N_k} \oplus W_{i-1}$

Since ACDEL-2D always achieves full diffusion in 2 rounds whatever N_b are, we define the number of rounds as $N_r = 14$, ensuring that the encryption and the decryption processes have the same structure.

Further comments on the selection criteria for this number will be given in *Section VI*.

E. Complexity by matrix multiplication

For $N_b = 4$, the number of multiplications in the field $GF(2^8)$ with ACDEL-2D is equal to the number of multiplications used to encrypt a message of the same length with Rijndael. For higher values of N_b, this number increases; however, when $N_b > 8$, only one execution of ACDEL-2D is carried out instead of Rijndael.

For example, to encrypt a 1024-bit message, ACDEL-2D performs 28160 multiplications; while Rijndael, for 256-bit blocks, runs 4 times needing 6656 multiplications, up to 4 times less. We assume that the MDS matrix of Rijndael is randomly chosen for our estimations; in practice with the MDS matrix of Rijndael, this number is less.

If the random affine transformations AffineKey_1 and AffineKey_2 are also used on the state, then ACDEL-2D is much more expensive than Rijndael in this sense. In this case, for $N_b = 4$, the number of multiplications is higher with respect to Rijndael.

This constitutes an obvious disadvantage of ACDEL-2D; however, it is suitable in applications where long messages need to be encrypted with high security and they can tolerate slower encryptions. Further comments on the design strategy of ACDEL-2D and its security are presented in *Section VI*.

IV. ACDEL-3D

In this section, we propose the dynamic block cipher ACDEL-3D where the state is a three-dimensional cuboid with 4 rows, N_b columns and N_h lines, for $N_b, N_h \geq 4$ are any natural numbers.

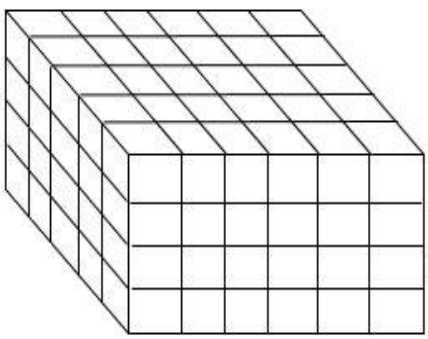


Fig. 1. Example of one three-dimensional state in ACDEL-3D with $N_b = 6$ and $N_h = 5$.

The state can be arranged as N_h matrices with 4 rows and N_b columns, each of which constitutes a slice, and the columns with the same position in each of these matrices form a sheet.

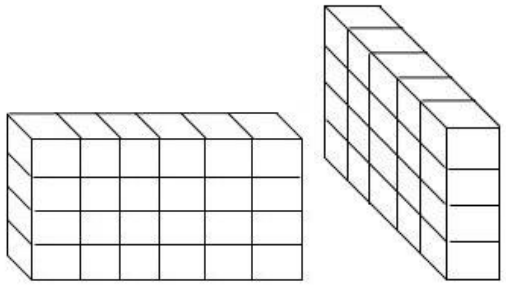


Fig. 2. Example of one slice (with 4 rows and $N_b = 6$ columns) and one sheet (with 4 rows and $N_b = 5$ lines) of the state shown in the Fig. 1.

The pseudocode of the encryption process is presented below.

Encryption process of ACDEL-3D

```
ACDEL-3D(State, CipherKey)
{
KeyExpansion(CipherKey, ExpandedKey)
RandomSubBytes(sequence_1, SubBytes)
RandomMixColumns(sequence_2, MixColumns)
RandomMixRows(sequence_3, MixRows)
RandomMixLanes(sequence_4, MixLanes)
RandomAffineKey_1(sequence_5, AffineKey_1)
RandomAffineKey_2(sequence_6, AffineKey_2)
RandomAffineKey_3(sequence_7, AffineKey_3)
RandomSwapCuboid(sequence_8, SwapCuboid)
AffineKey_3(State, ExpandedKey[0])
for (i = 1; i < N_r; i + 3)
{
SubBytes(State)
MixColumns(State)
AffineKey_1(State, ExpandedKey[i])
SubBytes(State)
MixRows(State)
AffineKey_2(State, ExpandedKey[i + 1])
SubBytes(State)
SwapCuboid(State)
MixLanes(State)
AffineKey_3(State, ExpandedKey[i + 2])
SwapCuboid^-1(State)
}
}
```

Here sequence_i is a pseudorandom sequence for all $1 \leq i \leq 8$ obtained through the key schedule or any pseudorandom number generator.

A. The random transformations

Let $S_1, S_2, \cdots, S_{N_h}$ be the N_h matrices that form the slices of the three-dimensional state, each of these with 4 rows and N_b columns, then the cuboid can be presented as a two-dimensional array

$$S = (S_1|S_2|\cdots|S_{N_h})$$

RandomSubBytes acts on every byte of the state cuboid as a random S-box constructed in the two possible ways shown in *Section III* for ACDEL-2D.

RandomMixColumns acts on the slices of the state cuboid multiplying all columns by a random MDS matrix α in $GL_{4x4}(GF(2^8))$ so that

$$\text{MixColumns(S)} = (\alpha \cdot S_1|\alpha \cdot S_2|\cdots|\alpha \cdot S_{N_h})$$

RandomMixRows acts on the slices of the state cuboid multiplying all rows by a random MDS matrix β in $GL_{N_b x N_b}(GF(2^8))$ so that

$$\text{MixRows(S)} = (S_1 \cdot \beta|S_2 \cdot \beta|\cdots|S_{N_h} \cdot \beta)$$

RandomMixLanes acts on the sheets of the state cuboid multiplying all lanes by a random MDS matrix μ in $GL_{N_h x N_h}(GF(2^8))$. In this case the state cuboid is presented as a two-dimensional array

$$S' = (S'_1|S'_2|\cdots|S'_{N_b})$$

where $S'_1, S'_2, \cdots, S'_{N_b}$ are the N_b matrices that form the sheets of the three-dimensional state, each of these with 4 rows and N_h columns. This way

$$\text{MixLanes}(S') = (S'_1 \cdot \mu|S'_2 \cdot \mu|\cdots|S'_{N_b} \cdot \mu)$$

Remark 4: For ACDEL-3D, it can be convenient to generate and store only one of the random MDS matrices β and μ, since one of these can be taken as a sub-matrix of the other. If two dimensions of the cuboid are equal then they can be convenient to use the same random MDS matrices for these dimensions.

Remark 5: The three-dimensional state used in ACDEL-3D is seen as a two-dimensional array of N_h matrices with 4 rows and N_b columns for the random transformations RandomMixColumns and RandomMixRows, and as a two-dimensional array of N_b matrices with 4 rows and N_h columns for the random transformation RandomMixLanes. Thus we can use N_h different MDS matrices α and β and N_b different MDS matrices μ for each one of the matrices $S_1, S_2, \cdots, S_{N_h}$ and $S'_1, S'_2, \cdots, S'_{N_b}$.

RandomAffineKey acts on the state cuboid by mixing the round keys in three possible ways. The first and second ways are the bitwise XOR and the random affine transformation on every byte of the state shown in *Section III* for ACDEL-2D. The third way consists of three random affine transformations AffineKey_1, AffineKey_2 and AffineKey_3 acting on the state of the next form:

Let α_1 be the affine transformation

$$\alpha_1(S_p) = S_p \cdot M_1 \oplus \text{ExpandedKey}[t]_p$$

where S_p is a slice matrix of the cuboid, M_1 is a random invertible matrix in $GL_{N_b x N_b}(GF(2^8))$, $1 \leq p \leq N_h$ and t is any round where α acts. Then

$$\text{AffineKey}_1(S) = (\alpha_1(S_1)|\alpha_1(S_2)|\cdots|\alpha_1(S_{N_h}))$$

Let β_1 be the affine transformation

$$\beta_1(S_p) = M_2 \cdot S_p \oplus \text{ExpandedKey}[t]_p$$

where S_p is a slice matrix of the cuboid, M_2 is a random invertible matrix in $GL_{4x4}(GF(2^8))$, $1 \leq p \leq N_h$ and t is any round where β acts. Then

$$\text{AffineKey}_2(S) = (\beta_1(S_1)|\beta_1(S_2)|\cdots|\beta_1(S_{N_h}))$$

Let μ_1 be the affine transformation

$$\mu_1(S'_p) = M_3 \cdot S'_p \oplus \text{ExpandedKey[t]}_p$$

where S'_p is a sheet matrix of the cuboid, M_3 is a random invertible matrix in $GL_{4x4}(GF(2^8))$, $1 \leq p \leq N_b$ and t is any round where μ acts. Then

$$\text{AffineKey}_3(S') = (\mu_1(S'_1)|\mu_1(S'_2)|\cdots|\mu_1(S'_{N_h}))$$

Remark 6: The encryption process of ACDEL-3D previously presented uses the third way of RandomAffineKey. In the first and second ways only one transformation needs to be generated, thus AffineKey_1, AffineKey_2 and AffineKey_3 are equal.

Remark 7: When the three dimensions of the state cuboid are equals ($N_b = N_h = 4$), the MDS matrices β and μ can be randomly generated from the set of all its possible choices as well as α by the methods presented in *Section V*.

RandomSwapCuboid operates between the two-dimensional states S and S', swapping the columns of the slices, the columns of the sheets and the bytes on all columns of the three-dimensional state. This way $(N_b!)^{N_h}(N_h!)^{N_b}(4!)^{N_bN_h}$ different swaps turn S into S' depending on N_h random permutations of S_{N_b}, N_b random permutations of S_{N_h} and N_bN_h random permutations in S_4.

Suppose that $\tau_0, \tau_1, \cdots, \tau_{N_h-1} \in S_{N_b}$, $\lambda_0, \lambda_1, \cdots, \lambda_{N_b-1} \in S_{N_h}$ and $\xi_0, \xi_1, \cdots, \xi_{N_bN_h-1} \in S_4$, then the byte in the j-th row and the k-th column of the i-th matrix of S' is computed as

$$S'_i[j,k] = S_{\lambda_k[i]}[\xi_{N_h i+k}[j], \tau_i[k]]$$

for all $0 \le j < 4$, $0 \le k < N_h$ and $0 \le i < N_b$.

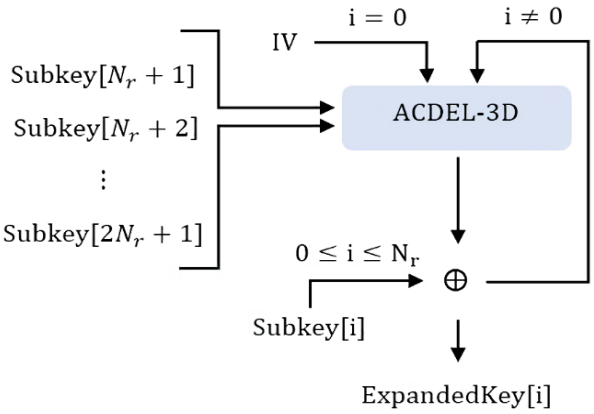
Remark 8: ACDEL-3D can encrypt long message blocks and full diffusion is always achieved in 3 rounds. The values of N_b and N_h can increase as much as desired since the random MDS matrices β and μ can be constructed through fixed MDS matrices from a Reed-Solomon code, by swapping its rows depending on random permutations and multiplying its rows by different non-zero elements in $GF(2^8)$.

B. The key expansion and the number of rounds

ACDEL-3D was designed to support any cipher key multiple of 32 bits (length of a column) from a minimum size of 512 bits.

Let N_k be the number of columns of the cipher key, then columns W_i for $N_k \le i < 2N_bN_h(N_r + 1)$ are derived from the given columns of the cipher key W_j for $0 \le j < N_k$ as shown in *Section III* for ACDEL-2D.

Once derived, in these $2(N_r + 1)$ sub-key matrices with N_bN_h columns, the first $(N_r + 1)$ ones are used as input for ACDEL-3D in CFB mode and the second $(N_r + 1)$ ones are used as the round keys. The $(N_r + 1)$ outputs are the round keys for the encryption process. This procedure is shown in the following figure.



Remark 9: In the key expansion procedure as many sub-keys as it is needed can be generated, so that the $N_r + 1$ round keys are guaranteed and also the pseudorandom sequences used for the generation of the random transformation. During the generation of these pseudorandom sequences if unwanted zero-elements are produced, these are ignored and additional key material is produced. In addition, as we have said before, any pseudorandom number generator can be used to produce the pseudorandom sequences.

Since ACDEL-3D always achieves full diffusion in 3 rounds whatever N_b and N_h are, we define the number of rounds as $N_r = 21$. Further comments on the selection criteria for this number will be given in *Section VI*.

On the contrary of ACDEL-2D, the decryption algorithm of ACDEL-3D is always performed in a straightforward way by using the inverses of the transformations of the encryption process.

C. Complexity by matrix multiplication

ACDEL-3D uses only one execution to encrypt a message with $4N_bN_h$ bytes, while 3D uses $N_bN_h/16$ executions in the best case that N_bN_h is a multiple of 16, and $\lceil N_bN_h/16 \rceil$ executions if not, either $N_b > 4$ or $N_h > 4$.

Similarly, Rijndael uses $N_bN_h/8$ executions if N_bN_h is a multiple of 8 and $\lceil N_bN_h/16 \rceil$ executions if not for all N_b and N_h. This way, ACDEL-3D can encrypt long messages in a single execution without using a block cipher mode of operation.

For example, if $N_b = N_h = 16$, we can encrypt a message with 8192 bits (1 KB of data) generating efficiently 16x16 random MDS matrices from the MDS matrix of the Russian standard Kuznyechik [18], while 3D needs 16 executions, Rijndael needs 32 executions and AES needs 64 executions.

In ACDEL-3D, the random MDS matrices α, β and μ operate in different rounds on each column, row and line of the cuboid respectively, this way RandomMixColumns uses $4^2 N_b N_h$ multiplications, RandomMixRows uses $4N_b^2 N_h$ multiplications and RandomMixLanes uses $4N_b N_h^2$ multiplications, so the number of multiplications during the encryption process is

$$\epsilon_1 = 28N_b N_h(4 + N_b + N_h)$$

In the most expensive case, ACDEL-3D uses the random MDS matrices α, β and μ and the random invertible matrices α_1, β_1 and μ_1, so the number of multiplications during the encryption process in this case is

$$\epsilon_2 = 28N_b N_h(64 + 2N_b + N_h)$$

For example, if $N_b = N_h = 16$ ACDEL-3D performs 258048 multiplications without using the matrices α_1, β_1 and μ_1, 3 times more than 3D and almost 5 times more than Rijndael. With α_1, β_1 and μ_1 ACDEL-3D performs 16384 times more than the number of multiplications performed with 3D and 26466 times more with Rijndael.

We assume that the MDS matrices of Rijndael and 3D are randomly chosen for our estimations; in practice, with the MDS matrices of both Rijndael and 3D, this number is less.

V. GENERATION OF THE RANDOM TRANSFORMATIONS

In this section, we present the algorithms for the generation of the random transformations used in ACDEL-2D and ACDEL-3D.

First, the theoretical bases and the complexity analysis for the generation of random invertible matrices in $GL_{8x8}(GF(2))$ can be found in [19]. Such matrices are needed for the transformations RandomSubBytes and RandomAffineKey in their first and second possibilities, for both ACDEL-2D and ACDEL-3D.

In this algorithm, the input is a pseudorandom binary sequence written as matrix

$$\begin{bmatrix} b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} & b_{1,4} & b_{1,5} & b_{1,6} & b_{1,7} \\ c_{2,0} & b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} & b_{2,4} & b_{2,5} & b_{2,6} \\ c_{3,0} & c_{3,1} & b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} & b_{3,4} & b_{3,5} \\ c_{4,0} & c_{4,1} & c_{4,2} & b_{4,0} & b_{4,1} & b_{4,2} & b_{4,3} & b_{4,4} \\ c_{5,0} & c_{5,1} & c_{5,2} & c_{5,3} & b_{5,0} & b_{5,1} & b_{5,2} & b_{5,3} \\ c_{6,0} & c_{6,1} & c_{6,2} & c_{6,3} & c_{6,4} & b_{6,0} & b_{6,1} & b_{6,2} \\ c_{7,0} & c_{7,1} & c_{7,2} & c_{7,3} & c_{7,4} & c_{7,5} & b_{7,0} & b_{7,1} \\ c_{8,0} & c_{8,1} & c_{8,2} & c_{8,3} & c_{8,4} & c_{8,5} & c_{8,6} & b_{8,0} \end{bmatrix}$$

where $b_{k,0}, b_{k,1}, \cdots, b_{k,8-k} \neq 0$ for all $1 \leq k \leq 8$.

In [19], it is also presented the algorithms for the generation of a random invertible matrix and its inverse in $GL_{4x4}(GF(2^8))$, used for the third possibility of RandomAffineKey. Furthermore, the methods presented below for the generation of a random MDS matrix in $GL_{4x4}(GF(2^8))$ use these algorithms.

For the transformations RandomMixLanes and RandomSwapCuboid in ACDEL-3D, as well as RandomMixRows and the second possibility for RandomSubBytes in ACDEL-2D and ACDEL-3D, random permutations is needed to be generated. In [20], the theoretical bases and the complexity analysis for the generation of a random permutation in the symmetric group S_n, $n \geq 2$ can be found.

In this algorithm, the input is a pseudorandom array $(x_1, x_2, \cdots, x_{n-1})$ where $x_i \in \{i, i+1, \cdots, n\}$ for all $1 \leq i \leq n-1$.

A. Generation of random MDS matrices

In this paper, we propose two methods for the generation of MDS matrices in $GL_{4x4}(GF(2^8))$ for RandomMixColumns, whose complexity is similar to that of the algorithms presented in [19] for the generation of random invertible matrices and its inverses in $GL_{4x4}(GF(2^8))$.

The first method for the generation of a random MDS matrix is described in [21], and the following definition of MDS matrix is used:

Any 4x4 matrix over $GF(2^n)$ with all non-zero elements is an MDS matrix if and only if all its square sub-matrices are not singular.

The second method proposed for the generation of a random MDS matrix is a new method using the following proposition found in [22]:

Any 4x4 matrix over $GF(2^n)$ with all non-zero elements is a MDS matrix, if and only if it is full rank, the inverse matrix having all non-zero elements and all its 2x2 sub-matrices are full rank.

Also, it is used the fact, if the polynomial $f(x) = b_{1,0} + b_{1,1}x + b_{1,2}x^2 + b_{1,3}x^3$ on $GF(2^8)$ is such that $b_{1,1}$, $b_{1,2}$ and $b_{1,3} \neq 0$ and $b_{1,0}$ is unknown, then all coefficients of the inverse $f^{-1}(x)$ module $g(x)$, a primitive polynomial of degree 4, depend on $b_{1,0}$.

Input:

- Primitive polynomials $g_1(x)$, $g_2(x)$ and $g_3(x)$ in $GF(2^8)[x]$ selected a priori so that $deg(g_1(x)) = 4$, $deg(g_2(x)) = 3$ and $deg(g_3(x)) = 2$.

- Pseudorandom sequence written as matrix

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

where $c_{i,j}, b_{k,t} \in GF(2^8)$ and $b_{k,0}, \cdots, b_{k,4-k} \neq 0$ for all $2 \leq i \leq 4$, $2 \leq k \leq 4$ and $0 \leq j \leq 2$, $0 \leq t \leq 3$, and also $b_{1,1}$, $b_{1,2}$ and $b_{1,3} \neq 0$.

{

Step 1: Computation of the first row.

The first row of a 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. The value $b_{1,0}$ will be determined in *Step 6* of the present algorithm.

Step 2: Computation of the row $2 \leq i \leq 4$.

From the values of the first row, matrix M and the previous algorithm for random generation of an invertible matrix $A = \{a_{i,j}\}_{4x4}$, $a_{i,j} \in GF(2^8)$, the values $a_{i,j}$ are computed, leaving matrix A in the following way:

$$A = \begin{bmatrix} - & b_{1,1} & b_{1,2} & b_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \\ a_{4,0} & a_{4,1} & a_{4,2} & a_{4,3} \end{bmatrix}$$

- The values $a_{i,j}$ are linear functions of $b_{1,0}$ and then if the values $a_{i,j}$ become equal to zero, linear equations with $b_{1,0}$ as unknown are formed.
- The determinants of all 2x2 sub-matrices are computed. If the determinants become equal to zero, linear and quadratic equations with $b_{1,0}$ as unknown are formed.
- The values of $b_{1,0}$ which do not satisfy the mentioned equations are stored.

Step 3: With the coefficients $b_{1,1}$, $b_{1,2}$ and $b_{1,3}$ of matrix M and $b_{1,0}$ as unknown, it is computed the coefficients $d_{1,0} = \lambda_0(b_{1,0})$, $d_{1,1} = \lambda_1(b_{1,0})$, $d_{1,2} = \lambda_2(b_{1,0})$ and $d_{1,3} = \lambda_3(b_{1,0})$ of the inverse $f^{-1}(x) = d_{1,0} + d_{1,1}x + d_{1,2}x^2 + d_{1,3}x^3$ module $g_1(x)$. Thus, we can form the matrix

$$M' = \begin{bmatrix} d_{1,0} & d_{1,1} & d_{1,2} & d_{1,3} \\ c_{2,0} & b_{2,0} & b_{2,1} & b_{2,2} \\ c_{3,0} & c_{3,1} & b_{3,0} & b_{3,1} \\ c_{4,0} & c_{4,1} & c_{4,2} & b_{4,0} \end{bmatrix}$$

Step 4: Using matrix M' and the algorithm described above to compute the inverse, we can compute

$$A^{-1} = \begin{bmatrix} d_{1,0} & d_{1,1} & d_{1,2} & d_{1,3} \\ d_{2,0} & d_{2,1} & d_{2,2} & d_{2,3} \\ d_{3,0} & d_{3,1} & d_{3,2} & d_{3,3} \\ d_{4,0} & d_{4,1} & d_{4,2} & d_{4,3} \end{bmatrix}$$

Note that $d_{i,j}$ depend on $b_{1,0}$, $b_{1,1}$, $b_{1,2}$ and $b_{1,3}$ for all $2 \leq i \leq 4$ and $0 \leq j \leq 3$, then matrix A^{-1} becomes

$$\begin{bmatrix} \delta_{1,0}(b_{1,0}) & \delta_{1,1}(b_{1,0}) & \delta_{1,2}(b_{1,0}) & \delta_{1,3}(b_{1,0}) \\ \delta_{2,0}(b_{1,0}) & \delta_{2,1}(b_{1,0}) & \delta_{2,2}(b_{1,0}) & \delta_{2,3}(b_{1,0}) \\ \delta_{3,0}(b_{1,0}) & \delta_{3,1}(b_{1,0}) & \delta_{3,2}(b_{1,0}) & \delta_{3,3}(b_{1,0}) \\ \delta_{4,0}(b_{1,0}) & \delta_{4,1}(b_{1,0}) & \delta_{4,2}(b_{1,0}) & \delta_{4,3}(b_{1,0}) \end{bmatrix}$$

Step 5: If the values of $\delta_{i,j}(b_{1,0})$ for all $1 \leq i \leq 4$ and $0 \leq j \leq 3$ become equal to zero, equations with $b_{1,0}$ as unknown are formed. The values of $b_{1,0}$ which do not satisfy the mentioned equations are stored.

Step 6: Random generation of the MDS matrix *A*.

From the values of $\boldsymbol{b_{1,0}}$ which do not satisfy the equations of *Step 2* and *Step 5*, one should be selected at random, leaving matrix $\boldsymbol{M}$ full, and then matrix $\boldsymbol{A}$ and its inverse are completed.

}

Output:

MDS matrix $\boldsymbol{A} = \begin{bmatrix} \boldsymbol{b_{1,0}} & \boldsymbol{b_{1,1}} & \boldsymbol{b_{1,2}} & \boldsymbol{b_{1,3}} \\ \boldsymbol{a_{2,0}} & \boldsymbol{a_{2,1}} & \boldsymbol{a_{2,2}} & \boldsymbol{a_{2,3}} \\ \boldsymbol{a_{3,0}} & \boldsymbol{a_{3,1}} & \boldsymbol{a_{3,2}} & \boldsymbol{a_{3,3}} \\ \boldsymbol{a_{4,0}} & \boldsymbol{a_{4,1}} & \boldsymbol{a_{4,2}} & \boldsymbol{a_{4,3}} \end{bmatrix}$

VI. FURTHER COMMENTS ON DESIGN STRATEGY AND SECURITY

Rijndael was designed by means of the Wide Trail Strategy (WTS), so that full diffusion is achieved in the minimal number of rounds as possible.

In the WTS, the diffusion step is constructed in such a way that one transformation acts on the columns of the state, providing high local diffusion; and another transformation acts on the rows of the state, providing high dispersion. This way the influence of one bit spreads to all the state as soon as possible.

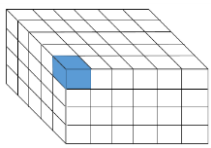
Considering this, MixColumns was chosen as an MDS matrix and ShiftRows as a diffusion optimal permutation, achieving full diffusion in three rounds (two rounds if $N_b = 4$) in the sense that every state bit depends on all state bits in the previous three rounds, or a change in one state bit is likely to affect half of the state bits after three rounds. Other criteria such as efficiency, were also taken into account by Rijndael's designers [3].

On the other hand, 3D is also based on the WTS, using in alternate rounds two diffusion optimal permutations on the slices and the sheets of the state cube. This way, 3D achieves full diffusion in three rounds, encrypting twice as much data as Rijndael in one execution.

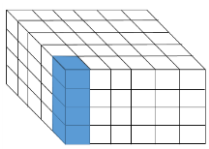
ACDEL-2D and ACDEL-3D allow to encrypt long message blocks at once, while Rijndael encrypts at most 256 bits at once and 3D encrypts always 512 bits at once. This way, to encrypt a message block larger than 512 bits, with Rijndael and 3D, more than one execution is needed as well as a block cipher mode of operation to make possible that each output bit depends on all input bits. Meanwhile, with ACDEL-2D and ACDEL-3D, each output bit depends on all input bits in a single execution.

Of course, some constructions like Rijndael and 3D can be built if longer message blocks need to be encrypted, increasing the values of the number of columns and the number of slices in case of 3D without replacing the original transformations. In this case, full diffusion is achieved in a bigger number of rounds.

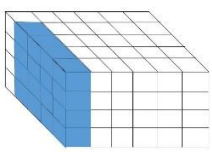
By rule, if $4^{i-1} < N_b \leq 4^i$, where $i \geq 1$, using a kind of construction like Rijndael full diffusion is achieved in $i + 1$ rounds to encrypt a $4N_b$-byte message block.



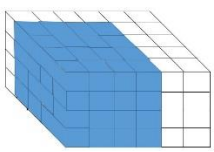
Round 1: $\theta_1(\pi(\gamma(k_1(\text{State}, \text{ExpandedKey}[1]))))$



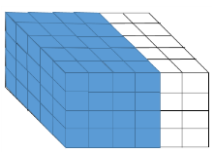
Round 2: $\theta_2(\pi(\gamma(k_2(\text{State}, \text{ExpandedKey}[2]))))$



Round 3: $\theta_1(\pi(\gamma(k_3(\text{State}, \text{ExpandedKey}[3]))))$



Round 4: $\theta_2(\pi(\gamma(k_4(\text{State}, \text{ExpandedKey}[4]))))$



Round 5: $\theta_1(\pi(\gamma(k_5(\text{State}, \text{ExpandedKey}[5]))))$

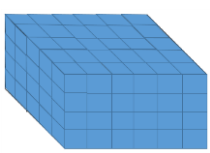


Fig. 3. Spreading of one-byte dependency through 5 rounds for the state shown in Fig. 1.

Similarly, if also $4^{j-1} < N_h \leq 4^j$, where $j \geq 1$, using a kind of construction like 3D full diffusion is achieved in $i + j + 1$ rounds if $i \geq j$ or $2(j + 1)$ rounds if $i < j$ to encrypt a $4N_bN_h$-byte message block.

As we said before, ShiftRows in Rijndael and θ_1 and θ_2 in 3D act on the state spreading the diffusion provided by the MDS matrix to every column. In general, all diffusion optimal permutations have column branch number equal to the branch number of MixColumns; however, MixRows has column branch number $N_b + 1$ and MixLanes has column branch number $N_h + 1$.

This way, our constructions allow to encrypt long message blocks at once and full diffusion is always achieved in two rounds for two-dimensional states or three rounds for three-dimensional states. This constitutes the main advantage of ACDEL-2D and ACDEL-3D over Rijndael, 3D, and other possible constructions.

Until today, none of the known attacks pose a threat to Rijndael in a practical sense. In addition, any cryptanalytic attack seems to be effective in reduced versions of Rijndael with more than six rounds [3]; however, some extra rounds were added in Rijndael in such a way that at least 2 full diffusion steps act as security margin.

In ACDEL-2D, we add 4 full diffusion steps as security margin after 6 rounds, for a final number of rounds $N_r = 14$. In ACDEL-3D, were added 4 full diffusion steps as security margin after 9 rounds, for a final number of rounds $N_r = 21$. More rounds could be added in both cases as security margin; however, the number of multiplications would be even higher and this amount is sufficient to prevent existing attacks.

Most of the security of our constructions relies on the WTS and the number of rounds previously defined. On the other hand, extra security is provided in both ACDEL-2D and ACDEL-3D by the unknowingness of the random transformations used in the encryption process. We consider that ACDEL-2D and ACDEL-3D are strong ciphers.

VII. Conclusion

In this paper, the dynamic cryptographic algorithms ACDEL-2D and ACDEL-3D, working on two-dimensional states and three-dimensional states respectively have been presented. In both ACDEL-2D and ACDEL-3D, all transformations are randomly generated depending on pseudorandom sequences, obtained directly from the Rijndael's key schedule or any other pseudorandom number generator.

Our constructions are based in the WTS of Rijndael, allowing to encrypt long message blocks at once without using a block cipher mode of operation, at the same time the minimal number of rounds to achieve full diffusion is always two for ACDEL-2D or three for ACDEL-3D.

In both proposals, a random MDS matrix is used alternately on columns and rows of the state; instead of Rijndael and 3D, which use a fixed MDS matrix on columns and a fixed diffusion optimal permutation on rows. Since the matrix multiplication is a very expensive operation, the number of multiplications performed in ACDEL-2D and ACDEL-3D is higher as the states are longer; however, both constructions have many applications in practice.

VII. Appendix

A. Inverse of one polynomial of degree 3

Next, we present an example of the inverse of one polynomial in $GF(2^8)[x]$ with degree 3, so that it can be used in the generation of a random MDS matrix through the new method described in this paper.

Let $f(x)$ be the polynomial

$$z^8x^3 + z^4x^2 + z^{32}x + z^2$$

and let $g(x)$ be the primitive polynomial

$$x^4 + z^9x^3 + z^2x + z^{13}$$

in $GF(2^8)[x]$, where $z \in GF(2^8)$ is a primitive element modulo the irreducible polynomial used to construct the field $GF(2^8)$ in Rijndael

$$P(y) = y^8 + y^4 + y^3 + y + 1$$

Let $f^{-1}(x) = d_{1,0} + d_{1,1}x + d_{1,2}x^2 + d_{1,3}x^3$ be the inverse of $f(x)$ modulo$g(x)$, so that $d_{1,0} = \lambda_0(b_{1,0})$, $d_{1,1} = \lambda_1(b_{1,0})$, $d_{1,2} = \lambda_2(b_{1,0})$, $d_{1,3} = \lambda_3(b_{1,0})$. If we take $z = y + 1$ then

$$d_{1,3} \equiv z^{89} \bmod(P(y))$$

$$d_{1,2} \equiv z^{64} \bmod(P(y))$$

$$d_{1,1} \equiv z^{46} \bmod(P(y))$$

$$d_{1,0} \equiv z^{119} \bmod(P(y))$$

REFERENCES

[1] Daemen J. and Rijmen V. The Rijndael block cipher. AES proposal. 1999. http/www.daimi.ai.dk/~iran/rijndael.pdf. Accessed on Dec 11, 2020.

[2] Federal Information Processing Standard. Announcing the Advanced Encryption Standard (AES). FIPS Publication 197, 2001.

[3] Daemen J. and Rijmen V. "The design of Rijndael: AES - The Advanced Encryption Standard". Second Edition. Springer. 2020.

[4] Nakahara J. and Abrahao E. "A New Involutory MDS Matrix for the AES." IJ Network Security 9 (2), 2009.

[5] Elumalai R. and Raji A. "Improving diffusion power of AES Rijndael with 8x8 MDS matrix." International Journal of Scientific & Engineering Research 2 (3), 2011.

[6] Liu Z. and De H. "Dynamic Encryption Algorithm Based on Rijndael." Advanced Materials Research. Vol. 490. Trans Tech Publications Ltd, 2012.

[7] Craig Suzanne. "A Simplified AES with Field Characteristic 7." Proceedings of the NCUR. 2014.

[8] Gowda S., Aravind H. and Usha S. "Design and ASIC Implementation of Modified Rijndael Cipher." (IRJET) International Research Journal of Engineering and Technology. 2016.

[9] Yang M., Xiao B. and Meng Q. "New AES Dual Ciphers Based on Rotation of Columns." Wuhan University Journal of Natural Sciences 24 (2), 2019.

[10] Bossert J, et al. "Pholkos - Efficient Large-state Tweakable Block Ciphers from the AES Round Function." IACR ePrint Archive, Vol. 275, 2020.

[11] Nakahara J. "3D: A three-dimensional block cipher." International Conference on Cryptology and Network Security. Springer, Berlin, Heidelberg, 2008.

[12] Ariffin S., Mahmod R. and Jaafar A.. "Immune systems approaches for cryptographic algorithm." Sixth International Conference on Bio-Inspired Computing: Theories and Applications. IEEE, 2011.

[13] Jamel S., et al. "The hybrid cubes encryption algorithm (HiSea)." Advances in Wireless, Mobile Networks and Applications. Springer, 2011.

[14] Chugunkov I., et al. "Three - dimensional data stochastic transformation algorithms for hybrid supercomputer implementation." 17th Mediterranean Electrotechnical Conference. IEEE, 2014.

[15] Federal Information Processing Standard. "SHA-3 standard: Permutation-based hash and extendable-output functions." FIPS Publication 202, 2015.

[16] Wang Q. and Jin C. "A non-alternate 3D structure and its practical security evaluation against differential and linear cryptanalysis." Science China Information Sciences 61 (5), 2018.

[17] Rijmen V., Daemen J., Preneel B., Bosselaers A. and De Win E. "The cipher SHARK". LNCS 1039, pp. 99–111. Springer, 1996.

[18] Federal Agency on Technical Regulation and Metrology. "National Standard of the Russian Federation GOST R34.12-2015". 2015.

[19] Freyre P, Díaz N and Morgado E. R. "Some algorithms related to matrices with entries in a finite field". Journal of Discrete Mathematical Sciences & Cryptography. India. Vol. 12, No. 5, pp. 509–519. 2009.

[20] Freyre P and Díaz N. "Generación aleatoria de permutaciones del grupo simétrico o del grupo alternado". Revista Investigación Operacional. Vol. 36, No. 2, 2015.

[21] Freyre P, Díaz N, Díaz R and Pérez C. "Random generation of MDS matrices". Proceedings of Current Trends in Cryptology CTCrypt2014. Russia, 2014.

[22] 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.

ABOUT THE AUTHOR

Pablo Freyre Arrozarena

Workplace: Institute of Cryptography. University of Havana.

Email: pfreyre@matcom.uh.cu

Education: Graduated in Mathematics in 1988; received Doctor's degree in 1998.

Current research direction: symmetric cryptography, mathematical aspects of cryptography and information security.

Nelson Díaz Pérez

Workplace: Institute of Cryptography. University of Havana.

Education: Graduated in Mathematics in 1985; received Master's degree in 2006.

Current research direction: symmetric cryptography, mathematical aspects of cryptography and information security.

Oristela Cuellar Justiz

Workplace: Center for the Study of Computational Mathematics. University of Informatics Sciences.

Email: oristelacj@uci.cu

Education: Graduated in Mathematics and Physics in 1987; received Doctor's degree in 2016.

Current research direction: symmetric cryptography, mathematical aspects of cryptography and information security.

Adrián Alfonso Peñate

Workplace: Institute of Cryptography. University of Havana.

Education: Graduated in Mathematics in 2014; received Master's degree in 2018.

Current research direction: symmetric cryptography, mathematical aspects of cryptography and information security.