Dynamic Cryptographic Algorithms Kuznyechik and Magma

Pablo Freyre, Oristela Cuellar, Nelson Díaz, Ramses Rodríguez and Adrián Alfonso

***Abstract*—The cryptographic algorithms Kuznyechik and Magma since 2015 are block cipher standardized in the Russian Federation, formally called GOST R 34.12-2015. Both use fixed functions as a priori selected and differ on the structure, the block length and the bit-level of the processed blocks. In the present paper, we provide a dynamic variant of Kuznyechik and Magma where some of their functions are randomly generated and dependent on pseudorandom sequences.**

***Tóm tắt*—Các thuật toán mã hóa Kuznyechik và MAGMA từ năm 2015 là mật mã khối được tiêu chuẩn hóa ở Liên bang Nga, được gọi chính thức là GOST R 34.12-2015. Hai thuật toán này đều sử dụng các hàm chức năng được lựa chọn ưu tiên và khác nhau về cấu trúc, độ dài khối và mức bit của các khối được xử lý. Trong bài báo này, nhóm tác giả cung cấp một biến thể động của Kuznyechik và MAGMA, trong đó một số hàm chức năng của chúng được tạo ngẫu nhiên và phụ thuộc vào chuỗi giả ngẫu nhiên.**

Keywords—Block cipher; Kuznyechik; Magma; random permutation; MDS matrix.

Từ khóa—Mã khối; Kuznyechik; Magma; hoán vị ngẫu nhiên; ma trận MDS.

I. INTRODUCTION

The aim of this paper is to present a dynamic variant of the cryptographic algorithms Kuznyechik and Magma (GOST R 34.12-2015) [1], [2], actually included in the block cipher standards of the Russian Federation.

Both algorithms are constructed by relying on strong cryptographic primitives, allowing to achieve good confusion and diffusion, however the internal functions used in the encryption process are fixed and selected a priori.

This manuscript is received on May 22, 2020. It is commented on June 16, 2020 and is accepted on July 17, 2020 by the first reviewer. It is commented on August 3, 2020 and is accepted on September 3, 2020 by the second reviewer.

In the case of Kuznyechik the MDS matrix is defined in $GL_{16}(GF(2^8))$ and we propose replacing the same one by another randomly generated in terms of a pseudorandom sequence as will be explained next. In the case of Magma, the cyclic shift 11 bits to the left is replaced by a Cauchy matrix in $GL_8(GF(2^4))$ or an invertible matrix having all non-zero elements as well as its inverse in $GL_8(GF(2^4))$, generated in terms of pseudorandom sequences that will be explained next.

The present paper begins with a description of the Kuznyechik and Magma block ciphers. We then present aspects on the functions used in both dynamic variants and finish with the algorithms for the random generation of the random functions used in dynamic Kuznyechik and dynamic Magma. We discuss some security aspects on both dynamic variants.

II. BLOCK CIPHER KUZNYECHIK

The cryptographic algorithm of the Russian Federation Kuznyechik (GOST R 34.12-2015) is a block cipher structured as Substitution-Permutation network and the input/output blocks have 128 bits of length (16 byte-words).

The secret key has 256 bits (32 byte-words) and the key schedule is a Feistel network of 8 rounds, meanwhile 10 rounds XSL are performed for the encryption/decryption process. Kuznyechik uses a S-box in S_{256} and a MDS matrix in $GL_{16}\big(GF(2^8)\big)$, both are fixed and selected a priori.

Now we present the encryption process of the algorithm Kuznyechik, where State means the one-dimensional array $p_{16}p_{15}\cdots p_1$ formed from the input block $p_1p_2\cdots p_{16}$ and transformed through the encryption process. The same operation must be presented to obtain the output block from the final State. More details of Kuznyechik are presented in [1], [2], [3].

Encryption process of Kuznyechik

```
Kuznyechik(State, CipherKey)
{
    KeyExpansion(CipherKey, ExpandedKey);
    For (i = 0; i < 10; i++)
    {
        AddRoundKey(State, ExpandedKey[i]);
        SubBytes(State);
        MixRows(State);
    }
    AddRoundKey(State, ExpandedKey[10]);
}
```

Where SubBytes and AddRoundKey are the S-box and the XOR addition with the round key, and MixRows is the MDS matrix of Kuznyechik u_t^{16} in $GL_{16}\left(GF(2^8)\right)$, where u_t is the accompanist matrix of the nextpolynomial $t(x)$ in $GF(2^8)$[x]

$$t(x) = (x \oplus 6)(x \oplus 12)(x \oplus 24)(x \oplus 48)(x \oplus 70)(x \oplus 75)(x \oplus 95)(x \oplus 103)(x \oplus 117)(x \oplus 140)(x \oplus 150)(x \oplus 196)(x \oplus 206)(x \oplus 210)(x \oplus 219)(x \oplus 239).$$

III. Block cipher Magma

The cryptographic algorithm of the Russian Federation Magma (GOST R 34.12-2015) is a block cipher structured as Feistel network, and the input/output blocks have 64 bits of length (16 nibble-words).

The secret key has 256 bits (8 words of 32 bits) and the key schedule is a simple repetition of these words, meanwhile 32 rounds are performed for the encryption/decryption process. Magma uses 8 different S-boxes in S_{16} and a cyclic shift 11 bits to the left.

Magma is a modification of the cryptographic algorithm GOST 28147-89, standardized first in the Russian Federation to protect secret and military information, which only differs in the choice of the S-boxes. In the first version, the S-boxes are secret and these could be selected at random, while in the second version these are public and fixed.

In the following, we present the encryption process of the algorithm Magma, where L_0 means the left half and R_0 means the right half of the input block, both are transformed through the encryption process until obtain at the output of the final round the output block $L_{32}||R_{32}$. More details of Magma are presented in [1], [2], [4].

Encryption process of Magma

```
Magma((L0, R0), CipherKey)
{
    KeyExpansion(Cipher, ExpandedKey);
    For (i = 0; i < 32; i++)
    {
        Ri = Li−1 ⊕ fki(Ri−1);
        Li = Ri−1;
    }
}
```

Where

$$f_{k_i}(R_{i-1}) = Rot_{11}\text{SubBytes}\ Add_{k_i}(R_{i-1})$$

is the round transformation of Magma, composed by the functions: Add_{k_i}- the addition modulo 2^{32} with the round key, SubBytes - the 8 S-boxes S_1 ... S_8 defined above, and Rot_{11} - the cyclic shift 11 bits to the left.

IV. Dynamic Kuznyechik

The encryption process in the dynamic algorithm Kuznyechik is similar to the original Kuznyechik, only differs in the sense that the function MixRows is preceded by the random functions PermRows, and MultRows, as we show next.

Encryption process of Dynamic Kuznyechik

```
DynamicKuznyechik(State, CipherKey)
{
    RandomPermRows(sequence1, PermRows);
    RandomMultRows(sequence2, MultRows);
    KeyExpansion(Cipher, ExpandedKey);
    For (i = 0; i < 10; i++)
    {
```

```
        AddRoundKey(State, ExpandedKey[i]);
        SubBytes(State);
        PermRows(State);
        MultRows(State);
        MixRows(State);
    }
    AddRoundKey(State, ExpandedKey[10]);
}
```

Here MixRows and AddRoundKey are the same transformations of the algorithm Kuznyechik, and SubBytes instead of the S-box of Kuznyechik can be one of the S-boxes obtained in [5].

PermRows is a random permutation $\pi \in S_{16}$ selected from sequence$_1$, a pseudorandom sequence generated from the Kuznyechik's key schedule or directly from any external pseudorandom number generator, such that the State is transformed as

$$(p_{16}, p_{15}, \cdots, p_1) \to (p_{\pi[1]}, p_{\pi[2]}, \cdots, p_{\pi[16]}).$$

MultRows is a random vector $(d_1, d_2, \ldots, d_{16})$ where $d_i \in GF(2^8)^*$ for all $1 \leq i \leq 16$, selected from sequence$_2$, a pseudorandom sequence generated from the Kuznyechik's key schedule or directly from any external pseudorandom number generator, such that the State is transformed as

$$(p_{16}, p_{15}, \cdots, p_1) \to (d_{16}p_{16}, d_{15}p_{15}, \cdots, d_1p_1).$$

V. Dynamic Magma

The encryption process in the dynamic algorithm Magma is similar to the original Magma, only differs in the sense that the cyclic shift 11 bits to the left is replaced by the random function MixRows, a Cauchy matrix or an invertible matrix with all non-zero elements as well as its inverse.

Encryption process of Dynamic Magma

```
DynamicMagma((L_0, R_0), CipherKey)
{
    KeyExpansion(Cipher, ExpandedKey);
    RandomMixRows(sequence_3, MixRows);
    For (i = 0; i < 32; i++)
    {
        R_i = L_{i-1} ⊕ f_{k_i}(R_{i-1});
        L_i = R_{i-1};
    }
}
```

Here SubBytes and Add_{k_i} are the same transformations of the algorithm Magma, although we strongly recommend the change of the actual key schedule of this standard.

MixRows is a random matrix selected from sequence$_3$, a pseudorandom sequence generated from the Magma's key schedule or directly from any external pseudorandom number generator, in such way that it can be:

1. A Cauchy matrix [6].
2. An invertible matrix with all non-zero elements as well as its inverse.

Thus the round transformation of the dynamic variant of Magma becomes in

$$f_{k_i}(R_{i-1}) = \text{MixRows SubBytes Add}_{k_i}(R_{i-1}).$$

VI. Random generation of the dynamic functions

Now we will present an algorithm that randomly generate a permutation of the symmetric group S_{16}. The theoretical bases and the complexity analysis of this algorithm can be seen in [7] for the general case of S_n, where $n > 1$ is any natural number.

The resultant permutations of this algorithm are used in the transformations PermRows for the case of the dynamic variant of Kuznyechic and MixRows for the case of the dynamic variant of Magma, in the way of a Cauchy matrix.

Input:

Pseudorandom sequence $(\gamma_1, \gamma_2, \cdots, \gamma_{16})$ where $\gamma_i \in \{i, i+1, \cdots, 16\}$ for all $1 \leq i \leq 16$.

```
{
    γ_16 = 16;
    For (j=1; j < 16; j++)
    {
        π[j] = j;
```

For (i=j; i > 0; i--)

{

$\pi[j] = (\pi[j] + \gamma_i - i)$;

If $\pi[j] > 16$ then $\pi[j] = (\pi[j] + i - 1) \bmod 16$;

}

}

}

Output:

Permutation $\pi = \begin{pmatrix} 1 & 2 & \cdots & 16 \\ \pi[1] & \pi[2] & \cdots & \pi[16] \end{pmatrix}$

A. Generation of a random Cauchy matrix

We present in this section an algorithm for the random generation of a Cauchy matrix for the dynamic variant of Magma, where it is assumed that $\pi[i] \in GF(2^4)$ for all $1 \leq i \leq 16$.

Input:

Pseudorandom permutation $\pi \in S_{16}$.

{

For (j = 1; j < 9; j++)

{

For (i = 1; i < 9; i ++)

$c_{i,j} = (\pi[i] + \pi[j + 8])^{-1}$;

}

}

Output:

Cauchy matrix $C = c_{i,j}$.

B. Generation of a random invertible matrix

We present in this section an algorithm for the random generation of an invertible matrix with all non-zero elements as well as its inverse, for the case of the dynamic variant of Magma. The theoretical bases and complexity analysis of this algorithm is similar to the algorithm described in [8] for the generation of a random MDS matrix.

First, we introduce an algorithm for the random generation of an invertible matrix $A = (a_{i,j})$ of size 4, where $a_{i,j} \in GF(2^8)$ for all $1 \leq i, j \leq 4$, and another for the random generation of A^{-1}, which will be used lately.

Computation of an invertible matrix:

Input:

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

- Pseudorandom sequence is written as the matrix

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

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

{

Step 1: Computation of the first row

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

$$\hat{a}_0 + \hat{a}_1 x + \hat{a}_2 x^2 + \hat{a}_3 x^3 = (a_0 + a_1 x + a_2 x^2 + a_3 x^3)(b_{1,0} + b_{1,1} x + b_{1,2} x^2 + b_{1,3} x^3) \bmod g_1(x)$$

$(a_0, a_1, a_2, a_3) = (\hat{a}_0, \hat{a}_1, \hat{a}_2, \hat{a}_3)$

Output: $row_1 = (a_0, a_1, a_2, a_3)$

Step 2: Computation of the j - th row, $2 \leq j \leq 4$

Input: (a_0, a_1, a_2, a_3) the j - th canonical vector

For $(i = j;\ i > 1;\ i\text{--})$

{

$\hat{a}_0 = a_0 + c_{i,0} a_{i-1}$

$\hat{a}_1 = a_1 + c_{i,1} a_{i-1}$

$\vdots$

$\hat{a}_{i-2} = a_{i-2} + c_{i,i-2} a_{i-1}$

$$\hat{a}_{i-1} + \hat{a}_i x + \cdots + \hat{a}_3 x^{4-i} = (a_{i-1} + a_i x + \cdots + a_3 x^{4-i})(b_{i,0} + b_{i,1} x + \cdots + b_{i,4-i} x^{4-i}) \bmod g_i(x)$$

$(a_0, a_1, a_2, a_3) = (\hat{a}_0, \hat{a}_1, \hat{a}_2, \hat{a}_3)$

}

Output: $row_j = (a_0, a_1, a_2, a_3)$

}

Output:

$$\text{Matrix } A = \begin{pmatrix} row_1 \\ row_2 \\ row_3 \\ row_4 \end{pmatrix}$$

Computation of an inverse matrix:

Input:

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

- Pseudorandom sequence is written as the matrix

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

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

{

Computation of the j – th row, $1 \leq j \leq 4$.

Step 1:

Input: (a_0, a_1, a_2, a_3) the j – th canonical vector

$$\hat{a}_0 + \hat{a}_1 x + \hat{a}_2 x^2 + \hat{a}_3 x^3 = (a_0 + a_1 x + a_2 x^2 + a_3 x^3)\left(b_{1,0} + b_{1,1} x + b_{1,2} x^2 + b_{1,3} x^3\right)^{-1} mod\ g_1(x)$$

$$(a_0, a_1, a_2, a_3) = (\hat{a}_0, \hat{a}_1, \hat{a}_2, \hat{a}_3)$$

Output: (a_0, a_1, a_2, a_3)

Step 2:

Input: (a_0, a_1, a_2, a_3)

For (i = 2; i≤ 4; i++)

{

$$\hat{a}_{i-1} + \hat{a}_i x + \cdots + \hat{a}_3 x^{4-i} = (a_{i-1} + a_i x + \cdots + a_3 x^{4-i})\left(b_{i,0} + b_{i,1} x + \cdots + b_{i,4-i} x^{4-i}\right)^{-1} mod\ g_i(x)$$

$$\hat{a}_0 = a_0 + c_{i,0} a_{i-1}$$

$$\hat{a}_1 = a_1 + c_{i,1} a_{i-1}$$

$$\vdots$$

$$\hat{a}_{i-2} = a_{i-2} + c_{i,i-2} a_{i-1}$$

$$(a_0, a_1, a_2, a_3) = (\hat{a}_0, \hat{a}_1, \hat{a}_2, \hat{a}_3)$$

}

Output: $row_j = (a_0, a_1, a_2, a_3)$

}

Output:

$$\text{Matrix } A^{-1} = \begin{pmatrix} row_1 \\ row_2 \\ row_3 \\ row_4 \end{pmatrix}$$

Now we are able to present the algorithm for the computation of an invertible matrix with all non-zero elements as well as its inverse.

In this algorithm, it is used the fact if $f(x) = b_{1,0} + b_{1,1} x + b_{1,2} x^2 + b_{1,3} x^3$ 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 the inverse $f^{-1}(x)$ modulo $g(x)$ a primitive polynomial of degree 4 have all their coefficients depending on $b_{1,0}$.

Input:

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

- Pseudorandom sequence is written as the 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 $b_{k,0}, b_{k,1}, \cdots, b_{k,4-k} \neq 0$ 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 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 i - th row, $2 \leq i \leq 4$

From the values of the first row, the matrix M and the previous algorithm for random generation of an invertible matrix $A = \{a_{i,j}\}_{4x4}$, $a_{i,j} \in GF(2^8)$ for all i and j, the values $a_{i,j}$ are calculated, leaving the 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 function 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 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 the 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 $f^{-1}(x)\ mod\ g_1(x) = \left(d_{1,0} + d_{1,1}x + d_{1,2}x^2 + d_{1,3}x^3\right) mod\ g_1(x)$. 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: With the matrix M' and the algorithm described above used to compute the inverse, we can compute the inverse matrix

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

Note that $d_{i,\mathrm{j}}$ is function of $d_{1,0}$, $d_{1,1}$, $d_{1,2}$ and $d_{1,3}$ for all $2 \leq i \leq 4$ and $0 \leq j \leq 3$, so all these coefficients are functions of $b_{1,0}$, and then the matrix A^{-1} is

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

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 A

From the values of $b_{1,0}$ which do not satisfy the equations of steps 2 and 5, one should be selected at random leaving matrix M full, and then it is obtained the matrix A and their inverse with the algorithms described above.

}

Output:

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

VII. SECURITY COMMENTS

Most of the cryptanalytic methods exploit the properties of the internal functions in the encryption and decryption process, for example differential and linear attacks [9 and 10]. We introduce some random functions in a big set of possibilities, making these cryptanalytic methods more difficult to implement since the linear transformations used in both cases are key dependent.

In this sense [11] shows how the different cryptanalysis can be used against the block cipher Kuznyechic, which is not possible in our dynamic variant, taking into account that the encryption process is used for short plain texts or in the contrary must be used one rekeyed mode of operation for big plain texts [12].

On the other hand, an efficient algorithm is proposed in [13] to recover the secret S-box of the previous version of Magma, GOST 28147-89, under special conditions given in the way in which they are used and taken advantage on the cyclic shifts. Assuming that this idea can be extended to recover secret and key dependent S-boxes in Magma; in our dynamic variant we present other scenario, the cyclic shift step is replaced by a random and secret matrix, making more difficult to know the same one due to the non-linear property of the S-boxes.

VIII. CONCLUSION

In the present paper, a dynamic variation of the cryptographic algorithms Kuznyechik and Magma has been described, which enables that some of its functions become random in terms of the secret key or pseudorandom sequences.

For the dynamic variant of the block cipher Kuznyechik, the function MixRows is randomized by the functions PermRows and MultRows, such that the action of the MDS matrix on the State is equivalent to the Kuznyechik's matrix has permuted its rows and each one has been multiplied by d_i, where $d_i \in GF(2^8)^*$ for all $1 \leq i \leq 16$.

We also propose an improvement of Kuznyechic using one of the S-boxes presented in [5] due to their stronger cryptographic properties.

On the other hand, our dynamic variant of the block cipher Magma differs from the original Magma and its previous variants GOST 28147-89 and 2GOST [14], that the cyclic shift 11 bits to the left is replaced by a random Cauchy matrix or a random invertible matrix with all non-zero elements as well as its inverse. Although the presented scheme guarantees an increase in security, we also recommend a variation in the key schedule since it offers a low security [15].

We note that, in the algorithm described above for the generation of Cauchy matrices, if $(x_0, \dots, x_7)(y_0, \dots, y_7)$ satisfy the conditions to construct a Cauchy matrix $C_{mxm} \in GL_8\big(GF(2^4)\big)$, then $(x_0 + a, \dots, x_7 + a)(y_0 + a, \dots, y_7 + a)$ generate the same Cauchy C_{mxm} matrix, where x_i and $y_i \in \mathrm{GF}(2^4)$ for all $i \in \{0, \dots, 7\}$ and $a \in GF(2^4)^*$.

For both dynamic variants of Kuznyechik and Magma respectively, it must be taken into account that the encryption process is used for short plain texts or in the contrary must be used a rekeyed mode of operation for big plain texts [12].

REFERENCES

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

[2] Dygin, D. et al (2015). "On a new Russian Encryption Standard." Matematicheskie Voprosy Kriptografii 6(2): 29-34.

[3] Ishchukova, E., L. Babenko and M. Anikeev (2017). "Two simplified versions of Kuznyechik cipher (GOST R 34.12-2015)". 10th International Conference on Security of Information and Networks, ACM.

[4] Ishchukova, E., L. Babenko and M. Anikeev (2016). "Fast Implementation and Cryptanalysis of GOST R 34.12-2015 Block Ciphers". 9th International Conference on Security of Information and Networks, ACM.

[5] de la Cruz, R.A. (2019). "Generation of 8-Bit S-Boxes Having Almost Optimal Cryptographic Properties Using Smaller 4-Bit S-Boxes and Finite Field Multiplication". Progress in Cryptology – LATINCRYPT2017. LNCS, 11368, Springer, Cham.

[6] Zhuojun L. Qiuping L. and Baofeng W. (2018). "Direct Construction of (involutory) MDS matrix from Block Vandermonde and Cauchy-like Matrix". 7th International Workshop, WAIFI 2018, Bergen, Norway.

[7] Freyre, P. and Díaz, N. (2015). "Generación aleatoria de permutaciones del grupo simétrico o del grupo alternado." Revista Investigación Operacional. 36(2): 186-191.

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

[9] Biham E. and Shamir A. (1991). "Differential Cryptanalysis of DES-like Cryptosystems". Journal of Cryptology vol. 4, no. 1, pp 3-72.

[10] Matsui M. (1993). "Lineal Cryptanalysis method for DES Cipher". Advances in Cryptology-Eurocrypt93. Springer-Verlag, pp 386-397.

[11] Marshalko V, Bondarenko I, Agafonoba V (2019). "Abstract of the results of the analysis of Kuznyechik". RusCrypto. Russia.

[12] Akhmetzyanova, L., et al (2017). "On the properties of the CTR encryption mode of Magma and Kuznyechik block ciphers with re-keying method based on CryptoPro Key Meshing". Matematicheskie Voprosy Kriptografii. 8(2).

[13] Markku-Juhani O. Saariment. (2019). "A chosen key attacks against the secret S-box of GOST". IACR 540.

[14] Dmukh, A., Dygin, D. and Marshalko, G. (2015). "A lightweight-friendly modification of GOST block cipher", IACR Cryptology ePrint Archive, 065.

[15] Nicolas C. (2011). "Security Evaluation of GOST 28147-89 in view of international standardization". IACR-211.

ABOUT THE AUTHORS

PhD. Pablo Freyre Arrozarena

Workplace: Institute of Cryptography. University of Havana.

Email: pfreyre@matcom.uh.cu

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

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

PhD. Oristela Cuellar Justiz

Workplace: Center for the Study of Computational Mathematics. University of Informatics Scien-ces. Email: oristelacj@uci.cu

The education process: Graduated of Mathematics and Physics in 1987; received Doctor's degree in 2016.

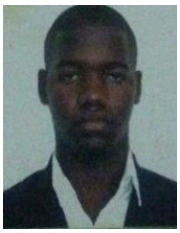
Current research direction: symmetric cryptography, mathematical aspects of cryptography and information security.

MSc. Nelson Díaz Pérez

Workplace: Institute of Cryptography. University of Havana.

The education process: Graduated of Mathematics in 1985; received Master's degree in 2006.

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



Lic. Ramses Rodriguez Aulet

Workplace: Institute of Cryptography. University of Havana.

The education process: Currently candidate of Master of Science in Mathematic at the University of Havana.

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

MSc. Adrián Alfonso Peñate

Workplace: Institute of Cryptography. University of Havana.

The education process: Graduated of Mathematics in 2014; received Master's degree in 2018.

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