BLOCK CIPHER DEFINED BY MATRIX PRESENTATION OF QUASIGROUPS

1st Smile Markovski
Faculty of Computer Science & Eng.
Ss. Cyril and Methodius University
Skopje, R. N. Macedonia
smile.markovski@finki.ukim.mk

2nd Vesna Dimitrova
Faculty of Computer Science & Eng.
Ss. Cyril and Methodius University
Skopje, R. N. Macedonia
vesna.dimitrova@finki.ukim.mk

3rd Z. Trajcheska, M. Kostadinoski, D. Buhov
Faculty of Computer Science & Eng.
Ss. Cyril and Methodius University
Skopje, R. N. Macedonia

***Abstract*—Designing new cryptosystems and their cryptanalysis is the basic cycle of advancement in the field of cryptography. In this paper we introduce a block cipher based on the quasigroup transformations, which are defined by the matrix presentation of the quasigroup operations. This type of quasigroup presentation is suitable for constructing a block cipher since it doesn't require too much memory space to store all the necessary data, so it can be used even for lightweight cryptographic purposes.**

For now, we are considering only the quasigroups of order 4. Constructions with quasigroups of higer order and examination of the strengths and weaknesses of this design will be considered in next papers.

***Index Terms*—block cipher, quasigroup, matrix form of quasigroup**

I. Introduction

A block cipher is a symmetric key algorithm. The encryption (and accordingly decryption) is done by splitting the plaintext message into blocks - sequential groups of bits with fixed length and applying an invariant transformation on each block. Here, we will introduce a block cipher, BCMPQ (Block Cipher by Matrix Presentation of Quasigroups), that is based mostly on quasigroup transformations presented in a matrix form. A groupoid $(Q, *)$, where $*$ is a binary operation, is called a quasigroup if:

$$(\forall\, a, b \in Q)(\exists!\, x, y \in Q)(x * a = b \wedge a * y = b) \qquad (1)$$

The design of BCMPQ is based on the matrix presentation of the quasigroup operations [**4**]. Since the quasigroups we are considering are of order 4, their elements can be represented by two bits. In the sequel, for $x, y \in Q$ we use the notation $x = [x_1, x_2],\ y = [y_1, y_2]$ where $x_1, x_2, y_1, y_2 \in \{0, 1\}$.

Now, the quasigroup operation can be presented in matrix form as

$$x * y = m^T + Ax^T + By^T + CAx^T \cdot CBy^T \qquad (2)$$

where $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ and $B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$ are nonsingular Boolean matrices and $m = [m_1, m_2]$ is a Boolean vector. There are 4 choices for the matrix C (see [**4**]), and here we take the fixed matrix $C = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$. The operation "$\cdot$" denotes the dot product, i.e., it is the sum of the products of the corresponding components of the vectors CAx^T and CBy^T.

The encryption/decryption functions of the cipher are built by using e- and d- transformations [**2**]. Namely, given $a_1a_2\dots a_n,\ a_i \in Q$, and a fixed element $l \in Q$, called the *leader*, we define:

$$e_l(a_1a_2\dots a_n) = (b_1b_2\dots b_n) \Leftrightarrow b_1 = l * a_1,\ b_i = b_{i-1} * a_i,\ i \geq 2.$$

$$d_l(a_1a_2\dots a_n) = (c_1c_2\dots c_n) \Leftrightarrow c_1 = l * a_1,\ c_i = a_{i-1} * a_i,\ i \geq 2.$$

We note that e_l and d_l are permutations of the set Q^n since the equalities $d_l(e_l(a_1a_2\dots a_n)) = a_1a_2\dots a_n = e_l(d_l(a_1a_2\dots a_n))$ are true for each $a_i \in Q$.

II. Design of the Block Cipher

In this section we will state the design of the three basic algorithms characteristic for every block cipher - the round key generation, the encryption and the decryption algorithm.

There are altogether 144 quasigroups of form (2). We choose randomly 128 of them, they are public, and we store them in memory as follows:

$$\mathit{seq\_num} \quad m_1, m_2, a_{11}, a_{12}, a_{21}, a_{22}, b_{11}, b_{12}, b_{21}, b_{22} \qquad (3)$$

where $\mathit{seq\_num}$ is a seven bit number while $m_1, m_2, a_{11}, a_{12}, a_{21}, a_{22},\ b_{11}, b_{12}, b_{21}, b_{22}$ are the bits appearing in the matrix form (2) of the quasigroup operation.

The encryption and decryption algorithms include the use of 16 quasigroups. They are denoted by $Q_1, Q_2, ..., Q_8$ and $T_1, T_2, ..., T_8$ and they are used in different steps. These matrices are determined by using the round key, which is generated out of the secret key and consists of 128 bits.

The key length of 128 bits is distributed in the following way:

- 16 bits for the leaders $l_1, l_2, ..., l_8$ (two bits per each leader)
- 56 bits for the quasigroups $Q_1, Q_2, ..., Q_8$ (7 bits per each quasigroup)
- 56 bits for the quasigroups $T_1, T_2, ..., T_8$ (7 bits per each quasigroup)

The 7 bits designated for each quasigroup are actually the binary representation of the $\mathit{sequence\_number}$ of the quasigroup (see (3)).

A. Round Key Generation

At first, let us denote by K the secret symmetric key of 128 bits. In order to generate a round (working) key k out of the secret key, we first determine a fixed quasigroup Q and a fixed leader $l = 0 = [0, 0]$. This quasigroup should be determined to be non-fractal (see [**1**] for non-fractal quasigroups) such that

$x * x \neq x$ for $x \in Q$. The round key is obtained by e-transformation. The procedure for generation a round key is described in Algorithm on Fig. 1.

```
Algorithm II.1: GENERATEROUNDKEY()

Input: The secret key $K = K_1 K_2 \ldots K_{128}$.
Output: The round key $k = k_1 k_2 \ldots k_{128}$.
Initialization:
    $Q$ is a fixed quasigroup.
    $l = 00$

for $i \leftarrow 0$ to 128
    do $k[i] \leftarrow K[i]$

$l_{tmp} \leftarrow l$
for $i \leftarrow 1$ to 127 step 2
    do
        $tmp \leftarrow [k_j, k_{j+1}]$
        $[k_j, k_{j+1}] = m^T + A\, l_{tmp}^T + B\, tmp^T + CA\, l_{tmp}^T \cdot CB\, tmp^T$
        $l_{tmp} \leftarrow [k_j, k_{j+1}]$

$l_{tmp} \leftarrow l$
for $i \leftarrow 128$ to 2 step 2
    do
        $tmp \leftarrow [k_{j-1}, k_j]$;
        $[k_{j-1}, k_j] = m^T + A\, l_{tmp}^T + B\, tmp^T + CA\, l_{tmp}^T \cdot CB\, tmp^T$;
        $l_{tmp} \leftarrow [k_{j-1}, k_j]$;
```

Figure 1: Algorithm for Key Generation

```
Algorithm II.2: ENCRYPTBLOCK()

Input:
    The round key $k = k_1 k_2 \ldots k_{128}$.
    The plaintext message $a = a_1 a_2 \ldots a_{64}$.
Output: The ciphertext message $c = c_1 c_2 \ldots c_{64}$.
Initialization:
    $l_i = [k_{2i-1}, k_{2i}]$ for $i = 1, 2, \ldots, 8$

    Lookup the quasigroup $Q_i$ using the sequence number binary
    presented as $(k_{7i-6}, k_{7i-5}, \ldots, k_{7i})$ where $i = 1, 2, \ldots, 8$.
    Initialize the matrices $A_{Q_i}$ and $B_{Q_i}$, as well as the vector
    $m_{Q_i}$ for $i = 1, 2, \ldots, 8$.

    Lookup the quasigroup $T_i$ using the sequence number binary
    presented as $(k_{7(i+8)-6}, k_{7(i+8)-5}, \ldots, k_{7(i+8)})$. Initial-
    ize the matrices $A_{T_i}$ and $B_{T_i}$, as well as the vector $m_{T_i}$ for
    $i = 1, 2, \ldots, 8$.

for $i \leftarrow 1$ to 8 step 1
    do
        $l_{tmp} \leftarrow l_i$;
        for $j \leftarrow 1$ to 7 step 2
            do
                $tmp \leftarrow [a_j, a_{j+1}]$;
                $[c_j, c_{j+1}] = m_{Q_i}^T + A_{Q_i}\, l_{tmp}^T + B_{Q_i}\, tmp^T + CA_{Q_i}\, l_{tmp}^T \cdot CB_{Q_i}\, tmp^T$;
                $l_{tmp} \leftarrow [c_j, c_{j+1}]$;

for $i \leftarrow 1$ to 4 step 1
    do
        $l_{tmp} \leftarrow l_i$
        for $j \leftarrow 1$ to 64 step 2
            do
                $tmp \leftarrow [c_j, c_{j+1}]$;
                $[c_j, c_{j+1}] = m_{T_i}^T + A_{T_i}\, l_{tmp}^T + B_{T_i}\, tmp^T + CA_{T_i}\, l_{tmp}^T \cdot CB_{T_i}\, tmp^T$;
                $l_{tmp} \leftarrow [c_j, c_{j+1}]$;

        $l_{tmp} \leftarrow l_i$;
        for $j \leftarrow 64$ to 2 step 2
            do
                $tmp \leftarrow [c_{j-1}, c_j]$;
                $[c_{j-1}, c_j] = m_{T_{i+1}}^T + A_{T_{i+1}}\, l_{tmp}^T + B_{T_{i+1}}\, tmp^T + CA_{T_{i+1}}\, l_{tmp}^T \cdot CB_{T_{i+1}}\, tmp^T$;
                $l_{tmp} \leftarrow [c_{j-1}, c_j]$;
```

Figure 2: Algorithm for Encryption

B. Encryption Algorithm

We are considering a block cipher with block length 64 bits. So, the plaintext message should be split into blocks of 64 bits. Afterwards, the encryption algorithm stated in Algorithm **??** should be applied on each block. (If the message length is not devided by 64, a suitable padding will be applied). The encryption algorithm consists of two steps. In the first step we use the matrices $Q_1, Q_2, \ldots, Q_8$ and in the second the matrices $T_1, T_2, \ldots, T_8$.

Briefly, in the first step we split the 64 bit block into 8 smaller blocks (mini-blocks) of 8 bits. We apply e-transformation on each of these mini-blocks with a different leader and a different quasigroup. Actually, we use the leader l_i and the quasigroup Q_i for the i-th mini-blocks. The resulting string is used as input in the next step.

In the second step, we apply e-transformations on each resulting string, repeating 8 times with alternately changing direction. In the i-th transformation we use the quasigroup T_i and the leader l_i. The detailed and formalized algorithm is presented in Algorithm on Fig. 2.

C. Decryption Algorithm

The purpose of the decryption algorithm is to reverse the result of the encryption algorithm and thus to obtain the plaintext message given the ciphertext.

For decryption purposes we use the following property of quasigroups. Given a quasigroup $(Q, *)$ a new quasigroup $(Q, \backslash)$, so called parastrophe, is defined by

$$x \backslash z = y \;\Leftrightarrow\; x * y = z.$$

Then the identity $x \backslash (x * y) = y$ holds true, and it is used for decryption purposes. Since we are working with matrix presentation of quasigroups, it is shown in **[4]** that, when $(Q, *)$ is of the form (2), then $(Q, \backslash)$ has the following matrix presentation:

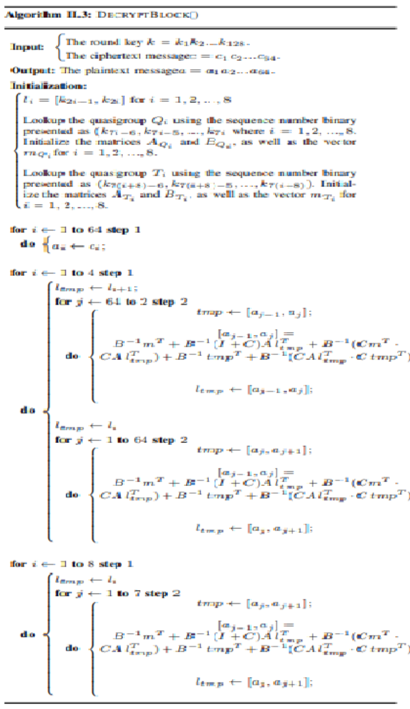
$$x \backslash z = B^{-1} m^T + B^{-1}(I + C)Ax^T + B^{-1}(Cm^T \cdot CAx^T) + B^{-1} z^T + B^{-1}(CAx^T \cdot Cz^T) \quad (4)$$

So, what we actually need to do to decrypt is to start from the ciphertext and reverse the e-transformation, using the quasigroups $T_8, T_7, \ldots, T_1$ sequentially at first, and then reverse the e-transformations of the mini-blocks (from the encryption algorithm) using the quasigroups $Q_8, Q_7, \ldots, Q_1$. This can be done using the inverse operation we mentioned shortly before. The detailed algorithm is presented in Algorithm on Fig. 3.

III. Estimation of the Memory Space Needed and Computing Power

Since the quasigroups data needs to be stored somewhere in the memory storage of the device where the algorithm will be implemented, it is important to estimate how much space is needed, particularly if we want to use this cipher in some smaller, lightweight devices where the resources, especially the memory space, are limited.
So, we need three parameters to define the quasigroup operation expressed by matrix operations. That are the vector m and the matrices A and B. To store m, A and B we need two bits for m, four bits for A and four bits for B. That is, 10 bits per each quasigroup. We are using 128 quasigroups, so that makes a total of 1280 bits, or 160 bytes, or 0.16 KB.

Having a look at the computational requirements of this block cipher, we need to determine the number of basic operations that are needed for encryption of one byte of the original message into respective byte of the encrypted message. Namely, here is



```
Algorithm II.3: DecryptBlock()
Input: { The round key $k = k_1k_2 \dots k_{128}$.
       { The ciphertext message$c = c_1\, c_2 \dots c_{64}$.
Output: The plaintext message$a = a_1\, a_2 \dots a_{64}$.
Initialization:
  { $l_i = [k_{2i-1}, k_{2i}]$ for $i = 1, 2, \dots, 8$
  { Lookup the quasigroup $Q_i$ using the sequence number binary
  {   presented as $(k_{7i-6}, k_{7i-5}, \dots, k_{7i}$ where $i = 1, 2, \dots, 8$.
  {   Initialize the matrices $A_{Q_i}$ and $B_{Q_i}$, as well as the vector
  {   $m_{Q_i}$ for $i = 1, 2, \dots, 8$.
  { Lookup the quasigroup $T_i$ using the sequence number binary
  {   presented as $(k_{7(i+8)-6}, k_{7(i+8)-5}, \dots, k_{7(i-8)})$. Initial-
  {   ize the matrices $A_{T_i}$ and $B_{T_i}$, as welll as the vector $m_{T_i}$ for
  {   $i = 1, 2, \dots, 8$.
for $i \leftarrow 1$ to 64 step 1
  do { $a_i \leftarrow c_i$;
for $i \leftarrow 1$ to 4 step 1
  do { $l_{tmp} \leftarrow l_{i+1}$;
     { for $j \leftarrow 64$ to 2 step 2
     {   do { $tmp \leftarrow [a_{j-1}, a_j]$;
     {      { $[a_{j-1}, a_j] =$
     {      {   $B^{-1}m^T + B^{-1}(I+C)A\, l_{tmp}^T + B^{-1}(Cm^T \cdot$
     {      {   $CA\, l_{tmp}^T) + B^{-1}\, tmp^T + B^{-1}[CA\, l_{tmp}^T \cdot C\, tmp^T)$
     {      { $l_{tmp} \leftarrow [a_{j-1}, a_j]$;
     { $l_{tmp} \leftarrow l_i$
     { for $j \leftarrow 1$ to 64 step 2
     {   do { $tmp \leftarrow [a_j, a_{j+1}]$;
     {      { $[a_{j-1}, a_j] =$
     {      {   $B^{-1}m^T + B^{-1}(I+C)A\, l_{tmp}^T + B^{-1}(Cm^T \cdot$
     {      {   $CA\, l_{tmp}^T) + B^{-1}\, tmp^T + B^{-1}[CA\, l_{tmp}^T \cdot C\, tmp^T)$
     {      { $l_{tmp} \leftarrow [a_j, a_{j+1}]$;
for $i \leftarrow 1$ to 8 step 1
  do { $l_{tmp} \leftarrow l_i$
     { for $j \leftarrow 1$ to 7 step 2
     {   do { $tmp \leftarrow [a_j, a_{j+1}]$;
     {      { $[a_{j-1}, a_j] =$
     {      {   $B^{-1}m^T + B^{-1}(I+C)A\, l_{tmp}^T + B^{-1}(Cm^T \cdot$
     {      {   $CA\, l_{tmp}^T) + B^{-1}\, tmp^T + B^{-1}[CA\, l_{tmp}^T \cdot C\, tmp^T)$
     {      { $l_{tmp} \leftarrow [a_j, a_{j+1}]$;
```

Figure 3: Algorithm for Decryption

a small overview of the number of additions and multiplications needed for encrypting two bits:

- for computing Ax^T we need 2 additions and 4 multiplications;
- for computing By^T we need 2 additions and 4 multiplications;
- for computing CAx^T we need 4 additions and 8 multiplications;
- for computing CBy^T we need 4 additions and 8 multiplications;
- for computing $CAx^T \cdot CBy^T$ we need 2 multiplications;
- in summery, for computing $m^T + Ax^T + By^T + CAx^T \cdot CBy^T$ we need 18 additions and 26 multiplications.

That means that for encrypting 8 bits (1 byte) we need to use 4*15=60 additions and 4*26=104 multiplications.

IV. Conclusion and future work

In this paper we introduced a block cipher which utilizes a matrix presentation of quasigroups. The aim of this paper is to show how small quasigroups, in this case of order 4, can be used a block cipher to be defined. As we have estimated, this cipher doesn't require too much space or computational power, so it is suitable for lightweight cryptographic applications and can be implemented and used for small devices. Also, the presented design is for transformation of message blocks of length 64 bits, and it is quite evident how the design can be made for message blocks of length $8l$, for any integer $l \geq 8$. (In fact, we can take l to be any positive integer, but for small l security will be vulnerable.)

In this paper only the main design is defined. We did not discussed several aspect important for a block cipher. Thus, we have to estimate the number of operations needed for encryption and decryption of one bite of information, and to compare it with other lightweight block cipher. Also, security of the cipher have to be considered too. The brute force attack is gained by the fact that we choose secret 16 quasigroups out of 128, so there are 2^{66} choices. Also, 8 two bit leaders have to be chosen, and there are 2^{16} choices for the leaders. So, there are 2^{82} possible choices for the quasigroups and the leaders.

A more thorough cryptanalysis of the cipher will be made in future in order to examine its security and reliability.

V. Acknowledgements

This research is partially supported by The Faculty for Computer Science and Engineering.

References

[1] Dimitrova, V: Quasigroup Processed Strings, their Boolean Representations and Application in Cryptography and Coding Theory. PhD Thesis (2010) Ss. Cyril and Methodius University, Skopje, Macedonia

[2] Markovski S., Gligoroski D., Bakeva V., Quasigroup String Processing: Part 1, Contributions, Section of Mathematical and Technical Sciences,, Macedonian Academy of Sciences and Arts, XX 1-2, 1999, pp. 13-28

[3] Mileva, A. (2010). Cryptographic Primitives with Quasigroup Transformations.

[4] Siljanoska, M., Mihova, M.,Markovski, S.: Matrix Presentation of Quasigroups of order 4, The 10th Conference for Informatics and Information Technology (CIIT 2013), Bitola, 2013

[5] Shcherbacov, V.A. (2009) Quasigroups in Cryptology. In Computer Science Journal of Moldova Volume 17, Number 2(50), Pages 193-228.