Quasigroups and Related Systems **23** (2015), 41 − 90

Design of crypto primitives based on quasigroups

Smile Markovski

Abstract. Today, the security of the modern world is undoubtedly dependent on the cryptographic primitives built into the various protocols used for secure communication. Let us mention here the most important, like block ciphers, stream ciphers, digital signatures and encryption schemes, hash functions, pseudo random number generators, ... The design of these, and many other crypto primitives, uses different concepts from number theory, group and finite field theory, Boolean algebras, etc. In this survey article we will present how quasigroups can be used for construction of various crypto primitives. We will discuss especially what type of quasigroups are used and how they can be constructed. Some open research problem will be mentioned as well.

1. Introduction

It is well known that *One-time-pad* is the only information theoretically secure cryptographic product, i.e., there is a mathematical proof of its security. All other cryptographic primitives that are massively used for different purposes in communication, banking, commerce and many other today human activities, have security based on three facts: first one is the believing (that some mathematical problems are hard to be solved), the second one is the experience (some cryptographic primitives cannot be broken after several years, even decades, of attacking), and the third one is the adjusting (whenever a weakness of some used cryptographic system is found, it is immediately repaired or changed). Thus, in the last decades no catastrophic damage was made of breaking some cryptographic product (maybe ENIGMA was the last one, more than seventy years ago).

Having in mind the previous, we realized that designs of new cryptographic primitives, products, systems, algorithms, protocols etc. have as well theoretical as practical importance. Nowadays, crypto primitives are produced mainly by using results from number and group theory, finite field theory, Boolean algebras and functions, and all of them are associative structures. We think that broadening the set of used theories with non-associative mathematical structures, like quasigroup

2010 Mathematics Subject Classification: 20N05, 94A60, 68P25
Keywords: Quasigroups (left, right, n-ary, huge, orthogonal, MQQ, MQLQ), cryptographic primitives, quasigroup transformations, cryptographic S-boxes, block ciphers, stream ciphers, PRNGs, hush functions, signatures, public encryptions schemes.

theory, that can be used for making suitable cryptographic primitives, is also important. In such a way cryptographic primitives with different properties of the existing ones can be obtained, hence the domain of the crypto primitives will be enlarged.

In this paper we show how the quasigroups can be used for building different type of cryptographic primitives. For that aim we define some type of quasigroups that are suitable for that purpose (Section 2), we give the definitions of several kinds of quasigroup transformations (Section 3), and we explain the constructions of some types of cryptographic primitives obtained by quasigroup transformations (Section 4). The last section contains discussion and conclusion.

2. Quasigroups

We start by giving a brief overview of the quasigroup theory that we will use in the sequel.

Definition 1. *A quasigroup $(Q,*)$ is a groupoid, i.e., a set Q with binary operation $*: Q^2 \to Q$, satisfying the law*

$$(\forall u, v \in Q)(\exists!\ x, y \in Q) \quad u * x = v \ \&\ y * u = v. \tag{1}$$

The definition is equivalent to the statements that $*$ is a cancellative operation ($x*y = x*z \Rightarrow y = z,\ y*x = z*x \Rightarrow y = z$) and the equations $a*x = b,\ y*a = b$ have solutions $x,\ y$ for each $a, b \in Q$.

In this paper we need only finite quasigroups, i.e., the order $|Q|$ of a quasigroup Q is a finite positive integer. Closely related combinatorial structures to finite quasigroups are the so called Latin squares:

Definition 2. *A Latin square L on a finite set Q of cardinality $|Q| = n$ is an $n \times n$-matrix with elements from Q such that each row and each column of the matrix is a permutation of Q.*

To any finite quasigroup $(Q,*)$, given by its multiplication table, there can be associated a Latin square L consisting of the matrix formed by the main body of the table, and each Latin square L on a set Q defines at most $|Q|!^2$ quasigroups $(Q,*)$ (obtained by all possible bordering).

A relation of *isotopism* and between two quasigroups are defined as follows.

Definition 3. *A quasigroup $(K,*)$ is said to be isotopic to a quasigroup $(Q,\bullet)$ if and only if there are bijections $\alpha,\ \beta,\ \gamma$ from K onto Q such that $\gamma(x*y) = \alpha(x)\bullet\beta(y)$ for each $x, y \in K$. Then the triple (α,β,γ) is called an isotopism from $(K,*)$ to $(Q,\bullet)$.*

Given a quasigroup $(Q,*)$ five new operations, so called parastrophes or adjoint operations, denoted by $\backslash$, $/$, $\bullet$, $\backslash\backslash$, $//$, can be derived from the operation $*$ as follows:

$$x*y = z \Leftrightarrow y = x\backslash z \Leftrightarrow\ x = z/y \Leftrightarrow y\bullet x = z \Leftrightarrow\ y = z\backslash\backslash x\ \Leftrightarrow\ x = y//z. \tag{2}$$

Then the algebras $(Q, *, \backslash, /)$ and $(Q, \bullet, \backslash\backslash, //)$ satisfy the identities

$$x \backslash (x * y) = y,\ x * (x \backslash y) = y,\ (x * y)/y = x,\ (x/y) * y = x \tag{3}$$

$$y = (x * y) \backslash\backslash x = (y \bullet x) \backslash\backslash x,\ x = y//(x * y) = y//(y \bullet x),$$
$$y = x * (y \backslash\backslash x) = (y \backslash\backslash x) \bullet x,\ x = (y//x) * y = y \bullet (y//x), \tag{4}$$

and $(Q, \backslash)$, $(Q, /)$, $(Q, \bullet)$ $(Q, \backslash\backslash)$, $(Q, //)$ are quasigroups too.

2.1. n-ary, left and right quasigroups

An n-ary quasigroup is a pair (Q, f) of a nonempty set Q and an n-ary operation f with the property that given any n of the elements $a_1, a_2, \dots, a_{n+1} \in Q$, the $n+1$-th is uniquely determined the equality $f(a_1, a_2, \dots, a_n) = a_{n+1}$ hold true. A quasigroup is a binary (2-ary) quasigroup. Given n-ary quasigroup (Q, f), we define n operations $f_1, f_2, \dots, f_n$ by

$$f(a_1, a_2, \dots, a_n) = a_{n+1} \Leftrightarrow f_i(a_1, \dots, a_{i-1}, a_{n+1}, a_{i+1}, \dots, a_n) = a_i.$$

Then the following identities holds, for each $i = 1, 2, \dots, n$:

$$f(a_1, \dots, a_{i-1}, f_i(a_1, \dots, a_n), a_{i+1} \dots, a_n) = a_i,$$
$$f_i(a_1, \dots, a_{i-1}, f(a_1, a_2, \dots, a_n), a_{i+1}, \dots, a_n) = a_i. \tag{5}$$

Definition 4. *A left (right) quasigroup $(Q, *)$ is a groupoid satisfying the law*

$$(\forall u, v \in Q)(\exists!\ x \in Q) \quad u * x = v$$
$$((\forall u, v \in Q)(\exists!\ y \in Q) \quad y * u = v.)$$

It is clear that a groupoid is a quasigroup iff it is left and right quasigroup.

Given a left (right) quasigroup $(Q, *)$ the parastrophe $\backslash$ $(/)$ can be derived from the operation $*$ as following.

$$x * y = z \Leftrightarrow y = x \backslash z \quad (x * y = z \Leftrightarrow x = z/y)$$

and then the algebra $(Q, *, \backslash)$ $((Q, *, /))$ satisfies the identities

$$x \backslash (x * y) = y,\ x * (x \backslash y) = y, \quad ((x * y)/y = x,\ (x/y) * y = x).$$

2.2. Huge quasigroups

A quasigroup can be constructed by using a Latin square, that will be the main body of the multiplication table of the quasigroup, or analytically by some functions. A quasigroup of small order is easily representable by its multiplication table (as in Table 3). Clearly, it cannot be done for quasigroups of huge orders 2^{16}, 2^{64}, 2^{128}, 2^{256}, $2^{512}, \dots$ (we say huge quasigroups), that are used in the constructions of some crypto primitives. There are several known constructions of huge quasigroups, and we describe some of them.

2.2.1. Huge quasigroups obtained by Feistel networks

Extended Feistel networks $F_{A,B,C}$ are defined in [91] as follows.

Let $(G,+)$ be an abelian group, let $f: G \to G$ be a mapping and let $A, B, C \in G$ be constants. The extended Feistel network $F_{A,B,C}: G^2 \to G^2$ created by f is defined for every $(l,r) \in G^2$ as

$$F_{A,B,C}(l,r) = (r + A, l + B + f(r + C)).$$

When f is a bijection, $F_{A,B,C}$ is an orthomorphism of the group $(G^2,+)$ (i.e., $F_{A,B,C}$ and $F_{A,B,C} - I$ are permutations), so a quasigroup $(G^2, *_{F_{A,B,C}})$ can be produced by Sade's diagonal method [122] as

$$X *_{F_{A,B,C}} Y = F_{A,B,C}(X - Y) + Y.$$

This construction is suitable for many applications, since the parameters A, B, C of an extended Feistel network $F_{A,B,C}$ can be used for different purposes. By iterating, starting from a group of small order, we can construct a huge quasigroup. Namely, if f is bijection on G, then $f_1 = F_{A,B,C}$ is a bijection on G^2, so we can define suitable extended Feistel network F_{A_1,B_1,C_1} by choosing constants $A_1, B_1, C_1 \in G^2$. Again, by Sade's diagonal method we can construct a quasigroup $(G^4, *_{F_{A_1,B_1,C_1}})$ of order $|G|^4$. Hence, in such a way, after k steps, we have a quasigroup of order $|G|^{2^k}$. Thus, when $G = \mathbb{Z}_2$, after 8 steps wee have a huge quasigroup of order 2^{256}. Note that only the starting bijection f has to be kept in memory.

More constructions of quasigroups by different types of Feistel networks are given in [111].

2.2.2. Huge quasigroups obtained by T-functions

Huge quasigroups can be defined by so called $T-$functions [18] and one way how it can be done is the following [127].

Let $Q = \mathbb{Z}_{2^w}$ and let represent the element of Q binary, as bit strings of length w. (Thus, for $w = 4$, the integer 9 is represented as 1001.) Let $x_w, \dots, x_1$ be Boolean variables, and let $\mathbf{b}$ be a constant Boolean vector. Let $\mathbf{A_1} = [f_{ij}]_{w\times w}$ and $\mathbf{A_2} = [g_{ij}]_{w\times w}$ be upper triangular matrices of linear Boolean expressions with variables $x_w, \dots, x_1$, such that: 1) $f_{ii} = 1$, $g_{ii} = 1$ and f_{iw} are constants for every $i = 1, \dots, w$; 2) for all $i < j < w$, f_{ij} can depend only on the variables $x_{w-j}, \dots, x_1$ and 3) for all $i < j$, g_{ij} can depend only on the variables $x_w, \dots, x_1$. Let $\mathbf{x} = (x_w, \dots, x_1)$, $\mathbf{y} = (y_w, \dots, y_1)$ be binary presentation of the variables $\mathbf{x}, \mathbf{y}$ over Q. Then, $(Q, *)$ is a quasigroup of order 2^w, where $*$ is defined by

$$\mathbf{x} * \mathbf{y} = \mathbf{A_1} \cdot (x_w, \dots, x_1)^T + \mathbf{A_2} \cdot (y_w, \dots, y_1)^T + \mathbf{b}^T.$$

The parastrophe $(Q, \backslash)$ is defined by

$$\mathbf{x} \setminus \mathbf{y} = \mathbf{A_2^{-1}} \cdot ((y_w, \dots, y_1)^T - \mathbf{A_1} \cdot (x_w, \dots, x_1)^T - \mathbf{b}^T).$$

2.2.3. Huge quasigroups obtained by simple isotopies

The compression function of the hash function Edon-R [58] uses two huge quasigroups of order 2^{256} and 2^{512} and their operations are defined by isotopies of the Abelian group $((\mathbb{Z}_{2^w})^8, +_8)$, $w = 32$ and $w = 64$, respectfully. ($+_8$ is a componentwise addition on two 8-dimensional vectors in $(\mathbb{Z}_{2^w})^8$). The quasigroup operation $*$ is defined by

$$X * Y = \pi_1(\pi_2(X) +_8 \pi_3(Y))$$

where $X = (X_0, X_1, \dots, X_7), Y = (Y_0, Y_1, \dots, Y_7) \in (\mathbb{Z}_{2^w})^8$ and $\pi_i : \mathbb{Z}_{2^w} \to \mathbb{Z}_{2^w}$, $1 \leqslant i \leqslant 3$, are permutations obtained in a suitable simple (and efficient) way.

2.3. Quasigroups for symbolic computations

Designs of some crypto primitives (like digital signatures, public key encryptions) need symbolic computations. (For example, for producing a public key consisting of polynomials.) For that aim, quasigroups capable for symbolic computations are defined as well.

2.3.1. Multivariate quadratic quasigroups (MQQ)

As we already mentioned, the elements of a finite quasigroups $(Q, *)$ of order 2^d can be represented binary as bit strings of d bits. Now, the binary operation $*$ can be interpreted as a vector valued operation $*_{vv} : \{0,1\}^{2d} \to \{0,1\}^d$ defined as:

$$\mathbf{x} * \mathbf{y} = \mathbf{z} \iff *_{vv}(x_1, \dots, x_d, y_1, \dots, y_d) = (z_1, \dots, z_d),$$

where $x_1 \dots x_d,\ y_1 \dots y_d,\ z_1 \dots z_d$ are binary representations of $\mathbf{x}, \mathbf{y}, \mathbf{z}$. Each z_i depends of the bits $x_1, x_2, \dots, x_d, y_1, y_2, \dots, y_d$ and is uniquely determined by them. So, each z_i can be seen as a $2d$-ary Boolean function $z_i = f_i(x_1, \dots, x_d, y_1, \dots, y_d)$, where $f_i : \{0,1\}^{2d} \to \{0,1\}$ strictly depends on, and is uniquely determined by, $*$.

A k-ary Boolean function $f(x_1, \dots, x_k)$ can be represented in a unique way by its algebraic normal form (ANF) as a sum of products in the field $GF(2)$:

$$ANF(f) = \sum_{I \subseteq \{1,2,\dots,k\}} \alpha_I \mathbf{x}^I,$$

where $\alpha_I \in \{0,1\}$ and $\mathbf{x}^I$ is the product of all variables x_i such that $i \in I$. The ANFs of the functions f_i give us information about the complexity of the quasigroup $(Q, *)$ via the degrees of the Boolean functions f_i. It can be observed that the degrees of the polynomials $ANF(f_i)$ rise with the order of the quasigroup. In general, for a randomly generated quasigroup of order 2^d, $d \geqslant 4$, the degrees are higher than 2.

The MQQ are defined in [51]. A quasigroup $(Q, *)$ of order 2^d is called Multivariate Quadratic Quasigroup (MQQ) of type $Quad_{d-k}Lin_k$ if exactly $d - k$ of its Boolean polynomials f_i are of degree 2 (i.e., they are quadratic) and k of them are of degree 1 (i.e., they are linear), where $0 \leqslant k < d$.

Theorem 1 below gives us sufficient conditions for a quasigroup $(Q, *)$ to be MQQ.

Theorem 1. *Let $\mathbf{A_1} = [f_{ij}]_{d\times d}$ and $\mathbf{A_2} = [g_{ij}]_{d\times d}$ be two $d \times d$ matrices of linear Boolean expressions, and let $\mathbf{b_1} = [u_i]_{d\times 1}$ and $\mathbf{b_2} = [v_i]_{d\times 1}$ be two $d \times 1$ vectors of linear or quadratic Boolean expressions. Let the functions f_{ij} and u_i depend only on variables $x_1, \dots, x_d$, and let the functions g_{ij} and v_i depend only on variables $x_{d+1}, \dots, x_{2d}$. If*

$$\mathbf{Det}(\mathbf{A_1}) = \mathbf{Det}(\mathbf{A_2}) = 1 \textit{ in } GF(2) \tag{6}$$

and if

$$\mathbf{A_1} \cdot (x_{d+1}, \dots, x_{2d})^T + \mathbf{b_1} \equiv \mathbf{A_2} \cdot (x_1, \dots, x_d)^T + \mathbf{b_2} \tag{7}$$

then the vector valued operation

$$*_{vv}(x_1, \dots, x_{2d}) = \mathbf{A_1} \cdot (x_{d+1}, \dots, x_{2d})^T + \mathbf{b_1}$$

defines a quasigroup $(Q, *)$ of order 2^d that is MQQ.

Similarly as in Theorem 1, a construction of so called Mutually Quadratic Left Quasigroups (MQLQ) is given in [124].

Theorem 2. *Let $x_1, \dots, x_w, y_1, \dots, y_w$ be Boolean variables, $w > 1$. Let $\mathbf{A_1} = [f_{ij}]_{w\times w}$ and $\mathbf{A_2} = [g_{ij}]_{w\times w}$ be two $w \times w$ nonsingular upper triangular matrices of random affine Boolean expressions, such that for every $i = 1, \dots, w$, $f_{ii} = 1$ and $g_{ii} = 1$, and for all i, j, $i < j \leqslant w$, f_{ij} and g_{ij} depend only on the variables $x_1, \dots, x_w$, $y_{i+1}, \dots, y_w$. Let $\mathbf{D_1} = [d_{ij}]_{w\times w}$, $\mathbf{D_2} = [d_{ij}]_{w\times w}$ and $\mathbf{D} = [d_{ij}]_{w\times w}$ be nonsingular Boolean matrices and let $\mathbf{b} = [b_i]_{w\times 1}$, $\mathbf{c_1} = [c_i]_{w\times 1}$, $\mathbf{c_2} = [c_i]_{w\times 1}$ and $\mathbf{c} = [c_i]_{w\times 1}$ be Boolean vectors.*

Then the vector valued operations

$$*_1(x_1, \dots, x_w, y_1, \dots, y_w) = \mathbf{A_1} \cdot (x_1, \dots, x_w) \oplus \mathbf{A_2} \cdot (y_1, \dots, y_w) \oplus \mathbf{b} \tag{8}$$

and

$$*_2(x_1, \dots, x_w, y_1, \dots, y_w) = \mathbf{D}(*_1(\mathbf{D}_1(x_1, \dots, x_w) \oplus \mathbf{c}_1, \mathbf{D}_2(y_1, \dots, y_w) \oplus \mathbf{c}_2)) \oplus \mathbf{c} \tag{9}$$

define left quasigroups $(Q, *_1)$ and $(Q, *_2)$ of order 2^w that are MQLQ, where $Q = \{0, 1, \dots \dots, 2^w - 1\}$.

The definition of the quasigroup $(Q, *)$ implies immediately that symbolic computations can be performed with linear polynomials on the field $GF(2)$.

More information for MQQ can be found in [125] and [17].

2.3.2. Matrix representation

All quasigroup operations on the set $Q = \{0, 1, 2, 3\}$ have so called matrix representations in the following form, given in the next theorem [137]:

Theorem 3. *Each quasigroup $(Q, *)$ of order 4 has a matrix representation of form*

$$\boldsymbol{x} * \boldsymbol{y} = \boldsymbol{m}^T + A\boldsymbol{x}^T + B\boldsymbol{y}^T + CA\boldsymbol{x}^T \circ CB\boldsymbol{y}^T, \tag{10}$$

where $\boldsymbol{x} = (x_1, x_2)$, $\boldsymbol{y} = (y_1, y_2) \in Q$ (x_i, y_i denotes bit variables), $\boldsymbol{m} = (m_1, m_2)$ is some constant from Q, A and B are nonsingular 2-dimensional matrices of bits, C is one of the matrices $\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$, $\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$, $\begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$, $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$, and $\circ$ denotes the component-wise multiplication of vectors. (Note: The addition and multiplication are in the field $GF(2)$.)

The matrix presentation of the parastrophe operation $\backslash$ of the quasigroup operation $*$ given by (10) is the following:

$$\begin{aligned} \mathbf{x} \backslash \mathbf{z} = B^{-1}\mathbf{m}^T + B^{-1}(I + C)A\mathbf{x}^T + B^{-1}(C\mathbf{m}^T \circ CA\mathbf{x}^T) + \\ + B^{-1}\mathbf{z}^T + B^{-1}(CA\mathbf{x}^T \circ C\mathbf{z}^T), \end{aligned} \tag{11}$$

where I denotes the identity matrix.

The variables $\mathbf{x}$ and $\mathbf{y}$ may vary over the polynomial ring $\mathbb{Z}[X]$, not only over Q.

Matrix representation for some types of quasigroups of order 8 are also investigated.

Here we note that for any order 2^n we have matrix representations of the so called linear quasigroups of order 2^n as follows. Let denote $\mathbf{x} = (x_1, \dots, x_n)$, where x_i are bit variables.

Theorem 4. *Let A and B be nonsingular binary $n \times n-$matrices and $\mathbf{m} \in Q = \{0, 1, 2, \dots, 2^n - 1\}$ be a constant. Then $(Q, *)$ is a quasigroup (called linear quasigroup), where*

$$\boldsymbol{x} * \boldsymbol{y} = \boldsymbol{m}^T + A\boldsymbol{x}^T + B\boldsymbol{y}^T. \tag{12}$$

By Theorem 4, we can take $\mathbf{m}$ to be $1 \times n$ matrix and A and B to be $n \times n-$matrices with entries that are Boolean expressions such that $det(A) \neq 0$, $det(B) \neq 0$, and then $\mathbf{x} * \mathbf{y}$ will be a Boolean expression too. So, we can use this matrix presentation for symbolic computations.

We can extended the previous result for k-ary case as well.

Theorem 5. *Let A_i, $i = 1, 2, \dots, k$, be nonsingular binary $n \times n-$matrices and $\mathbf{m} \in Q = \{0, 1, 2, \dots, 2^n - 1\}$ be a constant. Then (Q, f) is a k-ary quasigroup (called linear $k-$quasigroup), where*

$$f(\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_k) = \boldsymbol{m}^T + A_1\boldsymbol{x}_1^T + A_2\boldsymbol{x}_2^T + \cdots + A_k\boldsymbol{x}_k^T. \tag{13}$$

2.3.3. Polynomial quasigroups

Quasigroups of order 2^n can be defined by using bivariate polynomials $P(x,y)$ over the ring $(Z_{2^n},+,\cdot)$, $n \geqslant 2$ ([123], [121], [95]), when the polynomials satisfy the following condition: each of the functions $P(x,0), P(x,1), P(0,y)$ and $P(1,y)$ is a permutation on $\mathbb{Z}_{2^n}$. Then the quasigroup $(\mathbb{Z}_{2^n},*)$, called polynomial quasigroup, is defined by $x*y=P(x,y)$.

If only the univariate polynomials $P(x,0)$, $P(x,1)$ $(P(0,y),P(1,y))$ are permutations, $(\mathbb{Z}_{2^n},*)$ is right quasigroup (left quasigroup), and vice versa.

Polynomial quasigroups can be of huge order and, since defined by polynomials, they can be used for symbolic computations as well.

We note that there are effective algorithms for computing the parastrophic operations of the polynomial quasigroups.

3. Quasigroup string transformations

Quasigroup String Transformations were introduced in [84] and were investigated in several other papers ([85], [89], [90], [81], [80]).

Consider an alphabet (i.e., a finite set) Q, and denote by $Q^+=\{a_1a_2\dots \dots a_n|\ a_i\in Q\}$ the set of all nonempty words (i.e., finite strings) formed by the elements of Q. (If there is no misunderstanding, we identify $a_1a_2\dots a_n$ and $(a_1,a_2,\dots,a_n)$.) Let $*$ be a quasigroup operation on the set Q, i.e., consider a quasigroup $(Q,*)$. For each $a\in Q$ we define two functions

$$e_{a,*},\ d_{a,*}: Q^+ \longrightarrow Q^+$$

as follows. Let $a_i\in Q,\ \alpha=a_1a_2\dots a_n$. Then

$$e_{a,*}(\alpha)=b_1b_2\dots b_n \Longleftrightarrow b_1=a*a_1,\ b_2=b_1*a_2,\dots,\ b_n=b_{n-1}*a_n$$

and

$$d_{a,*}(\alpha)=c_1c_2\dots c_n \Longleftrightarrow c_1=a*a_1,\ c_2=a_1*a_2,\dots,\ c_n=a_{n-1}*a_n.$$

The functions $e_{a,*},\ d_{a,*}$ are called e- and d-transformation of Q^+ based on the operation $*$ with leader a, and their graphical representation is shown on Fig. 1 and Fig. 2.

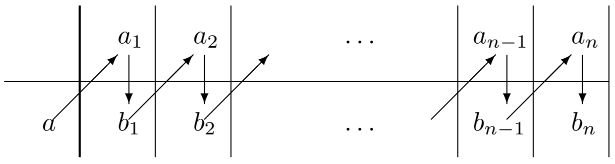


Figure 1: Graphical representation of $e_{a,*}$ function

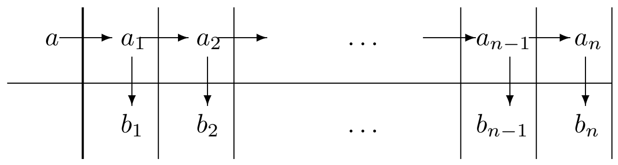


Figure 2: Graphical representation of $d_{a,*}$ function

$\bullet$	0	1	2	3
0	2	1	0	3
1	3	0	1	2
2	1	2	3	0
3	0	3	2	1

$\backslash$	0	1	2	3
0	2	1	0	3
1	1	2	3	0
2	3	0	1	2
3	0	3	2	1

$/$	0	1	2	3
0	3	1	0	2
1	2	0	1	3
2	0	2	3	1
3	1	3	2	0

Figure 3: A quasigroup $(Q, \bullet)$ and its parastrophes $(Q, \backslash)$ and $(Q, /)$

Example 1. Take $Q = \{0, 1, 2, 3\}$ and let the quasigroup $(Q, \bullet)$ and its parastrophes $(Q, \backslash)$ and $(Q, /)$ be given by the multiplication schemes in Figure 3.

Consider the string $\alpha = 1\ 0\ 2\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 2\ 1\ 0\ 2\ 2\ 0\ 1\ 0\ 1\ 0\ 3\ 0\ 0$ and choose the leader 0. Then by the transformations $e_{0,\bullet}$ and $d_{0,\bullet}$ we will obtain the following transformed strings $e_{0,\bullet}(\alpha)$ and $d_{0,\bullet}(\alpha)$:

$e_{0,\bullet}(\alpha) = 1\ 3\ 2\ 2\ 1\ 3\ 0\ 2\ 1\ 3\ 0\ 2\ 1\ 0\ 1\ 1\ 2\ 1\ 1\ 1\ 3\ 3\ 0\ 1\ 3\ 1\ 3\ 0$,

$d_{0,\bullet}(\alpha) = 1\ 3\ 0\ 2\ 3\ 2\ 2\ 2\ 2\ 2\ 2\ 2\ 2\ 1\ 0\ 1\ 2\ 3\ 0\ 3\ 1\ 1\ 3\ 1\ 3\ 3\ 0\ 2$.

We present four consecutive applications of the e-transformation on Table 1.

After that we apply four times the transformation $d_{0,\backslash}$ on the last obtained string $\beta = {e_{0,\bullet}}^4(\alpha)$ (see Table 1):

Notice that we have obtained

$$\alpha = {d_{0,\backslash}}^4(\beta) = {d_{0,\backslash}}^4({e_{0,\bullet}}^4(\alpha)) = ({d_{0,\backslash}}^4 \circ {e_{0,\bullet}}^4)(\alpha). \qquad \square$$

In fact, by (3), the following property is true([85]):

Theorem 6. *Let $(Q, *, \backslash, /)$ be a finite quasigroup. Then for each string $\alpha \in Q^+$ and for each leader $l \in Q$ we have that $e_{l,*}$ and $d_{l,\backslash}$ are mutually inverse permutations of Q^+, i.e., $d_{l,\backslash}(e_{l,*}(\alpha)) = \alpha = e_{l,*}(d_{l,\backslash}(\alpha))$.*

leader	$1\ 0\ 2\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 2\ 1\ 0\ 2\ 2\ 0\ 1\ 0\ 1\ 0\ 3\ 0\ 0 = \alpha$
0	$1\ 3\ 2\ 2\ 1\ 3\ 0\ 2\ 1\ 3\ 0\ 2\ 1\ 0\ 1\ 1\ 2\ 1\ 1\ 1\ 3\ 3\ 0\ 1\ 3\ 1\ 3\ 0 = e_{0,\bullet}(\alpha)$
0	$1\ 2\ 3\ 2\ 2\ 0\ 2\ 3\ 3\ 1\ 3\ 2\ 2\ 1\ 0\ 1\ 1\ 2\ 2\ 2\ 0\ 3\ 0\ 1\ 2\ 2\ 0\ 2 = {e_{0,\bullet}}^2(\alpha)$
0	$1\ 1\ 2\ 3\ 2\ 1\ 1\ 2\ 0\ 1\ 2\ 3\ 2\ 2\ 1\ 0\ 1\ 1\ 1\ 1\ 3\ 1\ 3\ 3\ 2\ 3\ 0\ 0 = {e_{0,\bullet}}^3(\alpha)$
0	$1\ 0\ 0\ 3\ 2\ 2\ 2\ 3\ 0\ 1\ 1\ 2\ 3\ 2\ 2\ 1\ 0\ 1\ 0\ 1\ 2\ 2\ 0\ 3\ 2\ 0\ 2\ 1 = {e_{0,\bullet}}^4(\alpha)$

Table 1: Consecutive e-transformations

leader	
0	1 0 0 3 2 2 2 3 0 1 1 2 3 2 2 1 0 1 0 1 2 2 0 3 2 0 2 1 $= \beta$
0	1 1 2 3 2 1 1 2 0 1 2 3 2 2 1 0 1 1 1 1 3 1 3 3 2 3 0 0 $= d_{0,\backslash}(\beta)$
0	1 2 3 2 2 0 2 3 3 1 3 2 2 1 0 1 1 2 2 2 0 3 0 1 2 2 0 2 $= d_{0,\backslash}{}^2(\beta)$
0	1 3 2 2 1 3 0 2 1 3 0 2 1 0 1 1 2 1 1 1 3 3 0 1 3 1 3 0 $= d_{0,\backslash}{}^3(\beta)$
	1 0 2 1 0 0 0 0 0 0 0 0 0 1 1 2 1 0 2 2 0 1 0 1 0 3 0 0 $= d_{0,\backslash}{}^4(\beta)$

Table 2: Consecutive d-transformations

By Theorem 6 we conclude that the transformations $e_{a,*}$ and $d_{a,\backslash}$ can be used for defining suitable functions for encryption and decryption. Much more, we can define in the similar way several pairs of quasigroup string transformations that can be used for defining suitable functions for encryption and decryption. Thus, let $a, a_1, a_2, \dots, a_n \in Q$ and let define the functions $e'_{a,*},\ d'_{a,*} : Q^+ \longrightarrow Q^+$ as follows:

$$e'_{a,*}(\alpha) = b_1 b_2 \dots b_n \Longleftrightarrow b_n = a_n * a,\ b_{n-1} = a_{n-1} * b_n, \dots,\ b_1 = a_1 * b_2,$$

$$d'_{a,*}(\alpha) = c_1 c_2 \dots c_n \Longleftrightarrow c_n = a_n * a\ c_{n-1} = a_{n-1} * a_n, \dots,\ c_1 = a_1 * a_2.$$

Then, by (4), Theorem 6 holds for the functions $e'_{a,*},\ d'_{a,\backslash}$ too. Also, for encryption/decryption purposes, in a suitable way transformations with the pair of functions $(e_{a,*},\ d_{a,/}), (e'_{a,*},\ d'_{a,/}), (e_{a,*},\ d_{a,\backslash}), (e'_{a,*},\ d'_{a,\backslash}), (e_{a,\bullet},\ d_{a,//}),\ (e'_{a,\bullet},\ d'_{a,//})$, $(e_{a,\bullet},\ d_{a,\backslash\backslash}), (e'_{a,\bullet},\ d'_{a,\backslash\backslash})$ can be defined in an obvious way.

Several quasigroup operations can be defined on the set Q and let $*_1,\ *_2,\ \dots,\ *_k$ be a sequence of (not necessarily distinct) such operations. We choose also leaders $l_1,\ l_2,\ \dots,\ l_k \in Q$ (not necessarily distinct either), and let $t^{(i)} \in \{e_{l_i,*},\ d_{l_i,/},\ e'_{l_i,*},\ d'_{l_i,/},\ d_{l_i,//},\ d'_{l_i,//},\ d_{a,\backslash\backslash},\ d'_{l_i,\backslash\backslash}, \dots\}$. Then, the transformation $T = t^{(1)} t^{(2)} \dots t^{(k)}$ of Q^+ is said to be a generalized T-transformation. It is a permutation of Q^+ with inverse $T^{-1} = (t^{(k)})^{-1} (t^{(k-1)})^{-1} \dots (t^{(1)})^{-1}$, where $(t^{(i)})^{-1} \in \{e_{l_i,*},\ e'_{l_i,*},\ e_{l_i,\bullet},\ e'_{l_i,\bullet},\ d_{l_i,/},\ d'_{l_i,/},\ d_{l_i,//},\ d'_{l_i,//},\ d_{a,\backslash\backslash},\ d'_{l_i,\backslash\backslash}, \dots\}$. The generalized transformations T, T^{-1} can be used as encryption/decryption functions.

3.1. Parastrophic quasigroup transformations

In order to exploit more completely one quasigroup, an idea for quasigroup string transformation that will be based on all isotopes of a quasigroup is given in [73]. Here we give a description of a slightly modification of this transformation, called parastrophic quasigroup transformation, as presented in [3]. For that aim we denote the parastrophic operation $\{*, \backslash, /, \bullet, //, \backslash\backslash\}$ of a quasigroup $(Q, *)$ respectively as $f_1, f_2, f_3, f_4, f_5, f_6$, and we write $f_1(x, y), f_2(x, y), \dots$ instead of $x * y, x\backslash y, \dots$ Note that some of the parastrophes f_i may coincides, depending of the quasigroup.

The parastrophic transformations PE is defined on finite quasigroups $(Q, *)$ of integers, i.e., $Q = \{1, 2, \ldots, t\}$. They are using the transformations e_{l,f_i} for transformations of block of letters, where l is a leader. Also, a positive integer p is used.

Let p be a positive integer and $x_1x_2 \ldots x_n$ be an input message. We define a parastrophic transformation $PE = PE_{l,p} : Q^+ \to Q^+$ by using auxiliary parameters d_i, q_i and s_i as follows.

To start, let $d_1 = p$, $q_1 = d_1$, $s_1 = (d_1 \bmod 6) + 1$ and take a starting block $A_1 = x_1x_2 \ldots x_{q_1}$. Denote by B_1 the block

$$B_1 = y_1y_2 \ldots y_{q_1-1}y_{q_1} = e_{l,f_{s_1}}(x_1x_2 \ldots x_{q_1-1}x_{q_1}).$$

Further, we calculate the numbers $d_2 = 4y_{q_1-1} + y_{q_1}$ (that determines the length of the next block), $q_2 = q_1 + d_2$ and $s_2 = (d_2 \bmod 6) + 1$. We denote $A_2 = x_{q_1+1} \ldots x_{q_2-1}x_{q_2}$ and

$$B_2 = y_{q_1+1} \ldots y_{q_2-2}y_{q_2-1}y_{q_2} = E_{y_{q_1},f_{s_2}}(x_{q_1+1} \ldots x_{q_2-2}x_{q_2-1}x_{q_2}).$$

Inductively, after getting the blocks $B_1, B_2, \ldots, B_{i-1}$ where $B_{i-1} = y_{q_{i-2}+1} \ldots$ $\ldots y_{q_{i-1}-1}y_{q_{i-1}}$, we calculate $d_i = 4y_{q_{i-1}-1} + y_{q_{i-1}}$, $q_i = q_{i-1} + d_i$, $s_i = (d_i \bmod 6) + 1$, $A_i = x_{q_{i-1}+1} \ldots x_{q_i-1}x_{q_i}$ and obtain the block

$$B_i = E_{y_{q_{i-1}},f_{s_i}}(x_{q_{i-1}+1} \ldots x_{q_i}).$$

Now, the parastrophic transformation $PE_{l,p}$ is defined by concatenation of the obtained blocks as

$$PE_{l,p}(x_1x_2 \ldots x_n) = B_1||B_2|| \ldots ||B_r. \tag{14}$$

(Note that the length of the last block A_r may be shorter than d_r, depending on the number of letters in input message).

3.2. Other types of transformations

For different purposes other types of quasigroups transformations are defined elsewhere. We will shortly mention some of them.

Special kind of E transformation is the quasigroup reverse string transformation R, introduced in [42], where the leaders are the elements of the string, taken in reverse order. Namely, a string of letters $\alpha = a_1a_2 \ldots a_n$ is transformed to $E(\alpha)$, where $E = e_{*,a_n} \circ e_{*,a_{n-1}} \circ \cdots \circ e_{*,a_1}$.

Let $(Q, *_1)$ and $(Q, *_2)$ be two orthogonal (finite) quasigroups, i.e., the equality $\{(x *_1 y, x *_2 y) | \; x, y \in Q\} = Q^2$ holds. Orthogonal quasigroup string transformation $OT : Q^+ \to Q^+$ of a string $x_1x_2 \ldots x_r$ is defined in [110] by the following iterative procedure:

$$OT(x_1) = x_1, OT(x_1, x_2) = (x_1 *_1 x_2, x_1 *_2 x_2)$$

and if $OT(x_1, x_2, \dots, x_{t-1}) = (z_1, z_2, , \dots, z_{t-1})$ is defined for $t > 2$, then

$$OT(x1, x2, , \dots, x_{t-1}, x_t) = (z_1, z_2, , \dots, z_{t-2}, z_{t-1} *_1 x_t, z_{t-1} *_2 x_t).$$

OT is a permutation of Q.

Let $Q = \mathbb{Z}_{2^n}$, let $(Q, *)$ be a quasigroup and let $+$ denote addition modulo 2^n. Elementary quasigroup additive and reverse additive string transformations $A, RA : Q^+ \to Q^+$ with leader l are defined in [92] as follows:

$$A(x_1x_2 \dots x_t) = (z_1z_2, \dots z_t) \Longleftrightarrow z_j = (z_{j-1} + x_j) * x_j, \ 1 \leqslant j \leqslant t, \quad z_0 = l,$$

$$RA(x_1x_2 \dots x_t) = (z_1z_2 \dots z_t) \Longleftrightarrow z_j = x_j * (x_j + z_j + 1), \ 1 \leqslant j \leqslant t, \quad z_{t+1} = l.$$

These transformations are not bijective mappings. One can create composite quasigroup transformations M by composition of different A and/or RA transformations with different leaders.

Quasigroup string transformations $F_i, G_i, \ i = 1, 2, 3$, defined by 3-ary quasigroup (Q, f) are given in [117]. The operations F_1, F_2, F_3 are defined by f, while G_i are defined by f_i. By (5) all transformations are permutations and F_i has inverse G_i. Here we present the definitions of F_1 and G_1. Let take leaders $a_1, a_2, a_3, a_4 \in Q$, and define

$$F_1(x_1x_2 \dots x_t) = (z_1z_2 \dots z_t) \Leftrightarrow z_j = \begin{cases} f(x_1, a_1, a_2), & j = 1 \\ f(x_2, a_3, a_4), & j = 2 \\ f(x_j, z_{j-2}, z_{j-1}), & j > 2, \end{cases}$$

$$G_1(x_1x_2 \dots x_t) = (z_1z_2 \dots z_t) \Leftrightarrow z_j = \begin{cases} f_1(x_1, a_1, a_2), & j = 1 \\ f_1(x_2, a_3, a_4), & j = 2 \\ f_1(x_j, x_{j-2}, x_{j-1}), & j > 2. \end{cases}$$

4. Crypto primitives based on quasigroups

In this section we will consider several designs of cryptographic primitives based on quasigroups, i.e., on different kinds of quasigroup transformations. We emphasize that for getting suitable cryptographic properties of the designs, we have to choose the used quasigroups very carefully. One most desirable property of the quasigroup is its shapelessness [55]. This means that the quasigroup $(Q, *)$ should not be associative, commutative, idempotent, have (left,right) unit, it should not have proper subquasigroups and it should not satisfies identities of kind

$$(((y * \underbrace{x) * x) * \dots) * x}_{k} = y, \quad \underbrace{x * (x * \dots (x * (x}_{k} * y))) = y$$

for some $k < 2n$, where $n = |Q|$. More complete definition of a shapeless quasigroup is given in [82], and several construction of huge shapeless quasigroups are given in [111].

According to the properties satisfied by quasigroups, the set of quasigroups $\mathbf{Q}_n$ of fix order n is classified in several classes. Thus, $\mathbf{Q}_n$ may consists of two disjoint classes, the class of fractal and the class of non-fractal quasigroups ([34],[82]). By considering growing the periods of the strings ${e_{l,*}}^t(\alpha)$ of a periodic string α, $\mathbf{Q}_n$ can be classified again in two disjoint classes, the class of exponential and the class of linear quasigroups. A quasigroup is said to be exponential if the period of the string ${e_{l,*}}^t(\alpha)$ is down bounded by an exponential function $const \cdot 2^{at}$, where $const$ and a are positive constant ([33],[80],[87]). We note that for some quasigroups the constant a is enough big, so they can be used to produce suitable crypto primitives.

In the subsequent section we discus constructions based on quasigroups of several crypto primitives.

4.1. S-boxes defined by quaisgroups

The main point of security in symmetric cryptography in almost all modern block ciphers are the substitution boxes (S-boxes). S-boxes have to confuse the input data into the cipher. Since S-boxes contain a small amount of data, the construction of an S-box should be made very carefully in order the needed cryptographic properties to be satisfied. It is especially important when ultra-lightweight block cipher are designed, like PRESENT ([14]). PRESENT S-boxes are derived as a result of an exhaustive search of all 16! bijective 4-bit S-boxes. Then 16 different classes are obtained and all S-boxes in these classes are optimalwith respect to linear and differential properties.

Instead of an exhaustive search of all 16! bijections of 16 elements as it was done for the design of PRESENT, quasigroups of order 4 can be applied for construction of cryptographically strong S-boxes, called Q-S-boxes [103].

There is no formal definition for S-boxes, they are usually defined as lookup tables that are interpreted as vector valued Boolean functions or Boolean maps $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^q$, where $\mathbb{F}_2$ is a Galois field with two elements. Defined as mappings, for S-boxes so called linearity and differential potential can be computed and correspondingly resistance against linear and differential attacks can be measured.

We already mentioned that quasigroups of order 2^n have vector valued representation. For example, the next quasigroup of order 4

$*$	0	1	2	3
0	0	1	3	2
1	1	0	2	3
2	2	3	0	1
3	3	2	1	0

has representation with the following pair of Boolean functions

$$f(x_0, x_1, y_0, y_1) \;=\; (x_0 + y_0,\;\; x_1 + y_0 + y_1 + x_0 y_0).$$

The algebraic degree of this quasigroup is 2, since the Boolean function $f_2(x_0, x_1, y_0, y_1) = x_1 + y_0 + y_1 + x_0 y_0$ has degree 2. Generally, the quasigroups of order

4 can have algebraic degree 1 (144 of them, so called linear) and 2 (432 of them, so called nonlinear), [44]. Only nonlinear quasigroups are used for construction of suitable S-boxes, i.e., Q-S-boxes. Note that quasigroups of order 4 are 4×2-bit S-boxes.

We want to generate 4×4-bit cryptographically strong S-boxes by using quasigroups of order 4. One criterion for good S-box is to have highest possible algebraic degree, so we search for 4×4-bit S-boxes that have algebraic degree 3 for all output bits. For obtaining 4×4-bit S-boxes, e-transformations will be used to raise the algebraic degree of the produced final bijections. As it is shown in Figure 4, one non-linear quasigroup of order 4 and at least 4 e-transformations will be used to reach the desired degree of 3 for all the bits in final output block.

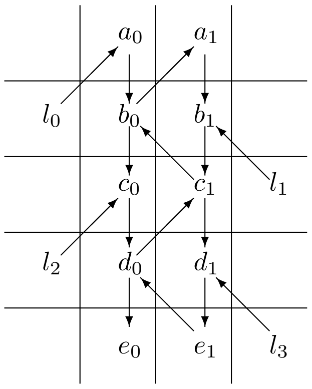


Figure 4: Four e-transformations that bijectively transforms 4 bits into 4 bits by a quasigroup of order 4.

So we can get a Q-S-box that satisfy one condition, to be of degree 3 for all output bits. If the other conditions are satisfied (linearity, differential potential, ...) we will put it in the set of optimal Q-S-boxes. The algorithm for this methodology is given in Table 3. We mention that the minimum number of rounds (iterations) is 4, and using the described methodology we can generate Q-S-boxes in different ways depending on the number of rounds and the number of leaders that we can choose. In our investigation we choose to work with 2, 4 and 8 different leaders and 4 and 8 rounds, respectively. We found all the Q-S-boxes that fulfill the predetermined criteria to be optimal.

Many experiments were made with 2, 4 and 8 different leaders and 4 and 8 rounds, respectively. The obtained results are given in the Table 4.

Some representative of optimal Q-S-boxes are given in Table 5.

An iterative method for construction of Q-S-boxes	
Step 1	Take one quasigroup of order 4 from the class of non-linear;
Step 2	Input the number of rounds;
Step 3	Input the leaders. Usually, their number is the same as the number of rounds;
Step 4	Generate all possible input blocks of 4 bits in the lexicographic ordering (they are 2^4);
Step 5	Take input blocks one by one, and for each of them:
Step 5.1	Apply e-transformation with leader l on the input block;
Step 5.2	Reverse the result from above and apply e-transformation with other leader l again;
Step 5.3	Continue this routine as many times as there is a number of rounds;
Step 5.4	Save the 4-bit result from the last round;
Step 6	At the end concatenate all saved results which generate permutation of order 16 or 4×4-bit Q-S-box;
Step 7	Investigate predetermined criteria;
Step 7.1	If the Q-S-box satisfies criteria, put it in the set of optimal S-boxes;
Step 7.2	If not, go to Step 3;
Step 8	Analyze the optimal set of newly obtained Q-S-boxes;

Table 3: Construction of one Q-S-box

Number of Leaders	Number of Rounds	Number of Optimal boxes
2	4	1 152
4	4	9 216
8	8	331 264

Table 4: The number of optimal Q-S-boxes under different parameters

4.2. Block ciphers

Block cipher is an enciphering method that encrypt a block M of plaintext of length n into a block C of ciphertext of length n, by using a secret key K. It uses an encryption function $E : \mathcal{P} \times \mathcal{K} \to \mathcal{C}$ and a decryption function $D : \mathcal{C} \times \mathcal{K} \to \mathcal{P}$, where $\mathcal{P}, \mathcal{C}$ and $\mathcal{K}$ are the spaces of plaintext, ciphertext and keys; usually $\mathcal{P} = \mathcal{C} = \{0,1\}^n$, $\mathcal{K} = \{0,1\}^k$. The functions $E(M,K)$ and $D(C,K)$ are permutation for fixed K and $D(E(M,K),K) = M$, and there are no different keys K_1, K_2 such that $E(M,K_1) = E(M,K_2)$. Note that when $\mathcal{P} = \mathcal{C} = \mathcal{K} = \{0,1\}^n$, then E is a quasigroup operation with parastrophe D. Besides the last property, there are no many block ciphers based on quasigroup. Here we show the design of the block

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	C	1	2	E	F	9	3	4	8	0	A	B	7	D	6	5

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	D	9	F	C	B	5	7	6	3	8	E	2	0	1	4	A

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	D	9	F	C	B	5	7	6	3	8	E	2	0	1	4	A

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	5	E	6	D	7	4	2	A	8	C	0	9	1	B	F	3

Table 5: Examples of optimal Q-S-boxes given in its hexadecimal notation

cipher BCMPQ (Block Cipher Defined by Matrix Presentation of Quasigroups), [83].

The design of BCDMPQ uses matrix presentation of quasigroups of order 4. Thus, given a quasigroup $(Q, *)$ of order 4, for all $x, y \in Q, x = (x_1, x_2), y = (y_1, y_2)$, x_i, y_i are bits:

$$x * y = m^T + Ax^T + By^T + CAx^T \circ CBy^T \tag{15}$$

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, $m = [m_1, m_2]$ is a Boolean vector and $C = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$. The operation "$\circ$" denotes the component wise product of two vectors.

There are 144 quasigroups of form (15). Out of them, a list of 128 is chosen and stored in memory as follows:

$$seq\_num \qquad m_1, m_2, a_{11}, a_{12}, a_{21}, a_{22}, b_{11}, b_{12}, b_{21}, b_{22} \tag{16}$$

where *seq_num* is a seven bit number (the number of the quasigroup in the list) 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 (15) of the quasigroup operation. (Note that a quasigroup of order 4 is given by using only ten bits, while 32 bits are needed for its Latin square.)

The encryption and decryption algorithms use 16 quasigroups: $Q_1, Q_2, \ldots, Q_8$, $T_1, \ldots, T_8$ in different steps. These matrices are determined by using the round key *key*, which is generated out of the secret key K 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, actually the value of *sequence_number*)
- 56 bits for the quasigroups $T_1, T_2, ..., T_8$ (7 bits per each quasigroup)

The design of this block cipher is based on three algorithms: round key generation, encryption and decryption.

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 shapeless quasigroup Q and a fixed leader $l = 0 = [0,0]$. The round key is obtained by e-transformations. The procedure for generation a round key is described in the RoundKeyGeneration Algorithm. (There, and in the next two algorithms, auxiliary variables are used, tmp is two bits variable and l_{tmp} is one bit variable.)

```
RoundKeyGeneration Algorithm
Input: The secret key $K = K_1K_2 \dots K_{128}$, $K_i$ are bits.
Output: The round key $key = k_1k_2 \dots k_{128}$, $k_i$ are bits.
Initialization: $(Q,*)$ is a fixed matrix quasigroup of order 4
such that $a * a \neq a$ for each $a \in Q$, $l = (0,0)$ is a two bit leader.
----------------------------------------------------------------
for $i = 1$ to 128 do
$k_i \leftarrow K_i$;
for $i = 1$ to 4 do
    $l_{tmp} \leftarrow l$;
    for $j = 1$ to 127 step 2 do
        $tmp \leftarrow (k_j, k_{j+1})$;
        $(k_j, k_{j+1}) = m^T + Al_{tmp}^T + Btmp^T + CAl_{tmp}^T \circ CBtmp^T$;
        $l_{tmp} \leftarrow (k_j, k_{j+1})$;
    $l_{tmp} \leftarrow l$;
    for $j = 128$ to 2 step 2 do
        $tmp \leftarrow (k_{j-1}, k_j)$;
        $(k_{j-1}, k_j) = m^T + Al_{tmp}^T + Btmp^T + CAl_{tmp}^T \circ CBtmp^T$;
        $l_{tmp} \leftarrow (k_{j-1}, k_j)$;
```

The message block length of BCDMPQ can be $8n$ for any n, but we take that $n = 8$, i.e., we consider the light version of the cipher. So, the plaintext message should be split into blocks of 64 bits. Afterwards, the Encryption 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 $Q_1, Q_2, ..., Q_8$ and in the second the matrices $T_1, T_2, ..., T_8$.

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 the Encryption Algorithm.

Encryption Algorithm

Input: The round key $key = k_1k_2\dots k_{128}$, k_i are bits, the plaintext message $a = a_1a_2\dots a_{64}$, a_i are bits.
Output: The ciphertext message $c = c_1c_2\dots c_{64}$.
Initialization: Put $l_i = (k_{2i-1}, k_{2i})$ for $i = 1, 2, \dots, 8$.

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

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

```
for $i = 1$ to 8 do
    $l_{tmp} \leftarrow l_i$;
    for $j = 1$ to 7 step 2 do
        $tmp \leftarrow (a_j, a_{j+1})$;
        $(c_j, c_{j+1}) = m_{Q_i}^T + A_{Q_i} l_{tmp}^T + B_{Q_I} tmp^T$
              $+ CA_{Q_i} l_{tmp}^T \circ CB_{Q_i} tmp^T$;
        $l_{tmp} \leftarrow (c_j, c_{j+1})$;
for $i = 1$ to 4 do
    $l_{tmp} \leftarrow l_i$;
    for $j = 1$ to 63 step 2 do
        $tmp \leftarrow (c_j, c_{j+1})$;
        $(c_j, c_{j+1}) = m_{T_i}^T + A_{T_i} l_{tmp}^T + B_{T_i} tmp^T + CA_{T_i} l_{tmp}^T \circ CB_{T_i} tmp^T$;
        $l_{tmp} \leftarrow (c_j, c_{j+1})$;
    $l_{tmp} \leftarrow l_{i+4}$;
    for $j = 64$ to 2 step 2 do
        $tmp \leftarrow (c_{j-1}, c_j)$;
        $(c_{j-1}, c_j) = m_{T_{i+4}}^T + A_{T_{i+4}} l_{tmp}^T + B_{T_{i+4}} tmp^T +$
                     $CA_{T_{i+4}} l_{tmp}^T \circ CB_{T_{i+4}} tmp^T$;
        $l_{tmp} \leftarrow (c_{j-1}, c_j)$;
```

For decryption purposes we use parastrophe $(Q, \backslash)$ of quasigroup $(Q, *)$. If $x * y = z$, then recall that $y = x \backslash z$ has matrix representation

Decryption Algorithm

Input: The round key $key = k_1k_2\dots k_{128},\ k_i$ are bits,
the ciphertext message $c = c_1c_2\dots c_{64},\ c_i$ are bits.
Output: The plaintext message $a = a_1a_2\dots a_{64}$.
Initialization: Put $l_i = (k_{2i-1}, k_{2i})$ for $i = 1, 2, \dots, 8$.

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

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

```
for $i = 1$ to 64 do  $a_i \leftarrow c_i$;
for $i = 1$ to 4 do
    $l_{tmp} \leftarrow l_{i+4}$;
    for $j = 64$ to 2 step 2 do
        $tmp \leftarrow (a_{j-1}, a_j)$;
        $(a_{j-1}, a_j) = B^{-1}_{T_{i+4}} m^T_{T_{i+4}} + B^{-1}_{T_{i+4}}(I + C)A_{T_{i+4}} l^T_{tmp} +$
        $B^{-1}_{T_{i+4}}(Cm^T_{T_{i+4}} \circ CA_{T_{i+4}} l^T_{tmp}) + B^{-1}_{T_{i+4}} tmp^T +$
        $B^{-1}_{T_{i+4}}(CA_{T_{i+4}} l^T_{tmp} \circ Ctmp^T)$;
        $l_{tmp} \leftarrow (a_{j-1}, a_j)$;
    $l_{tmp} \leftarrow l_i$;
    for $j = 1$ to 63 step 2 do
        $tmp \leftarrow (a_j, a_{j+1})$;
        $(a_{j-1}, a_j) = B^{-1}_{T_i} m^T_{T_i} + B^{-1}_{T_i}(I + C)A_{T_i} l^T_{tmp} +$
        $B^{-1}_{T_i}(Cm^T_{T_i} \circ CA_{T_i} l^T_{tmp}) + B^{-1}_{T_i} tmp^T +$
        $B^{-1}_{T_i}(CA_{T_i} l^T_{tmp} \circ Ctmp^T)$;
        $l_{tmp} \leftarrow (a_j, a_{j+1})$;
for $i = 1$ to 8 do
    $l_{tmp} \leftarrow l_i$;
    for $j = 1$ to 7 step 2 do
        $tmp \leftarrow (a_j, a_{j+1})$;
        $(a_{j-1}, a_j) = B^{-1}_{Q_i} m^T_{Q_i} + B^{-1}_{Q_i}(I + C)A_{Q_i} l^T_{tmp} +$
        $B^{-1}_{Q_i}(Cm^T_{Q_i} \circ CA_{Q_i} l^T_{tmp}) + B^{-1}_{Q_i} tmp^T + B^{-1}_{Q_i}(CA_{Q_i} l^T_{tmp} \circ Ctmp^T)$;
        $l_{tmp} \leftarrow (a_j, a_{j+1})$;
```

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

So, what we actually need to do to decrypt is to start from the ciphertext and reverse the e-transformation, using the quasigroups $T_8, T_7, ..., T_1$ sequentially at first, and then reverse the e-transformations of the mini-blocks (from the encryption algorithm) using the quasigroups $Q_8, Q_7, ..., Q_1$. This can be done using the inverse operation we mentioned shortly before. The decryption of a ciphertext $c_1c_2\dots c_{64}$ is done by the Decryption Algoritam.

Period $q = 2.66$					Period $q = 2.48$					Period $q = 2.43$					Period $q = 2.37$				
$\bullet_0$	0	1	2	3	$\bullet_1$	0	1	2	3	$\bullet_2$	0	1	2	3	$\bullet_3$	0	1	2	3
0	0	2	1	3	0	1	3	0	2	0	2	1	0	3	0	3	2	1	0
1	2	1	3	0	1	0	1	2	3	1	1	2	3	0	1	1	0	3	2
2	1	3	0	2	2	2	0	3	1	2	3	0	2	1	2	0	3	2	1
3	3	0	2	1	3	3	2	1	0	3	0	3	1	2	3	2	1	0	3

Table 6: Quaigroups used in the design of Edon80

For the cipher BCDMPQ only preliminary security investigations were done. The avalanche effect and propagation of one bit and two bits changes were considered and satisfactory results were obtained. It is an open research problem to check the resistance on the other block cipher attacks.

4.3. Stream ciphers

Stream ciphers are classified mainly as synchronous (when the keystream is generated independently of plaintext and cyphertext) and asynchronous (when the keystream is generated by the key and a fixed number of previous ciphertext symbols). A synchronous stream cipher is binary additive when the alphabet consists of binary digits and the output function is the XORing of the keystream and the plaintext. Also, totally asynchronous stream cipher is defined (when the keystream is generated by the key and all previous ciphertext symbols). There are several designs of stream cipher based on quasigroups, and here we will consider two of them.

4.3.1. Edon80

Edon80 is a binary additive stream cipher that is an unbroken eSTREAM finalists [53]. Schematic and behavioral description of Edon80 is given on the Figure 5. Edon80 works in three possible modes:

1) *KeySetup*,
2) *IVSetup* and
3) *Keystream* mode.

For its proper work Edon80 beside the core (that will be described later) has the following additional resources:

1. One register *Key* of 80 bits to store the actual secret key,
2. One register *IV* of 80 bits to store padded initialization vector,
3. One internal 2-bit counter *Counter* as a feeder of Edon80 Core in Keystream mode,
4. One 7 bit SetupCounter that is used in IVSetup mode,
5. One $4 \times 4 = 16$ bytes ROM bank where 4 quasigroups (i.e., Latin squares) of order 4, indexed from $(Q, \bullet_0)$ to $(Q, \bullet_3)$, are stored.

Those 4 predefined quasigroups are described in Table 6.

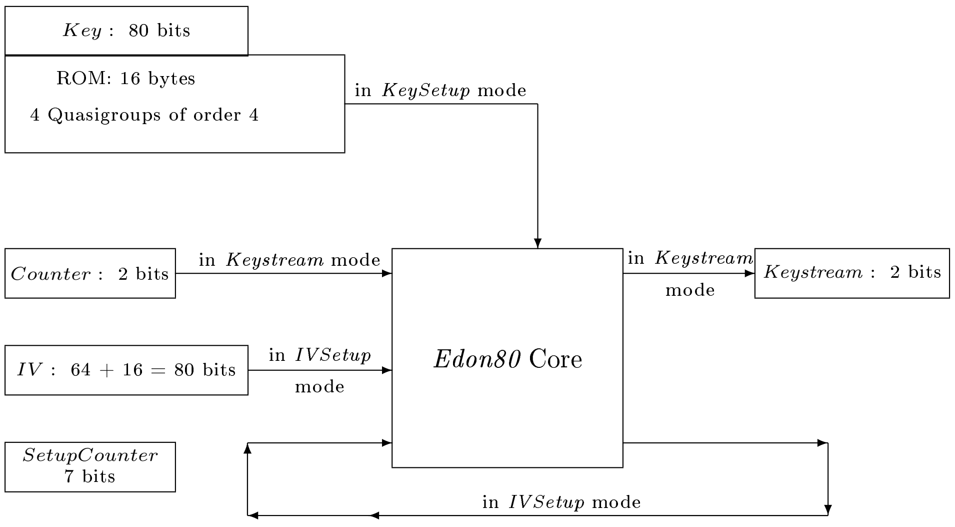


Figure 5: *Edon80* components and their relations.

The structure of the *Edon80* Core is described in the next two figures. The internal structure of *Edon80* can be seen as pipelined architecture of 80 simple 2-bit transformers called e-transformers. The schematic view of a single e-transformer is shown on Figure 6.

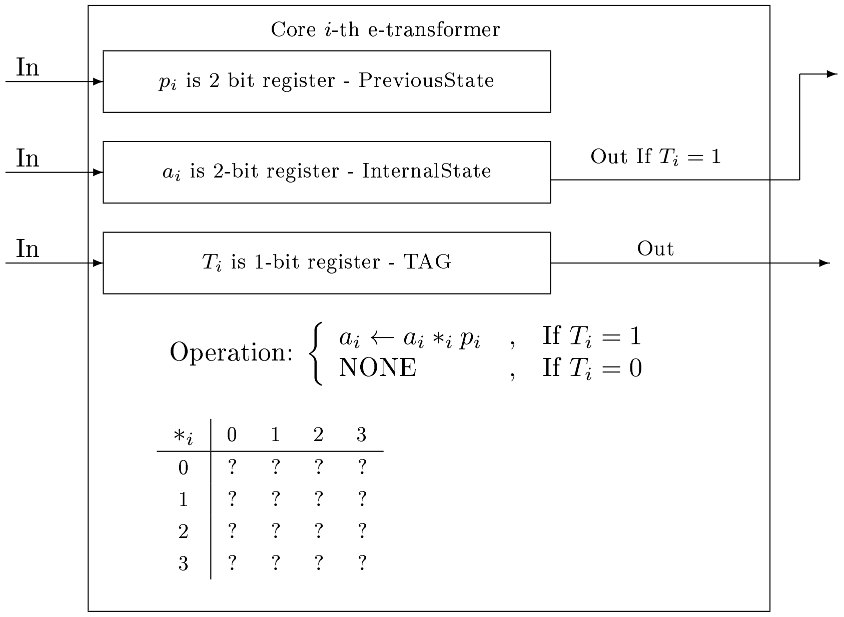
The structure that performs the operation $*_i$ in e-transformers is a quasigroup operation of order 4. We refer an e-transformer by its quasigroup operation $*_i$. So, in Edon80 we have 80 of this e-transformers, cascaded in a pipeline, one feeding another. The Figure 7 shows the pipelined core of *Edon80*.

We will not discuss in all details Edon80. What we want to emphasize is that the chosen quasigroups have enough big periods of growths. Thus, if any of the quasigroups is used k times in an e-transformations, the period of the obtained string will be correspondingly 2.66^k, 2.48^k, 2.43^k, 2.37^k. (Note that $2.48^{80} \approx 2^{104.8}$.) We have to state that 64 out of 576 quasigroups of order 4 have so big periods of growth, any 4 of them could be taken in the construction of Edon80.

Edon80 shows that, when adequately designed, the quasigroups of very small order can produce crypto primitives of high quality.

4.3.2. Edon X, Y, Z

Here we present a design of three different kinds of stream ciphers: the synchronous stream cipher EdonX, the asynchronous stream cipher EdonY and the totaly asynchronous stream cipher EdonZ.


Core i-th e-transformer
In
In
In
p_i is 2 bit register - PreviousState
a_i is 2-bit register - InternalState
T_i is 1-bit register - TAG
Out If $T_i = 1$
Out
Operation: $\begin{cases} a_i \leftarrow a_i *_i p_i & , \text{ If } T_i = 1 \\ \text{NONE} & , \text{ If } T_i = 0 \end{cases}$

$*_i$	0	1	2	3
0	?	?	?	?
1	?	?	?	?
2	?	?	?	?
3	?	?	?	?

Figure 6: Schematic representation of a single e-transformer of *Edon80*.

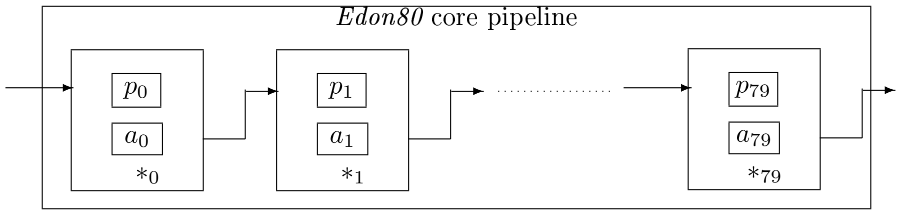


Figure 7: Edon80 core of 80 pipelined e-transformers.

All 3 ciphers EdonX,Y,Z have a same *Initialization phase*, and it is a very important phase of their designs. We denote by K_{in} secretly shared initial key an it is transformed in the initial phase to the working key K. The keys, as well as the messages, consist of s-bit words of any desired length $s \geqslant 4$. The used quasigroups are defined on the set of all s-bit words, and they have 2^s elements. The length n of the initial key (in s-bit words) can be any positive integer, larger n for higher security. This flexibility of the choice of the key length is one of the important performances of this Edon family.

The initialization phase of Edon family is described by following algorithm, where from the secret key K_{in} and the public quasigroup $(Q, \bullet)$ is obtained as output a secret working key K and a secret working quasigroup $(Q, *)$, that is an isotope of $(Q, \bullet)$.

```
Initialization of Edon X,Y,Z family of stream ciphers
Phase 1. Input of initial key
  1. Input: an integer $s$ – the length of the words, an integer $n$ – the initial length of the
     secret key, an integer $m$ – the length of the working key, a quasigroup $(Q, \bullet)$ of order $2^s$
     and the initial secret value of the key $K_{in} = K_0||K_1|| \cdots ||K_{n-1}$   ($K_i$ are s-bit words)
Phase 2. Padding the key
  2. Set $K := K_{in}||n_1||n_2$,        where $n_1$ is the most significant and $n_2$ is the least
     significant $s$-bit word of $n$.
Phase 3. Expanding the key to 512 s-bit words
  3. Set $K_{ex} := K||K|| \cdots ||K||K'$,        where $K'$ consists of the first $l$ s-bits words of $K$
     such that the total length of $K_{ex}$ is 512 s-bits words.
Phase 4. Transformation of $K_{ex}$ with the given quasigroup $(Q, \bullet)$ of order $2^s$
  4. For $i = 0$ to 511 do
      begin
        Set $leader := K[i \bmod (n+2)]$;
        $K_{ex} \leftarrow e_{leader,\bullet}(K_{ex})$;
        $K_{ex} \leftarrow RotateLeft(K_{ex})$;
      end;
Phase 5. Transformation $(Q, *) \leftarrow Isotope(Q, \bullet)$
  5. $(Q, *) \leftarrow (Q, \bullet)$;
     For $i = 0$ to 511 step 8 do
      begin
        Set $row_1 := K_{ex}[i]$; Set $row_2 = K_{ex}[i+1]$;
        $(Q, *) \leftarrow SwapRows(Q, row_1, row_2)$;
        Set $column_1 := K_{ex}[i+2]$; Set $column_2 = K_{ex}[i+3]$;
        $(Q, *) \leftarrow SwapColumns(Q, column_1, column_2)$;
        Set $\gamma := (K_{ex}[i+4], K_{ex}[i+6])$;
        $(Q, *) \leftarrow \gamma(Q, *)$;
      end;
Phase 6. Setting the working key $K = K_0|| \cdots ||K_{m-1}$ (the last $m$ s-bits words of $K_{ex}$)
  6. Set $K = K_0||K_1|| \cdots ||K_{m-1} := K_{ex}[512-m]|| \cdots ||K_{ex}[511]$
```

In the above algorithm K_{ex} means expanded key (it is an auxiliary variable) and the symbol $||$ means concatenation of s-bit words. The notation $K_{in}[j]$ ($K_{ex}[j]$, $K[j]$) means the j-th s-bit words of the K_{in} (K_{ex}, K). Thus, $K[j]$ and K_j have the same meaning. The function $RotateLeft(K_{ex})$ cyclicly rotates the values of the K_{ex} such that $K_{ex}[i] \leftarrow K_{ex}[i+1], i = 0, 1, 2, \ldots, 510$ and $K_{ex}[511] \leftarrow K_{ex}[0]$. The name of the functions *SwapRows* and *SwapColumns* speaks for themselves - they are functions by which the rows or columns of a quasigroup (i.e., the Latin square) are swapped.

EdonX

EdonX operates on nibbles, i.e., on 4-bit variables and consequently it uses quasigroups $Q = \{0, 1, \ldots, 15\}$ of order 16 for doing quasigroup transformations on the streams of data. The working key K is stored in $m \geqslant 64$ internal variables K_i, i.e., $K = K_0 K_1 \ldots K_{m-1}$ and $K_i \in Q$.

The secret key $K_{in} = K_{in}[0]K_{in}[1] \ldots K_{in}[n-1]$, $K_{in}[j] \in Q$ of length n, $32 \leqslant n \leqslant 256$, and an initial public quasigroup $(Q, \bullet)$ of order 16. The decryption function of EdonX is the same as the encryption function. The encryption/decryption function of EdonX uses also two auxiliary 4-bit variables T and X, and one additional integer variable *Counter*. The operation $\oplus$ is the bitwise XOR operation on nibbles.

```
*EdonX* encryption and decryption function
**Phase 1. Initialization**
  From the secret initial key $K_{in}$ of length $n$
  and the initial quasigroup $(Q, \bullet)$ obtain
  new working key $K$ of length $m$ and new quasigroup
  $(Q, *) \leftarrow Isotope(Q, \bullet)$.
**Phase 2. En(De)cryption**
  1. $Counter \leftarrow 0$; $p = \lfloor \mathbb{F}racm2 \rfloor$;
  2. $X \leftarrow K[Counter \texttt{ mod } n]$;
  3. $T \leftarrow K[Counter + p \texttt{ mod } n]$;
  4. For $i = 0$ to $m - 1$ do
       begin
         $X \leftarrow K_i * X$;
         $T \leftarrow T \bullet X$;
         $K_i \leftarrow X$;
       end;
       $K_{m-1} \leftarrow T$;
  5. Output: $X \oplus Inputnibble$;
  6. $Counter \leftarrow Counter + 1$;
  7. Go to 2;
```

It is shown that EdonX is resistant to chosen plaintex/ciphertext attacks. In order to proof that theorems of this type are proved:

Theorem 7. *Any quasigroup* $(Q, *)$ *of order* 16, *where* $Q = \{0, 1, 2, \ldots, 15\}$, *is a solution of the system of functional equations*

$$\begin{aligned}
&x_0 = y_0 * y_{i \texttt{ mod } m}\\
&x_1 = y_1 * x_0\\
&x_2 = y_2 * x_1\\
&\cdots\\
&x_{m-2} = y_{m-2} * x_{m-3}\\
&a = y_{m-1} * x_{m-2}\\
&z = ((\ldots(y_{i+p \texttt{ mod } m} \bullet x_0) \bullet x_1) \bullet \cdots) \bullet x_{m-2}) \bullet a
\end{aligned} \tag{17}$$

with one unknown quasigroup operation $*$ *and unknown variables* $x_0,\ x_1, \dots,$ $x_{m-2},\ y_0,\ y_1, \dots,\ y_{m-1},\ z$ *over* Q*, where* $\bullet$ *is given quasigroup operation on* Q*,* $a \in Q$ *is fixed element,* i *is a nonnegative integer and* $p = \lfloor \frac{m}{2} \rfloor$*.*

EdonX can be used as secure pseudo-random number generator like any synchronous stream cipher. For that aim take the message $M = 000\dots$ to consist of zeros only and let us analyze the output string $C = C_0C_1C_2\dots$. Since $C_i = X_i \oplus 0 = X_i$ $(i = 0, 1, 2, \dots)$, the output string C in this case consists of the values of the variable X.

From the encryption/decryption algorithm of EdonX the following system of iterative functions can be obtained:

$$\begin{array}{l} K_{\lambda,0} = K_{\lambda-1,0} * K_{\lambda-1,i \bmod m} \\ K_{\lambda,1} = K_{\lambda-1,1} * K_{\lambda,0} \\ \dots\dots\dots \\ K_{\lambda,m-2} = K_{\lambda-1,m-2} * K_{\lambda,m-3} \\ X_{\lambda,m-1} = K_{\lambda-1,m-1} * K_{\lambda,m-2} \\ K_{\lambda,m-1} = ((\dots(K_{\lambda-1,i+p \bmod m} \bullet K_{\lambda,0}) \bullet K_{\lambda,1}) \cdots \bullet K_{\lambda,m-2}) \bullet X_{\lambda,m-1} \end{array} \tag{18}$$

What we are interested for are the values of $X_{\lambda,m-1}$ for $\lambda = 0, 1, \dots,$ since the output string C is just the string $X_{0,m-1}X_{1,m-1}X_{2,m-1}X_{3,m-1}\dots$. "What is the period and the nature of the string C"? The answer depends on the theory of discrete chaos systems, that is not developed yet! In several experiments with a reduced system (18) with initial keys of length 4 and $m = 16$ is obtained that either the ergodic part had length greater than 2^{32} or the periodic part had a period greater than 2^{32} (or both). It is resonable to to conjecture that in the standard version of EdonX (when $m = 64$ and the initial keys have length at least 32) either an ergodic part of length 2^{128} or a period 2^{128} (or both) will be obtained.

EdonY

The proof that EdonY is self-synchronized is a direct consequence of the following theorem.

Theorem 8. *Let* $E = e_{l_1,*_1} \circ \cdots \circ e_{l_n,*_n}$ *and* $D = d_{l_n,\backslash_n} \circ \cdots \circ d_{l_1,\backslash_1}$ *be transformations obtained with* n *quasigroup transformations* $*_1, \dots, *_n$ *on* Q*, leaders* $l_1, \dots, l_n$ *and corresponding parastrophes* $\backslash_1, \dots, \backslash_n$*. Assume that* $E(b_1b_2\dots b_k) = c_1c_2\dots c_k,\ k > n,$ *and* $d \neq c_i$ *for some fixed* i $(b_j, c_j, d \in Q)$*. Then, for some* $d_1, \dots, d_{n+1} \in Q$*,*

$$D(c_1\dots c_{i-1}dc_{i+1}\dots c_k) = \begin{cases} b_1\dots b_{i-1}d_1\dots d_{n+1}b_{i+n+1}\dots b_k, & k > i+n \\ b_1\dots b_{i-1}d_1\dots d_{k-i+1}, & k \leqslant i+n \end{cases}.$$

In the construction of EdonY we use a public quasigroup $(Q, \bullet)$ of order 32 defined on 5-bits letters, $Q = \{0, 1, 2, \dots, 31\}$ and a secret key K_{in} stored in n internal variables $K_i \in Q$, i.e., $K_{in} = K_0K_1\dots K_{n-1}$ and $n \geqslant 32$.

The EdonY encryption algorithm and decryption algorithm are precisely defined by the following procedures, where $M = M_0M_1M_2M_3M_4\dots$ $(C = C_0C_1C_2C_3C_4\dots)$ is the input plaintext (output ciphertext) string. The variables X and Y in the decryption algorithm are auxiliary 5-bits variables.

```
EdonY encryption algorithm
Phase 1. Initialization
  From the secret initial key $K_{in}$ of length $n$
  and the initial quasigroup $(Q,\bullet)$ obtain
  new working key $K$ of length $m$ and new quasigroup
  $(Q,*) \leftarrow Isotope(Q,\bullet)$.
Phase 2. Encryption
  1. $Counter \leftarrow 0$; $p = \lfloor \mathbb{F}racn2 \rfloor$;
  2. $K_0 \leftarrow K_0 * (M_{Counter} * K_{Counter+p \bmod n})$
  3. For $i = 1$ to $n-1$ do
       begin
         $K_i \leftarrow K_i * K_{i-1}$;
       end;
  4. Output: $C_{Counter} = K_{n-1}$;
  5. $Counter \leftarrow Counter + 1$;
  6. Go to 2;
```

```
EdonY decryption algorithm
Phase 1. Initialization
  From the secret initial key $K_{in}$ of length $n$
  and the initial quasigroup $(Q,\bullet)$ obtain
  new working key $K$ of length $m$ and new quasigroup
  $(Q,*) \leftarrow Isotope(Q,\bullet)$.
Phase 2. Decryption
  1. $Counter \leftarrow 0$; $p = \lfloor \mathbb{F}racn2 \rfloor$;
  2. $X \leftarrow K_{n-1}$
     $K_{n-1} \leftarrow C_{Counter}$;
  3. For $i = n-2$ down to 0 do
       begin
         $Y \leftarrow K_i$
         $K_i \leftarrow X \setminus K_{i+1}$;
         $X \leftarrow Y$
       end;
  4. Output: $M_{Counter} = (X \setminus K_0)/K_{Counter+p \bmod n}$;
  5. $Counter \leftarrow Counter + 1$;
  6. Go to 2;
```

It follows from Theorem 8 that EdonY is self synchronized since one error in the cipher-text C will propagate $n+1$ errors in the recovered plaintext M', i.e.,

Initialization
From the secret initial key K_{in} of length n and the initial quasigroup $(Q, \bullet)$ obtain new working key K of length 64 and new quasigroup $(Q, *) \leftarrow Isotope(Q, \bullet)$.

Encryption. **Input**: Key K of length n and message M. **Output**: Message C.	**Decryption**. **Input**: Key K of length n and message C. **Output**: Message M.
1) $X \leftarrow InputNibble$; 2) $T \leftarrow 0$; 3) For $i = 0$ to $n - 1$ do $X \leftarrow K_i * X]$; $T \leftarrow T \oplus X$; $K_i \leftarrow X$; 4) $K_{n-1} \leftarrow T$; 5) Output X; 6) Go to 1;	1) $X, T \leftarrow InputNibble$; 2) $temp \leftarrow K_{n-1}$; 3) For $i = n - 1$ downto 0 do $X \leftarrow temp \setminus X$; $T \leftarrow T \oplus X$; $temp \leftarrow K_{i-1}$; $K_{i-1} \leftarrow X$; 4) $K_{n-1} \leftarrow T$; 5) Output X; 6) Go to 1;

Table 7: Totaly Asynchronous Stream Cipher

the original message M and M' will differ in $n + 1$ consecutive letters. If there will be a string of errors in C of length r, then the recovered plaintext will have $r + n$ errors.

There are proofs that EdonY is resistent to dictionary and to chosen plaintext/ciphertext attacks. Considering only the known ciphertext attacks, the resistance follows from the next theorem.

Theorem 9. *Given a ciphertext C, for each quasigroup operation $*$ on $Q = \{0, 1, \ldots, 31\}$ and each key $K = K_0K_1 \ldots K_{n-1}$ there is a plaintext M such that C is its ciphertext.*

EdonZ

EdonZ operates on nibbles, so it uses a quasigroup $(Q, \bullet)$, $Q = \{0, \ldots, 15\}$, of order 16. The secret key K_{in} is stored in $n = 64$ internal variables K_i, that have values in the range $Q = \{0, 1, \ldots, 15\}$.

EdonZ encryption and decryption algorithms use also temporal 4-bit variables T, X, and *temp*. EdonZ differs from the synchronous EdonX in the way how the initial value of the variables X and T are set and how the final computation of X is done. However, in decrypting algorithm EdonX does not use the left parastrophe of the $(Q, *)$ since it is binary additive stream cipher, but EdonZ needs $(Q, \setminus)$.

Next we will give an example that will work on the principles of EdonZ, but for the simplicity of the explanation, quasigroup of order 4 will be used and the working key will be of length 4. We take that the working key is $K = 2\ 3\ 2\ 3$, the message is $M = \{0, 0, 1, 0, 2, 3, 0, \dots\}$ and the quasigroup and its parastrophe are the following.

$*$	0	1	2	3
0	3	0	2	1
1	1	2	0	3
2	0	3	1	2
3	2	1	3	0

$\backslash$	0	1	2	3
0	1	3	2	0
1	2	0	1	3
2	0	2	3	1
3	3	1	0	2

Several steps of EdonZ encryption are as following.

		M_0			M_1			M_2			M_3		
	K	X	T	K	X	T	K	X	T	K	X	T	$\cdots$
i		0	0		0	0		1	0		0	0	$\cdots$
0	2	0	0	0	3	3	3	1	1	1	1	1	
1	3	2	2	2	2	1	2	3	2	3	1	0	
2	2	1	3	1	0	1	0	1	3	1	2	2	
3	3	1	2	2	0	1	1	2	1	1	0	2	
Output $C = X$		1			0			2			0		

We emphasize that EdonZ is used in the definition of the random error-correcting code RCBQ with cryptograpic properties ([55], [119]).

4.4. Pseudo random number generators

A truly random sequence can be obtained only by theory. Namely, if we take that a sequence is random only if it passes all of the statistical test for randomness, then we can never check if a sequence is random until all of the tests, infinitely many, are passed. So, sequences that look randomly are used in many applications were random sequences are needed. They are produced by some deterministic algorithms or physical phenomenas and are called Pseudo Random Sequences (PRS). PRS have to pass all known approved battery o statistical tests for randomness (like Diehard, NIST, ...) The algorithms for producing PRS are called Pseudo Random Sequence Generator (PRSG), i.e., PRNG when we have number sequences.

Many PRNG that are used for many purposes are biased, for example the next produced bit (or symbol) can be predictable with probability greater than 1/2. Then, the obtained sequence of such a generator should be unbiased. By using quasigroup transformations several type of PRNG can be designed. In fact, all of the previous stream ciphers can be used as PRNG, and they are cryptographically secure, since a key is used. What is a problem with those PRNG is their efficiency, since they are designed for other purposes.

Very simple PRNG can be obtained by the following procedure.

QPRNG can produce pseudo random sequences from very biased sequences, even from periodical sequences as well. We emphasize that in QPRNG the choice of the quasigroup is very important, it should be shapeless and exponential with

```
Quasigroup PRNG (QPRNG)

 Phase I. Initialization
1. Choose a positive integer $s \geqslant 4$;
2. Choose a quasigroup $(A, *)$ of order $s$;
3. Set a positive integer $k$;
4. Set a leader $l$, a fixed element of $A$ such that $l * l \neq l$;
Phase II. Transformations of the random
           string $b_0b_1b_2b_3\dots,\quad b_j \in A$
5. For $i = 1$ to $k$ do $L_i \leftarrow l$;
6. $j \leftarrow 0$;
7. do
      $b \leftarrow b_j$;
      $L_1 \leftarrow L_1 * b$;
      For $i = 2$ to $k$ do $L_i \leftarrow L_i * L_{i-1}$;
      Output: $L_k$;
      $j \leftarrow j + 1$;
   loop;
```

Table 8: Algorithm for simple QPRNG

as higher period of growth as possible. In fact, for quasigroups of order 4 one can compute the period of growth of all 576 quasigroups. The Table 9 shows that suitable quasigroups of order 16 can be find enough easily too.

As an example of the capacity of QPRNG we consider the PRNG used in GNU C v2.03 that do not passed all of the statistical tests in the Diehard Battery v0.2 beta [30], but after using QPRNG on the obtained sequence from GNU with a quasigroup of order 256 and for $k = 1$ (only one application of an e-transformation) all tests of Diehard were passed ([87]).

```
    ***** TEST SUMMARY FOR GNU C (v2.03) PRNG *****
    All p-values:
0.2929,0.8731,0.9113,0.8755,0.4637,0.5503,0.9435,0.7618,0.9990,0.0106,1.0000,0.0430,
1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,
1.0000,1.0000,1.0000,0.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,
. . . . . . . . . . .. . . . . . .
0.7921,0.4110,0.3050,0.8859,0.4783,0.3283,0.4073,0.2646,0.0929,0.6029,0.4634,0.8462,
0.2385,0.6137,0.1815,0.4001,0.1116,0.2328,0.0544,0.4320,0.0000,0.0000,0.0000,0.0000,
. . . . . . . . . . .. . . . . . .
0.0003,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,
0.0753,0.0010,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0233,0.0585,0.0000,0.0000,
0.0000,0.0000,0.0000,0.2195,0.0321,0.0000,0.0000,0.9948,0.0006,0.0000,0.0000,0.0688,
. . . . . . . . . . .. . . . . . .
0.2303,0.1190,0.8802,0.0377,0.6887,0.4175,0.0803,0.3687,0.7010,0.7425,0.1003,0.0400,
0.9488,0.3209,0.5965,0.0676,0.0021,0.2337,0.5204,0.5343,0.0630,0.2008,0.6496,0.4157,
0.9746,0.1388,0.4657,0.5793,0.6455,0.8441,0.5248,0.7962,0.8870

    Overall p-value after applying KStest on 269 p-values = 0.000000
```

Value of c	Number of quasigroups with period growth $2^{c\,k}$	Value of c	Number of quasigroups with period growth $2^{c\,k}$
$0.00 \leqslant c < 0.25$	4	$2.00 \leqslant c < 2.25$	79834
$0.25 \leqslant c < 0.50$	23	$2.25 \leqslant c < 2.50$	128836
$0.50 \leqslant c < 0.75$	194	$2.50 \leqslant c < 2.75$	174974
$0.75 \leqslant c < 1.00$	686	$2.75 \leqslant c < 3.00$	199040
$1.00 \leqslant c < 1.25$	2517	$3.00 \leqslant c < 3.25$	175848
$1.25 \leqslant c < 1.50$	7918	$3.25 \leqslant c < 3.50$	119279
$1.50 \leqslant c < 1.75$	18530	$3.50 \leqslant c < 3.75$	45103
$1.75 \leqslant c < 2.00$	42687	$3.75 \leqslant c \leqslant 4.00$	4527

Table 9: Period growth of 10^6 randomly chosen quasigroups of order 16 after 5 applications of e-transformations (k=5 in QPRNG)

```
    *** TEST SUMMARY FOR GNU C v2.03 + QUASIGROUP PRNG IMPROVER ***
    All p-values:
0.5804,0.3010,0.1509,0.5027,0.3103,0.5479,0.3730,0.9342,0.4373,0.5079,0.0089,0.3715
0.0584,0.1884,0.1148,0.0662,0.8664,0.5070,0.7752,0.1939,0.9568,0.4948,0.1114,0.2042,
0.4883,0.4537,0.0281,0.0503,0.0346,0.6085,0.1596,0.1545,0.0855,0.5665,0.0941,0.7693,
. . . . . . . . . . . .. . . . . . .
0.6544,0.9673,0.8787,0.9520,0.8339,0.4397,0.3687,0.0044,0.7146,0.9782,0.7440,0.3042,
0.8465,0.7123,0.8752,0.8775,0.7552,0.5711,0.3768,0.1390,0.9870,0.9444,0.6101,0.1090,
. . . . . . . . . . . .. . . . . . .
0.8538,0.6871,0.8785,0.9159,0.4128,0.4513,0.1512,0.8808,0.7079,0.2278,0.1400,0.6461,
0.3353,0.1064,0.6739,0.2066,0.5119,0.0558,0.5748,0.5064,0.8982,0.6422,0.7512,0.8633,
0.4625,0.0843,0.0903,0.7641,0.6253,0.8523,0.7768,0.8041,0.5360,0.0826,0.0378,0.8710,
. . . . . . . . . . . .. . . . . . .
0.2115,0.8156,0.8468,0.9429,0.8382,0.1463,0.4212,0.6948,0.4816,0.3454,0.2114,0.3493,
0.3448,0.0413,0.2422,0.6363,0.2340,0.8404,0.0065,0.7319,0.8781,0.2751,0.5197,0.4105,
0.0832,0.1503,0.1148,0.3008,0.0121,0.0029,0.4423,0.6239,0.0651,0.3838,0.0165,0.2770,
0.2074,0.0004,0.7962,0.4750,0.4839,0.9152,0.1681,0.0822,0.0518
Overall p-value after applying KStest on 269 p-values = 0.018449
```

4.5. Hash functions

Hash functions on a set A are mappings $h : A^+ \to A^n$ that take a variable-size input messages and map them into fixed-size output, known as hash result, message digest, hash-code etc. They are used in checking data integrity, digital signature schemes, commitment schemes, password based identification systems, digital timestamping schemes, pseudo-random string generation, key derivation, one-time passwords etc.

The first attempts for using quasigroup transformations for creating cryptographic hash functions do not have actual implementations ([37], [38], [86], [47]). In [140] is proposed a hash function as one elementary e-transformation on the message $x_1x_2\dots x_t$:

$H(x_1x_2\dots x_t) = (((a \bullet x_1) \bullet x_2) \cdots \bullet x_t = e_{a,\bullet}(x_1x_2\dots x_t)$.

The huge working quasigroup $(Q, \bullet)$ is obtained from the modular subtraction quasigroup $(Q, *)$ defined by $x * y = x + (r - y)\ mod\ r,\ \ |Q| = r$, and three secret permutations π, w, ρ as $x \bullet y = \pi^{-1}(w(x) + (r - \rho(y))\ mod\ r)$. The leader a is used as initialization vector.

A generic hash function with quasigroup reverse string transformation R has been described in [54], with first implementation named Edon-R(256, 384, 512) given in [46]. Another interesting application of quasigroups is the quasigroup folding, a 2 time slower security fix of the MD4 family of hash functions [48], with shapeless randomly generated quasigroup of order 16. Similar technique has been used in [49] , where new hash function SHA-1Q2 has been constructed from SHA-1 by message expansion part with quasigroup folding and has only 8 internal iterative steps (and it is 3% faster than SHA-1).

Further on we will consider the candidate of NIST SHA-3 competition, Edon-$\mathcal{R}$ and NaSHA, whose designs were based on huge quasigroup transformations.

Edon-$\mathcal{R}$

Edon-$\mathcal{R}$ [58] is wide-pipe iterative hash function with standard MD-straitening. It was the fastest First round candidate of NIST SHA-3 competition.

The chaining value H_i and the message input M_i for the ith round are composed of two q-bits blocks, $q = 256, 512$, i.e., $H_i = (H_i^1, H_i^2)$ and $M_i = (M_i^1, M_i^2)$, and the new chaining value H_{i+1} is produced as follows

$H_{i+1} = (H_{i+1}^1, H_{i+1}^2) = R(H_i^1, H_i^2, M_i^1, M_i^2)$,

$\mathcal{R}$ is little bit modified reverse string transformation, in a sense that two parts from the message are taken reversed when are used like a leaders, and the order of leaders is $\bar{M}_i^2, H_i^1, H_i^2, \bar{M}_i^1$. The compression function $\mathcal{R}$ uses two huge quasigroups of order 2^{256} and 2^{512}. Algorithmic description of the quasigroup of order 2^{256} is given in the Table . There X_i, Y_i and Z_i are 32-bit variables, so $X = (X_0, X_1, \dots, X_7)$, $Y = (Y_0, Y_1, \dots, Y_7)$ and $Z = (Z_0, Z_1, \dots, Z_7)$ are 256-bits variables. (Note that the operation is $X * Y = Z$.) Operation "+" denotes addition modulo 2^{32} , operation $\oplus$ is the logical operation of bitwise exclusive or and the operation $ROTL^r(X_i)$ is the operation of bit rotation of the 32-bit X_i, to the left for r positions.

NaSHA

NaSHA [92] is another First round candidate to the NIST SHA-3 competition based on quasigroups . It is also wide-pipe iterative hash function with standard MD-straitening. NaSHA-(m, k, r) has three parameters m, k, r, where m denotes message length, k is the number of elementary quasigroup string transformations

Quasigroup operation of order 2^{256}

Input: $X = (X_0, X_1, \dots, X_7)$ and $Y = (Y_0, Y_1, \dots, Y_7)$,
where X_i and Y_i are 32-bit variables.
Output: $Z = (Z_0, Z_1, \dots, Z_7)$ where Z_i are 32-bit variables.
Temporary 32-bit variables: $T_0, \dots, T_{15}$.

```
    $T_0 \leftarrow ROTL^0(0xAAAAAAAA + X_0 + X_1 + X_2 + X_4 + X_7);$
    $T_1 \leftarrow ROTL^4(X_0 + X_1 + X_3 + X_4 + X_7);$
    $T_2 \leftarrow ROTL^8(X_0 + X_1 + X_4 + X_6 + X_7);$
 1. $T_3 \leftarrow ROTL^{13}(X_2 + X_3 + X_5 + X_6 + X_7);$
    $T_4 \leftarrow ROTL_{17}(X_1 + X_2 + X_3 + X_5 + X_6);$
    $T_5 \leftarrow ROTL_{22}(X_0 + X_2 + X_3 + X_4 + X_5);$
    $T_6 \leftarrow ROTL^{24}(X_0 + X_1 + X_5 + X_6 + X_7);$
    $T_7 \leftarrow ROTL^{29}(X_2 + X_3 + X_4 + X_5 + X_6);$

    $T_8 \leftarrow T_3 \oplus T_5 \oplus T_6;$
    $T_9 \leftarrow T_2 \oplus T_5 \oplus T_6;$
    $T_{10} \leftarrow T_2 \oplus T_3 \oplus T_5;$
 2. $T_{11} \leftarrow T_0 \oplus T_1 \oplus T_4;$
    $T_{12} \leftarrow T_0 \oplus T_4 \oplus T_7;$
    $T_{13} \leftarrow T_1 \oplus T_6 \oplus T_7;$
    $T_{14} \leftarrow T_2 \oplus T_3 \oplus T_4;$
    $T_{15} \leftarrow T_0 \oplus T_1 \oplus T_7;$

    $T_0 \leftarrow ROTL^0(0x55555555 + Y_0 + Y_1 + Y_2 + Y_5 + Y_7);$
    $T_1 \leftarrow ROTL^5(Y_0 + Y_1 + Y_3 + Y_4 + Y_6);$
    $T_2 \leftarrow ROTL^9(Y_0 + Y_1 + Y_2 + Y_3 + Y_5);$
 3. $T_3 \leftarrow ROTL_{11}(Y_2 + Y_3 + Y_4 + Y_6 + Y_7);$
    $T_4 \leftarrow ROTL_{15}(Y_0 + Y_1 + Y_3 + Y_4 + Y_5);$
    $T_5 \leftarrow ROTL_{20}(Y_2 + Y_4 + Y_5 + Y_6 + Y_7);$
    $T_6 \leftarrow ROTL_{25}(Y_1 + Y_2 + Y_5 + Y_6 + Y_7);$
    $T_7 \leftarrow ROTL_{27}(Y_0 + Y_3 + Y_4 + Y_6 + Y_7);$

    $Z_5 \leftarrow T_8 + (T_3 \oplus T_4 \oplus T_6);$
    $Z_6 \leftarrow T_9 + (T_2 \oplus T_5 \oplus T_7);$
    $Z_7 \leftarrow T_{10} + (T_4 \oplus T_6 \oplus T_7);$
 4. $Z_0 \leftarrow T_{11} + (T_0 \oplus T_1 \oplus T_5);$
    $Z_1 \leftarrow T_{12} + (T_2 \oplus T_6 \oplus T_7);$
    $Z_2 \leftarrow T_{13} + (T_0 \oplus T_1 \oplus T_3);$
    $Z_3 \leftarrow T_{14} + (T_0 \oplus T_3 \oplus T_4);$
    $Z_4 \leftarrow T_{15} + (T_1 \oplus T_2 \oplus T_5);$
```

Table 10: An algorithmic description of a quasigroup of order 2^{256}.

of type A and RA, and r is from the order 2^{2^r} of used quasigroups. To the competition was sent NaSHA-$(m, 2, 6)$, $m = 224, 256, 384, 512$. Every round consists of one linear transformation obtained from an LFSR, followed by MT quasigroup string transformation, that is a composition of k alternate quasigroup string transformations A and RA. NaSHA uses novel design principle: the quasigroups used in every iteration in compression function are different, and depend on the processed message block. Even in one iteration, different quasigroups are used for two quasigroup transformations. Quasigroups in NaSHA are obtained by using Extended Feistel Networks as orthomorphisms and complete mappings on the groups $(\mathbb{Z}_{2^{16}}, \oplus)$, $(\mathbb{Z}_{2^{32}}, \oplus)$ and $(\mathbb{Z}_{2^{64}}, \oplus)$. NaSHA is of order 2^{64} and is produced from known starting bijection of order 2^8 by using xoring, addition modulo 2^{64} and table lookups.

The MQQ (Multivarite Quadratic Quasigroups) family of crytptosystems was first defined in 2007 [51]. Subsequently, a signature [57], and an improved encryption variant was proposed [59]. As the name suggests, the cryptosystems from this family are based on multivariate quadratic quasigroups – MQQs, defined over a finite field. It belongs to the broader family of multivariate public key cryptosystems ($\mathcal{MQ}$) whose security relies on the hardness of solving quadratic polynomial systems of equations over finite fields, known to be $\mathcal{NP}$-hard problem.

A typical ($\mathcal{MQ}$) public key cryptosystem relies on the knowledge of a trapdoor for a particular system of polynomials over a finite field $\mathbb{F}_q$. The public key of the cryptosystem is usually given by a multivariate quadratic map $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$, i.e.,

$$\mathcal{P}(x_1, \dots, x_n) = \begin{pmatrix} p_1(x_1, \dots, x_n) = \sum\limits_{1 \leqslant i \leqslant j \leqslant n} \widetilde{\gamma}_{ij}^{(1)} x_i x_j + \sum\limits_{i=1}^{n} \widetilde{\beta}_i^{(1)} x_i + \widetilde{\alpha}^{(1)} \\ \vdots \\ p_m(x_1, \dots, x_n) = \sum\limits_{1 \leqslant i \leqslant j \leqslant n} \widetilde{\gamma}_{ij}^{(m)} x_i x_j + \sum\limits_{i=1}^{n} \widetilde{\beta}_i^{(m)} x_i + \widetilde{\alpha}^{(m)} \end{pmatrix}$$

for some coefficients $\widetilde{\gamma}_{ij}^{(s)}, \widetilde{\beta}_i^{(s)}, \widetilde{\alpha}^{(s)} \in \mathbb{F}_q$. It is obtained by obfuscating a structured central map

$$\mathcal{F} : (x_1, \dots, x_n) \in \mathbb{F}_q^n \to \big(f_1(x_1, \dots, x_n), \dots, f_m(x_1, \dots, x_n)\big) \in \mathbb{F}_q^m,$$

using two bijective affine mappings $\mathcal{S}, \mathcal{T}$ over $\mathbb{F}_q^n$ that serve as a sort of mask to hide the structure of $\mathcal{F}$. The public key is defined as

$$\mathcal{P} = \mathcal{T} \circ \mathcal{F} \circ \mathcal{S}.$$

The mappings $\mathcal{S}$ and $\mathcal{T}$ are part of the private key s. Besides them, the private key may also contain other secret parameters that allow creation, but also easy inversion of the transformation $\mathcal{F}$. Without loss of generality, we can assume that the private key is $s = (\mathcal{F}, \mathcal{S}, \mathcal{T})$.

$$\begin{tikzcd}
\text{input } \mathbf{x} \arrow[d] \\
\mathbf{x} = (x_1, \dots, x_n) \arrow[d, "\text{private: } \mathcal{S}"'] \arrow[ddd, bend left=90, "\substack{\textbf{public:} \\ \mathcal{P} = (p_1, \dots, p_n)}"] \\
\mathbf{x}' \arrow[d, "\text{private: } \mathcal{F}"'] \\
\mathbf{y}' \arrow[d, "\text{private: } \mathcal{T}"'] \\
\text{output } \mathbf{y}
\end{tikzcd}$$

Figure 8: A general $\mathcal{MQ}$ trapdoor

Graphically, the trapdoor of an $\mathcal{MQ}$ scheme can be depicted as in Figure 8.

MQQ-SIG is a signature scheme that has excellent performance in signing. In particular, it is the fastest signature scheme in the ECRYPT benchmarking of cryptographic systems (eBACS) [13]. It is defined over $\mathbb{F}_2$ and has the minus modifier applied because of the possibility for direct algebraic attack and MinRank attack otherwise.

The length of the messages that can be signed is $n/2$, and the signing process is performed by prepending a random string of length $n/2$.

A high-level schematic presentation of the signing and verification process is given in Figure 9 and the corresponding algorithmic description in Algorithm 1.

The central mapping $\mathcal{F}$ of MQQ-SIG is a quasigroup string transformation using one quasigroup q. Both $\mathcal{F}$ and the inverse $\mathcal{F}^{-1}$ are depicted in Figure 10 and Figure 11. Algorithm 2 gives a detailed description of the construction of the central map $\mathcal{F}$.

The MQQs used are of relatively small order 2^8 that allows storing them in a lookup table, used for the signing process.

The algorithm used for construction of the MQQ is presented in Algorithm 3.

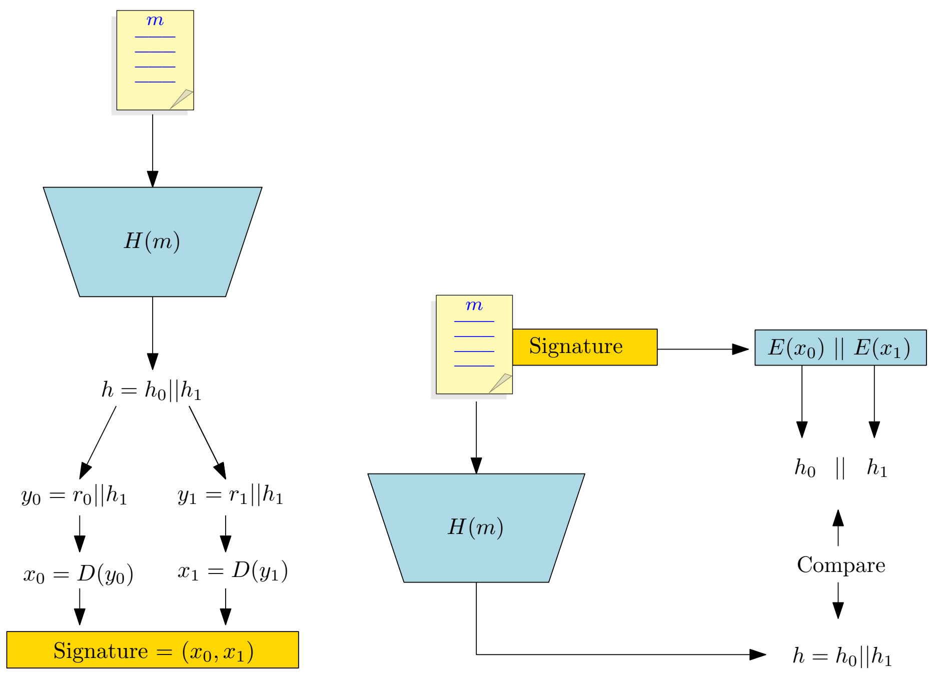


Figure 9: The signing and verification process of MQQ-SIG

$$\begin{tikzcd}
X_1 \arrow[r, "*"] \arrow[d] & X_2 \arrow[d] \arrow[dr] & X_3 \arrow[l, "*"'] \arrow[r, "*"] & \cdots \arrow[dr] & X_{\frac{n}{8}-1} \arrow[l, "*"'] \arrow[r, "*"] & X_{\frac{n}{8}} \arrow[d] \\
Y_1 & Y_2 & Y_3 & \cdots & Y_{\frac{n}{8}-1} & Y_{\frac{n}{8}}
\end{tikzcd}$$

Figure 10: Graphical representation of the central map $\mathcal{F}$ in MQQ-SIG.

$$\begin{tikzcd}
Y_1 \arrow[d] & Y_2 \arrow[d] & Y_3 \arrow[dl, "q_/"'] & \cdots & Y_{n/8-1} \arrow[dl, "q_/"'] & Y_{n/8} \arrow[d] \\
X_1 \arrow[ur, "q_\backslash"] & X_2 \arrow[r] & X_3 \arrow[ur, "q_\backslash"] & \cdots \arrow[r] & X_{n/8-1} \arrow[ur, "q_\backslash"] & X_{n/8}
\end{tikzcd}$$

Figure 11: Graphical representation of the inverse map $\mathcal{F}^{-1}$ in MQQ-SIG, using the right and left parastrophes.

Algorithm 1 MQQ-SIG

Key Generation

1. Use Algorithm 2 to construct the central map $\mathcal{F}$.
2. Construct the affine mappings $\mathcal{S}$ and $\mathcal{S}'$ as defined in [57].
3. Pick a hash function $H : \{0,1\}^* \to \{0,1\}^n$.
4. Construct the mapping $\mathcal{P}' = \mathcal{S} \circ \mathcal{F} \circ \mathcal{S}'$ and define the public key $\mathcal{P}$ as the last $\frac{n}{2}$ coordinates of $\mathcal{P}'$. Denote by $\mathcal{P}'^{-1}$ the inverse of $\mathcal{P}'$. Algorithm 4 is used to compute $\mathcal{P}'^{-1}$.

Output: The public key $\mathcal{P}$ and the private key $(\mathcal{F}, \mathcal{S}, \mathcal{S}')$.

Signature Generation
Input: A message $m \in \{0,1\}^*$ to be signed.

1. Compute $h = h_0 || h_1 \leftarrow H(m)$, where h_0 and h_1 are both $\frac{n}{2}$ bits long.
2. Generate two random $\frac{n}{2}$-bit values, r_0 and r_1, and set $\mathbf{y_0} = r_0 || h_0$ and $\mathbf{y_1} = r_1 || h_1$.
3. Compute $\mathbf{x_0} = \mathcal{P}'^{-1}(\mathbf{y_0})$ and $\mathbf{x_1} = \mathcal{P}'^{-1}(\mathbf{y_1})$.

Output: The digital signature $(\mathbf{x_0}, \mathbf{x_1})$.

Signature Verification
Input: A message-signature pair $(m, (\mathbf{x_0}, \mathbf{x_1}))$.

1. Compute $h = h_0 || h_1 \leftarrow H(m)$.
2. Compute $\mathbf{z_0} \leftarrow \mathcal{P}(\mathbf{x_0})$ and $\mathbf{z_1} \leftarrow \mathcal{P}(\mathbf{x_1})$.

Accept the signature if $\mathbf{z_0} = h_0$ and $\mathbf{z_1} = h_1$, otherwise reject.

Algorithm 2 Construct$\mathcal{F}$

Input: A vector $\mathbf{x} = (x_1, \dots, x_n)$, where n is a multiple of 8.

1. Represent the vector $\mathbf{x}$ as a string $X = X_1X_2 \dots X_k$, where $k = \frac{n}{8}$, and $X_i = (x_{8(i-1)+1}, \dots, x_{8i})$ for every $i \in \{1, \dots, k\}$.
2. Use Algorithm 3 to construct an MQQ $(\mathbb{F}_2^8, q)$.
3. Compute the string $Y = Y_1Y_2 \dots Y_k$, where
$$Y_i = \begin{cases} X_1 & \text{if } i = 1, \\ q(X_{i-1}, X_i) & \text{if } i = 2, 4, \dots, k, \\ q(X_i, X_{i-1}) & \text{if } i = 3, 5, \dots, k-1. \end{cases} \tag{19}$$
4. Represent Y as a vector $\mathbf{y} = (y_1, \dots, y_n)$.

Output: The vector $\mathbf{y}$.

Algorithm 3 ConstructMQQ

Repeat

1. Construct $d \times d$ upper triangular Boolean matrices $\mathbf{U}_i$, $i \in \{1, \dots d-1\}$ that have all elements 0 except the elements in the rows from $\{1, \dots, i\}$ that are strictly above the main diagonal. Choose these elements uniformly at random from $\mathbb{F}_2$.
2. Choose randomly three nonsingular $d \times d$ matrices $\mathbf{A_1}$, $\mathbf{A_2}$ and $\mathbf{B}$ over $\mathbb{F}_2$ and a vector $\mathbf{c} \in \mathbb{F}_2^d$.
3. Form the $d \times d$ block matrix
$$\mathbf{U}(\mathbf{x}) = \mathbf{I}_d + \begin{bmatrix} \mathbf{0} & \mathbf{U}_1 \cdot \mathbf{A_1} \cdot \mathbf{x} & \mathbf{U}_2 \cdot \mathbf{A_1} \cdot \mathbf{x} & \dots & \mathbf{U}_{d-1} \cdot \mathbf{A_1} \cdot \mathbf{x} \end{bmatrix}.$$
4. Construct the mapping $q(\mathbf{x}, \mathbf{y}) = \mathbf{B} \cdot \mathbf{U}(\mathbf{x}) \cdot \mathbf{A_2} \cdot \mathbf{y} + \mathbf{B} \cdot \mathbf{A_1} \cdot \mathbf{x} + \mathbf{c}$, that defines an MQQ of order 2^d.

Until the following conditions about the matrices $\mathfrak{Q}^{(i)}$ of the coordinates q_i are satisfied:
$$\forall i \in \{1, \dots, d\}, \ \mathrm{Rank}(\mathfrak{Q}^{(i)}) \geqslant 2d - 4, \tag{20a}$$
$$\exists j \in \{1, \dots, d\}, \ \mathrm{Rank}(\mathfrak{Q}^{(j)}) = 2d - 2, \tag{20b}$$

Output: The MQQ $q(\mathbf{x}, \mathbf{y})$.

An important feature of the MQQs used, as it can be seen from Algorithm 3, is their bilinear nature, *i.e.*, the variables from the two operands are only mixed with each other quadratically, and there is no quadratic mixing of variables from one operand. This property makes the private key smaller, in particular, the bilinear MQQs only require 81 bytes of memory. Further, it also enables fast signing even in constrained environments by solving systems of linear equations.

The Algorithm used for computing the inverse of $\mathcal{F}$ is given in Algorithm 4.

```
**Algorithm 4** ComputeInverse$\mathcal{F}$
**Input:** A vector $\mathbf{y} = (y_1, \dots, y_n) \in \mathbb{F}_2^n$ , where $n$ is a multiple of 8.
  1. Represent the vector $\mathbf{y}$ as a string $Y = Y_1 Y_2 \dots Y_k$, where $k = \frac{n}{8}$, and $Y_i \in \mathbb{F}_2^8$
     for every $i \in \{1, \dots, k\}$.
  2. Compute the string $X = X_1 X_2 \dots X_k$, where
     $X_i = \begin{cases} Y_1, & \text{if } i = 1 \\ \text{the solution of } Y_i = q(X_{i-1}, X_i), & \text{if } i = 2, 4, \dots, k \\ \text{the solution of } Y_i = q(X_i, X_{i-1}), & \text{if } i = 3, 5, \dots, k-1 \end{cases}$
  3. Represent $X$ as a vector $\mathbf{x} \in \mathbb{F}_2^n$.
**Output:** The vector $\mathbf{x}$.
```

In [125], the authors propose a natural interpretation of the private key, in particular the secret quasigroup q. Instead of storing q, the holder of the private key can store the isotopic $q_0(\mathbf{x}, \mathbf{y}) = \mathbf{U}(\mathbf{A}_1^{-1} \cdot \mathbf{x}) \cdot \mathbf{y} + \mathbf{x} + \mathbf{c_0}$, and the invertible $\mathbf{A}_1, \mathbf{A}_2, \mathbf{B}$. In this case, the bilinear quasigroup can be stored in 50.5 bytes, rather than 81 bytes using the naive approach from the original paper.

In MQQ-SIG, as much as half of the public polynomials are removed in order to defend from Gröbner bases attacks. While this is not a problem for a signature scheme, an encryption scheme can not be build with such a heavy use of the minus modifier. Therefore, in the subsequent proposal MQQ-ENC for an encryption scheme [59], the authors propose to use left quasigroups instead.

Let (Q, q) be a left quasigroup of order p^{kd}. We say that (Q, q) is a Left Multivariate Quadratic Quasigroup (LMQQ) if q can be represented as a function $q = (q^{(1)}, q^{(2)}, \dots, q^{(d)}) : \mathbb{F}_{p^k}^{2d} \to \mathbb{F}_{p^k}^d$, where for every $s = 1, \dots, d$, $q^{(s)}$ is a quadratic polynomial over $\mathbb{F}_{p^k}$. For simplicity, we take that $Q = \mathbb{F}_{p^k}^d$.

The following theorem provides sufficient conditions for a multivariate mapping to define a quasigroup.

Theorem 10. *The function* $q_0 = (q^{(1)}, q^{(2)}, \dots, q^{(d)}) : \mathbb{F}_{p^k}^{2d} \to \mathbb{F}_{p^k}^{d}$ *such that for every* $s = 1, \dots, d$, *the component* $q_0^{(s)}$ *is of the form*

$$\begin{aligned} q_0^{(s)}(x_1, \dots, x_d, y_1, \dots, y_d) &= p^{(s)}(y_s) + \sum_{1 \leqslant i,j \leqslant d} \alpha_{i,j}^{(s)} x_i x_j + \sum_{s<i,j \leqslant d} \beta_{i,j}^{(s)} y_i y_j + \\ + \sum_{1 \leqslant i \leqslant d, s<j \leqslant d} \gamma_{i,j}^{(s)} x_i y_j &+ \sum_{1 \leqslant i \leqslant d} \delta_i^{(s)} x_i + \sum_{s<i \leqslant d} \epsilon_i^{(s)} y_i + \eta^{(s)}, \end{aligned} \tag{21}$$

where $p^{(s)}(x) = ax$, $a \neq 0$, *or* $p^{(s)}(x) = ax^2$, $a \neq 0$, $p = 2$, *defines an LMQQ* $(\mathbb{F}_{p^k}^{d}, q_0)$ *of order* p^{kd}.

For the purpose of MQQ-ENC, and using the form from Theorem 10 the LMQQs can be constructed using Algorithm 5.

```
Algorithm 5 CreateLMQQ($(d, p, k)$)
Input $d, p, k \in \mathbb{N}$, where $p$ is prime.
  1. For all $s \in \{1, \dots, d\}$ generate at random from $\mathbb{F}_{p^k}$ the coefficients:
       • $\alpha_{i,j}^{(s)}$, $\delta_i^{(s)}$, for all $i, j$, $1 \leqslant i, j \leqslant d$, and $\beta_{i,j}^{(s)}$, $\epsilon_i^{(s)}$, for all $i, j$, $s < i, j \leqslant d$,
       • $\gamma_{i,j}^{(s)}$, for all $i, j$, $1 \leqslant i \leqslant d, s < j \leqslant d$, and the constant term $\eta^{(s)}$.
  2. For all $s \in \{1, \dots, d\}$
       • If $p = 2$ generate at random a bit $r \in \mathbb{F}_2$, otherwise set $r = 0$.
       • Choose at random $a^{(s)} \in \mathbb{F}_{p^k} \setminus \{0\}$. If $r = 0$ set $p^{(s)} = a^{(s)} x_s$, otherwise
         set $p^{(s)} = a^{(s)} x_s^2$.
  3. For all $s \in \{1, \dots, d\}$ construct $q_0^{(s)}(\mathbf{x}, \mathbf{y})$ given by (21), and the LMQQ
     $q_0 = (q_0^{(1)}, q_0^{(2)}, \dots, q_0^{(d)})$.
  4. Generate at random over $\mathbb{F}_{p^k}$, $d \times d$ nonsingular matrices $\mathbf{D}, \mathbf{D_y}$, and vectors
     $\mathbf{c}, \mathbf{c_y}$ of dimension $d$.
Output the quintet $(q_0, \mathbf{D}^{-1}, \mathbf{D_y}^{-1}, \mathbf{c}, \mathbf{c_y})$ and the LMQQ of order $p^{kd}$:
     $q(\mathbf{x}, \mathbf{y}) = \mathbf{D} \cdot q_0(\mathbf{x}, \mathbf{D_y} \cdot \mathbf{y} + \mathbf{c_y}) + \mathbf{c}$.
```

As in MQQ-SIG, an efficient algorithm for inverting the central mapping is based on efficiently computing the parastrophe $q_\backslash$ of q at a given point. In other words, the problem is reduced to to solving the system of d quadratic equations in d variables $y_1, y_2, \dots, y_d$ over $\mathbb{F}_{p^k}$

$$q(\mathbf{u}, \mathbf{y}) = \mathbf{v} \tag{22}$$

Even though this is a non trivial problem in general, the specific structure of the LMQQs in use, allows this system to be solved in polynomial time, very efficiently and fast.

The MQQ-ENC cryptosystem is defined as a triplet of probabilistic algorithms MQQ-ENC$= (\mathcal{G}^{MQQ}, \mathcal{E}^{MQQ}, \mathcal{D}^{MQQ})$, associated to a message space $Mspace(nk) = \{0,1\}^{nk/2}$, and random coins $Coins(nk) = \{0,1\}^{nk/4}$, given by Algorithms 6, 7 and 8 as follows.

Algorithm 6 Key-Generation algorithm $\mathcal{G}^{MQQ}$

```
Input: $1^{nk}$,
  1. Run CreateST$(n, 2, k, r_1, r_2, rem)$ to obtain
     $(\sigma_1, \sigma_2, \mathbf{M}_0, (a_i^{(1)})_{r_1+1}, (a_i^{(2)})_{r_2+1})$ and the affine mappings $\mathcal{S}$ and $\mathcal{T}$.
  2. Run CreateLMQQ$(8, 2, k)$ to obtain $(q_0, \mathbf{D}, \mathbf{D}_\mathbf{y}, \mathbf{c}, \mathbf{c}_\mathbf{y})$ and $q$.
  3. Represent the vector $(x_1, x_2, \dots, x_n)$ of variables over $\mathbb{F}_{2^k}$ as a vector
     $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{n/8})$ of variables over $\mathbb{F}_{2^k}^8$, where $\mathbf{x}_i = (x_{8i-7}, x_{8i-6}, \dots, x_{8i})$.
  4. Define a mapping $\mathcal{F} : \mathbb{F}_{2^k}^n \to \mathbb{F}_{2^k}^n$ (a quasigroup string transformation) by:
     $$\begin{aligned}(y_1, \dots, y_n) &= \mathcal{F}(x_1, \dots, x_n) \Leftrightarrow \\ (\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_{n/8}) &= (q(11\dots1, \mathbf{x}_1), q(\mathbf{x}_1, \mathbf{x}_2), \dots, q(\mathbf{x}_{n/8-1}, \mathbf{x}_{n/8}))\end{aligned} \tag{23}$$
  5. Construct the mapping $P_{full} : \mathbb{F}_{2^k}^n \to \mathbb{F}_{2^k}^n$ as $P_{full} = \mathcal{T} \circ \mathcal{F} \circ \mathcal{S}$. We use the
     notation $\mathcal{P}_{full} = (p_1, p_2, \dots, p_n)$, where $p_i(x_1, \dots, x_n)$, $1 \leqslant i \leqslant n$.
  6. The vector of polynomials $\mathcal{P} : \mathbb{F}_{2^k}^n \to \mathbb{F}_{2^k}^{n-rem}$ is obtained by removing the
     last $rem$ coordinates from $\mathcal{P}_{full}$, i.e., $\mathcal{P} = (p_1, p_2, \dots, p_{n-rem})$.
  7. Choose a universal hash function $H : \{0,1\}^{3nk/4} \to \{0,1\}^{nk/4}$.
  8. Set $\mathsf{pk} = (\mathcal{P}, H)$,
     and $\mathsf{sk} = (\sigma_1, \sigma_2, \mathbf{M}_0, (a_i^{(1)})_{r_1+1}, (a_i^{(2)})_{r_2+1}, q_0, \mathbf{D}^{-1}, \mathbf{D}_\mathbf{y}^{-1}, \mathbf{c}, \mathbf{c}_\mathbf{y})$.
Output: Public private key pair $(\mathsf{pk}, \mathsf{sk})$.
```

Algorithm 7 Encryption algorithm $\mathcal{E}^{MQQ}$

```
Input: Public key $\mathsf{pk} = (\mathcal{P}, H)$ and plaintext message $m = \{m_1, m_2, \ldots, m_{n/2}\} \in Mspace(nk)$,
    1. Generate a random string $r = \{r_1, r_2, \ldots, r_{n/4}\} \in Coins(nk)$.
    2. Evaluate $(h_1, \ldots, h_{n/4}) = H(m, r) = H(m_1, \ldots, m_{n/2}, r_1, \ldots, r_{n/4})$.
    3. Evaluate $\mathcal{P}(m, r, H(m, r)) = \mathcal{P}(m_1, \ldots, m_{n/2}, r_1, \ldots, r_{n/4}, h_1, \ldots, h_{n/4})$.
Output: Ciphertext $c = \mathcal{P}(m, r, H(m, r))$.
```

Algorithm 8 Decryption algorithm $\mathcal{D}^{MQQ}$

```
Input: Private key $\mathsf{sk} = (\sigma_1, \sigma_2, \mathbf{M}_0, (a_i^{(1)})_{r_1+1}, (a_i^{(2)})_{r_2+1}, q_0, \mathbf{D}^{-1}, \mathbf{D}_{\mathbf{y}}^{-1}, \mathbf{c}, \mathbf{c}_{\mathbf{y}})$
and cipher $c = (c_1, \ldots, c_{n-rem}) \in \mathbb{F}_{2^k}^{n-rem}$,
For all $(c_{n-rem+1}, c_{n-rem+2}, \ldots, c_n) \in \mathbb{F}_{2^k}^{rem}$ do
    1. Evaluate $(m'_1, m'_2, \ldots, m'_n) = \mathcal{S}^{-1} \circ \mathcal{F}^{-1} \circ \mathcal{T}^{-1}(c_1, c_2, \ldots, c_n)$, where $\mathcal{F}^{-1}$ is
       evaluated by:
           $(u_1, u_2, \ldots, u_n) = \mathcal{F}^{-1}(v_1, v_2, \ldots, v_n) \Leftrightarrow$
           $(\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{n/8}) = (q_\backslash(\mathbf{u}_0, \mathbf{v}_1), q_\backslash(\mathbf{u}_1, \mathbf{v}_2), q_\backslash(\mathbf{u}_2, \mathbf{v}_3), \ldots, q_\backslash(\mathbf{u}_{n/8-1}, \mathbf{v}_{n/8}))$ (24)
       Here, $\mathbf{u}_0 = (11\ldots1)$, and for every $i \in \{0, \ldots, n/8-1\}$, $\mathbf{u}_{i+1} = q_\backslash(\mathbf{u}_i, \mathbf{v}_{i+1})$
       is evaluated by running Algorithm $\mathbf{Q}_\backslash(\mathbf{u}_i, \mathbf{v}_{i+1}, 8, 2, k, q_0, \mathbf{D}^{-1}, \mathbf{D}_{\mathbf{y}}^{-1}, \mathbf{c}, \mathbf{c}_{\mathbf{y}})$.
       The vector $(u_1, \ldots, u_n)$ over $\mathbb{F}_{2^k}$ is represented as a vector $(\mathbf{u}_1, \ldots, \mathbf{u}_{n/8})$
       over $\mathbb{F}_{2^k}^8$, where $\mathbf{u}_i = (u_{8i-7}, u_{8i-6}, \ldots, u_{8i})$. Analogously, the same is done
       for the vector $(v_1, v_2, \ldots, v_n)$.
    2. If $H(m'_1, m'_2, \ldots, m'_{3n/4}) = (m'_{3n/4+1}, m'_{3n/4+2}, \ldots, m'_n)$ then break;
End for;
Output: Plaintext $m'$ or $\perp$ if the above test failed for all $(c_{n-rem+1}, \ldots, c_n) \in \mathbb{F}_{2^k}^{rem}$.
```

5. Conclusion

The aim of this article was to present how quasigroups can be exploit for building suitable cryptographic primitives. There were presented constructions of several types of quasigroups and several types of quasigroups string transformations. Designs with these types of quasigroups and transformations were illustrated in constructions of S-boxes, block cipher, stream ciphers, pseudo random number generators, hash functions and public key security and signatures. There are also other applications of quasigroups in cryptography (MAC, Identity encryption schemes, Authenticated encryption, ...) but we found that what we had presented is quite enough to conclude that, slowly but surely, quasigroups are taking there role in cryptography. We emphasize that there are several other survey papers where different applications of quasigroups in cryptography are discussed as well: [132], [60], [133], [134], [135], [108], etc.

We have to notice that cryptographic properties are not discussed in this paper. The efficiency and security of the crypto products based on quasigroup is an open research problem for cryptographers and cryptanalysts. There are many broken designs based on quasigroups, but also there are some with perfect crypto properties.

At the end, one can notice that the presented results were mostly from Macedonian quasigroupists and cryptographers; that was done intentionally.

Acknowledgment. The author is grateful to Aleksandra Mileva, Vesna Dimitrova, Simona Samardjiska, Aleksandra Popovska-Mitrovikj and Hristina Mihajloska for their support and help in writing of this article.

References

[1] **R. Ahlavat, K. Gupta and S.K. Pal**, *Fast generation of multivariate quadratic quasigroups for cryptographic applications*, IMA Conference on Mathematics in Defence, Farnborough, UK, 2009, http://www.ima.org.uk/_db_documents/defence09_ahlawat_v2.pdf.

[2] **V. Bakeva and V. Dimitrova**, *Some probabilistic properties of quasigroup processed strings useful for cryptanalysis*, Proc. of ICT Innovation 2010, Springer Berlin Heidelberg, 2010, pp. 61–70.

[3] **V. Bakeva, V. Dimitrova and A. Popovska-Mitrovikj**, *Parastrophic quasigrouop string processing*, Proc. of the 8^{th} Conf. Informatics and Information Technology, Bitola, Macedonia, 2011, 19–21.

[4] **V. Bakeva and N. Ilievska**, *A probabilistic model of error-detecting codes based on quasigroups*, Quasigroups and Related Systems **17** (2009), 151–164.

[5] **V. Bakeva, A. Popovska-Mitrovikj and V. Dimitrova**, *Resistance of statistical attacks of parastrophic quasigroup transformation*, arXiv: 1404.0781v1.

[6] **S. Bakhtiari, R. Safavi-Naini and J. Pieprzyk**, *A message authentication code based on Latin square*, Proc. of ACISP 97, LNCS **1270** (1997), 194–203.

[7] **M. Battey and A. Parakh**, *A quasigroup based random number generator for resource constrained environments*, IACR Cryptology ePrint Archive, Report 2012: 471.

[8] **V.D. Belousov**, *Foundations of the theory of quasigroups and loops*, (Russian), Nauka, Moscow, 1967.

[9] **V.D. Belousov**, *n-ary quasigroups*, (Russian), Ştiinţa, Kishinev, 1972.

[10] **G.B. Belyavskaya**, *Recursively r-differentiable quasigroups within S-systems and MDS-codes*, Quasigroups and Related Systems **20** (2012), 157–168.

[11] **G.B. Belyavskaya, V.I. Izbash and G.L. Mullen**, *Check character systems using quasigroups: II*, Designs, Codes and Cryptography **37** (2005), 405–419.

[12] **G.B. Belyavskaya, V.I. Izbash and V.A. Shcherbacov**, *Check character systems over quasigroups and loops*, Quasigroups and Related Systems **10** (2003), 1–28.

[13] **D.J. Bernstein and T. Lange (eds.)**, eBACS: ECRYPT Benchmarking of Cryptographic Systems, 2014.

[14] **A. Bogdanov, L.R. Knudsen, G. Le, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin and C. Vikkelsoe**, *PRESENT: An ultra-lightweight block cipher*, Proc. of CHES 2007, Springer-Verlag, pp. 450–466.

[15] **G. Carter, E. Dawson and L. Nielsen**, *DESV: A Latin square variation of DES*, Proc. of the Workshop on Selected Areas in Cryptography, Ottawa, Canada, 1995, pp. 144–158.

[16] **S. Chakrabarti, S.K. Pall and G. Gangopadhyay**, *An improved 3-quasigroup based encrytion scheme*, Proc. of ICT Innovations 2012, http://ictinnovations.org/2012/htmls/papers/WebProceedings2012.pdf, 2012, pp. 173–184.

[17] **Y. Chen, S.J. Knapskog and D. Gligoroski**, *Multivariate quadratic quasigroups (MQQs): Construction, bounds and complexity*, Inscrypt, 6th International Conference on Information Security and Cryptology. Science Press of China, 2010.

[18] **N. Courtois, A. Klimov, J. Patarin and A. Shamir**, *Efficient algorithms for solving overdefined systems of multivariate polynomial equations*, LNCS **1807** (2000), 392–407.

[19] **P. Csőrgő and V. Shcherbacov**, *On some quasigroup cryptographical primitives*, arXiv: 1110.6591v1.

[20] **J. Daemen**, *Cipher and hash function design. Strategies based on linear and differential cryptanalysis*, Doctoral dissertation. Katholieke Universiteit Leuven, 1995.

[21] **H.M. Damm**, *Totally anti-symmetric quasigroups for all orders $n \neq 2, 6$*, Discrete Math. **307** (2007), 715–729.

[22] **H.M. Damm**, *Half quasigroups and generalized quasigroup orthogonality*, Discrete Math. **311** (2011), 145–153.

[23] **F. Dawson, D. Donowan and A. Offer**, *Quasigroups, isotopisms and authentication schemes*, Australasian J. Combin. **13** (1996), 75–88.

[24] **J. Dénes and A.D. Keedwell**, *Latin squares and their applications*, Akadémiai Kiado, Budapest, 1974.

[25] **J. Dénes and A.D. Keedwell (Eds.)**, *Latin squares: New developments in the theory and applications*, Annals Discr. Math. **46**, Elsevier, 1991.

[26] **J. Dénes and A.D. Keedwell**, *A new authentication scheme based on Latin squares*, Discrete Math. **106/107** (1992), 157–161.

[27] **J. Dénes and A.D. Keedwell**, *Some applications of non-associative algebraic systems in cryptology*, Pure Math. Appl. **12** (2001), 147–195.

[28] **M. Dichtl**, *Bad and good ways of post-processing biased physical random numbers*, LNCS **4593** (2007), 137–152.

[29] **M. Dichtl and P. Bőffgen**, *Breaking another quasigroup-based cryptographic scheme*, Cryptology ePrint Archive, Report 2012: 661.

[30] http://stat.fsu.edu/pub/diehard/, http://en.wikipedia.org/wiki/Diehard_tests

[31] **V. Dimitrova**, *Quasigroup processed strings, their Boolean presentation and application in cryptography and coding theory*, Doctoral dissertation. University Sts. Cyril and Methodius, Skopje, 2010.

[32] **V. Dimitrova, V. Bakeva, A. Popovska-Mitrovikj and A. Krapež**, *Cryptographic properties of parastrophic quasigroup transformation*, Advances in Intelligent Systems and Computing - ICT Innovations 2012, Springer, 2013, pp. 235–243.

[33] **V. Dimitrova and S. Markovski**, *On Quasigroup pseudo random sequence generators*, Proc. of the 1st Balkan Confer. in Informatics, Thessaloniki, Greece, 2004, pp. 235–243.

[34] **V. Dimitrova and S. Markovski**, *Classification of quasigroups by image patterns*, Proc. of the Fifth International Confer. for Informatics and Information Technology, Bitola, Macedonia, 2007, pp. 152–160.

[35] **H. Dobbertin**, *One-to-one highly nonlinear power functions on $GF(2n)$*, Applicable Algebra in Engineering, Communication and Computing **9** (1998), 139–152.

[36] **A. Drapal**, *Hamming distances of groups and quasigroups*, Discrete Math. **235** (2001), 189–197.

[37] **J. Dvorsky, E. Ochodkova and V. Snašel**, *Hash function based on quasigroups*, (Czech), Proc. of Mikulášska kryptobesídká, Praha, 2001, pp. 27–36.

[38] **J. Dvorsky, E. Ochodkova and V. Snašel**, *Hash function based on large quasigroups*, (Czech), Proc. of Velikonocní i kryptologie, Brno, 2002, pp. 1–8.

[39] **J. Dvorsky, E. Ochodkova and V. Snašel**, *Generation of large quasigroups: an application in cryptography*, Proc. of AAA (Arbeitstagung Allgemeine), 2002.

[40] **J.C. Faugère, R.S. Ødegård, L. Perret and D. Gligoroski**, *Analysis of the MQQ Public Key Cryptosystem*, LNCS **6467**, 169–183.

[41] **D. Gligoroski**, *Stream cipher based on quasigroup string transformations in Z_p^**, Contributions, Sec. Math. Tech. Sci., MANU, 2004.

[42] **D. Gligoroski**, *Candidate one-way functions and one-way permutations based on quasigroup string transformations*, IACR Cryptology ePrint Archive, Report 2005: 352.

[43] **D. Gligoroski, S. Andova and S.J. Knapskog**, *On the importance of the key separation principle for different modes of operation*, LNCS **4991** (2008), 404–418.

[44] **D. Gligoroski, V. Dimitrova and S. Markovski**, *Quasigroups as Boolean functions, their equation systems and Groebner bases*, In M. Sala, T. Mora, L. Perret, S. Sakata, C. Traverso (Eds.) Groebner Bases, Coding, and Cryptography, Springer Berlin, 2009.

[45] **D. Gligoroski, S.J. Knapskog**, *Adding MAC functionality to Edon80*, International J. Computer Science and Network Security **7** (2007), 194–204.

[46] **D. Gligoroski, S.J. Knapskog**, *Edon-$R(256, 384, 512)$-an efficient implementation of Edon-R family of cryptographic hash functions*, Comment. Math. Univ. Carolin. **49** (2008), 219–239.

[47] **D. Gligoroski, S. Markovski and V. Bakeva**, *On infinite class of strongly collision resistant hash functions "Edon-F" with variable length of output*, Proc. of 1st International Confer. on Mathematics and Informatics for Industry, Thessaloniki, 2003, pp. 302–308.

[48] **D. Gligoroski, S. Markovski and S.J. Knapskog**, *A fix of the $MD4$ family of hash functions – quasigroup fold*, NIST Cryptographic Hash Workshop, Gaithersburg, Maryland, USA,
http://csrc.nist.gov/groups/ST/hash/documents/Gligoroski_MD4Fix.pdf, 2005.

[49] **D. Gligoroski, S. Markovski and S.J. Knapskog**, *A secure hash algorithm with only 8 folded $SHA-1$ steps*, Intern. J. Computer Science and Network **6(10)** (2006), 194–205.

[50] **D. Gligoroski, S. Markovski and S.J. Knapskog**, *On periods of Edon-$(2m, 2k)$ family of stream ciphers*, SASC 2006 Conference.

[51] **D. Gligoroski, S. Markovski and S.J. Knapskog**, *Multivariate quadratic trapdoor functions based on multivariate quadratic quasigroups*, Proc. Amer. Confer. Appl. Math., Harvard, USA, 2008, pp. 44–49.

[52] **D. Gligoroski, S. Markovski and S.J. Knapskog**, *A pblic key block cipher based on multivariate quadratic quasigrops*, IACR Cryptology ePrint Archive, Report 2008: 320.

[53] **D. Gligoroski, S. Markovski and S.J. Knapskog**, *The stream cipher Edon80*, LNCS **4986** (2008), 152–169.

[54] **D. Gligoroski, S. Markovski and Lj. Kocarev**, *Edon-R, an infinite family of cryptographic hash functions*, International J. Network Security **8** (2006), 293–300.

[55] **D. Gligoroski, S. Markovski and Lj. Kocarev**, *Error-correcting codes based on quasigroups*, Proc. of 16th International Confer. on Computer Communications and Networks - ICCCN 2007, Honolulu, 2007, pp. 165–172.

[56] **D. Gligorovski and R.S. Ødegård**, *On the complexity of Khovratovich et. al. preimage attack on EDON-R*, IACR Cryptology ePrint Archive, Report 2009: 120.

[57] **D. Gligorovski, R.S. Ødegård, R.E. Jensen, L. Perret, J.C. Faugère, S.J. Knapskog and S. Markovski**, *MQQ-SIG: An ultra-fast and provably CMA resistant digital signature scheme*, LNCS **7222** (2011), 184–203.

[58] **D. Gligorovski, R.S. Ødegård, M. Mihova, S.J. Knapskog, Lj. Kocarev, A. Drapal and V. Klima**, *Cryptographic hash function EDON-R*, from http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/documents/Edon-RUpdate.zip, 2008.

[59] **D. Gligoroski and S. Samardjiska**, *The multivariate probabilistic encryption scheme MQQ-ENC.* IACR Cryptology ePrint Archive, Report 2012: 328.

[60] **M.M. Glukhov**, *On application of quasigroups in cryptology*, (Russian). Appl. Disc. Math. **2** (2008), 28–32.

[61] **S. Golomb, L. Welch and J. Dénes**, *Encryption system based on crossed inverse quasigroups*, US patent, WO0191368, 2001.

[62] **M. Hell and T. Johansson**, *A key recovery attack on Edon80*, LNCS **4833** (2007), 568–581.

[63] **Y. Hu**, *Security analysis of cryptosystem based on quasigroups*, Proc. of IEEE Intern. Conf. on Progress in Informatics and Computing, 2010, pp. 431–435.

[64] **L. Ji, X. Liangyu and G. Xu**, *Collision attack on NaSHA*-512, IACR Cryptology ePrint Archive, Report 2008: 519, 2008.

[65] **D.M. Johnson, A.L. Dulmage and N.S. Mendelsohn**, *Orthomorphisms of groups and orthogonal latin squares I*, Canadian J. Math. **13** (1961), 356–372.

[66] **A.D. Keedwell**, *Crossed inverse quasigroups with long inverse cycles and applications to cryptography*, Australasian J. Combin. **20** (1999), 241–250.

[67] **A.D. Keedwell and V.A. Shcherbacov**, *Construction and properties of* (r,s,t)-*inverse quasigroups. I*, Discrete Math. **266** (2003), 275–291.

[68] **D. Khovratovich, I. Nikolic and R.P. Weinmann**, *Cryptanalysis of Edon-R*, http://ehash.iaik.tugraz.at/uploads/7/74/Edon.pdf, 2008.

[69] **V. Klima**, *Multicollisions of EDON-R hash function and other observations*, http://cryptography.hyperlink.cz/BMW/EDONR_analysis_vk.pdf, 2008.

[70] **C. Kościelny**, *A method of constructing quasigroup-based stream-ciphers*, Appl. Math. Computer Sci. **6** (1996), 109–121.

[71] **C. Kościelny**, *Generating quasigroups for cryptographic applications*, Intern. J. Appl. Math. Computer Sci. **12** (2002), 559–569.

[72] **C. Kościelny and G.L. Mullen**, *A quasigroup-based public-key cryptosystem*, Intern. J. Appl. Math. Computer Sci. **9** (1999), 955–963.

[73] **A. Krapež**, *An application of quasigroup in cryptology*, Math. Macedonica **8** (2010), 47–52.

[74] **A. Krapež**, *Cryptographically suitable quasigroups via functional equations*, Advances in Intelligent Systems and Computing - ICT Innovations 2012, Springer Berlin Heidelberg, 2013, pp. 265–273.

[75] **C.F. Laywine, and G.L. Mullen**, *Discrete mathematics using Latin squares*, New York: John Wiley Sons, Inc., 1998.

[76] **G. Leander and V. Poschmann**, *On the classification of* 4 *bit S-boxes*, LNCS **4547** (2007), 159–176.

[77] **G. Leurent**, *Key recovery attack against secret-prefix Edon-R*, IACR Cryptology ePrint Archive, Report 2009: 135.

[78] **Z. Li and D. Li**, *Collision attack on NaSHA*-384/512, IACR Cryptology ePrint Archive, Report 2009: 026.

[79] **R.J.M. Maia, P.S.L. M. Barreto and B.T. de Oliveira**, *Implementation of multivariate quadratic quasigroup for Wireless Sensor Network*, LNCS **6480** (2010), 64–78.

[80] **S. Markovski**, *Quasigroup string processing and applications in cryptography*, Proc. 1-st Inter. Conf. Mathematics and Informatics for industry MII 2003, 14-16 April, Thessaloniki, 2003, pp. 278–290.

[81] **S. Markovski and V. Bakeva**, *Quasigroup string processing: Part* 4, Contributions, Sec. math. Tech.Sci., MANU, **22** (2001).

[82] **S. Markovski, V. Dimitrova and S. Samardziska**, *Identities sieves for quasigroups*, Quasigroups and Related Systems **18** (2010), 149–164.

[83] **S. Markovski, V. Dimitrova, Z. Trajcheska, M. Petkovska, M. Kostadinoski and D. Buhov**, *Block cipher defined by matrix presentation of quasigroups*, Proc. of the 11th Confer. on Informatics and Information Technology, CIIT 2014, Bitola (to appear).

[84] **S. Markovski, D. Gligoroski and S. Andova**, *Using quasigroups for one-one secure encoding*, Proc. of VIII Conf. Logic and Computer Science LIRA97, Novi Sad, Serbia, 1997, pp. 157–162.

[85] **S. Markovski, D. Gligoroski and V. Bakeva**, *Quasigroup string processing - Part* 1, Contributions, Sec. Math. Tech. Sci., MANU **20** (1999), 13–28.

[86] **S. Markovski, D. Gligoroski and V. Bakeva**, *Quasigroup and hash functions*, Proc. of the 6th ICDMA, Bansko, 2001, pp. 43–50.

[87] **S. Markovski, D. Gligoroski and Lj. Kocarev**, *Unbiased random sequences from quasigroup string transformations*, LNCS **3557** (2005), 163–180.

[88] **S. Markovski, D. Gligoroski and B. Stojčevska**, *Secure two-way on-line communication by using quasigroup enciphering with almost public key*, Novi Sad J. Math. **30(2)** (2000), 43–49.

[89] **S. Markovski and V. Kusakatov**, *Quasigroup string processing - Part* 2, Contributions, Sec. Math. Tech. Sci., MANU **21** (2000), 15–32.

[90] **S. Markovski and V. Kusakatov**, *Quasigroup string processing - Part* 3, Contributions, Sec. Math. Tech. Sci., MANU **23-24** (2002-2003), 7–27.

[91] **S. Markovski and A. Mileva**, *Generating huge quasigroups from small nonlinear bijections via extended Feistel function*, Quasigroups and Related Systems **17** (2009), 91–106.

[92] **S. Markovski and A. Mileva**, *NaSHA*, Submission to NIST, First Round SHA-3 Candidate, http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/documents/NaSHAUpdate.zip, 2008.

[93] **S. Markovski, A. Mileva, V. Dimitrova and D. Gligoroski**, *On a conditional collision attack on NaSHA*-512, IACR Cryptology ePrint Archive, Report 2009: 034.

[94] **S. Markovski, S. Samardziska, D. Gligoroski and S.J. Knapskog**, *Multivariate Trapdoor functions based on multivariate left quasigroups and left polynomial quasigroups*, Proc. of the Second Intern. Confer. on Symbolic Computation and Cryptography, Royal Holloway, University of London, Egham, UK, 2010, pp. 237–251.

[95] **S. Markovski, Z. Šunić and D. Gligoroski**, *Polynomial functions on the units of* $\mathbb{Z}_{2^n}$, Quasigroups and Related System **18** (2010), 11–34.

[96] **S.I. Marnas, L. Angelis and G.L. Bleris**, *All-Or-Nothing Transform using quasigroups*, Proc. 1st Balkan Conference in Informatics, Thessaloniki, 2004, pp. 183–191.

[97] **M. Matsumoto, M. Saito, T. Nishimura and M. Hagita**, *A fast stream cipher with huge state space and quasigroup filter for software*, LNCS **4876** (2007), 246–263.

[98] **M. Matsumoto, M. Saito, T. Nishimura and M. Hagita**, *CryptMT*3 *stream cipher*, LNCS **4986** (2007), 7–19.

[99] **T. Matsumoto and H. Imai**, *Public quadratic polynomial-tuples for efficient signature-verification and message-encryption*, LNCS **330** (1988), 419–453.

[100] **B.D. McKay, A. Meynert and W. Myrvold**, *Small Latin squares, quasigroups and loops*, J. Combinatorial Designs **15** (2007), 98–119.

[101] **A.J. Menezes, P.C. van Oorschot and S.A. Vanstone**, *Handbook of Applied Cryptography*, CRC Press, 2001.

[102] **K.A. Meyer**, *A new message authentication code based on the non-associativity of quasigroups*, Doctoral dissertation, Iowa State University, 2006.

[103] **H. Mihajloska and D. Gligoroski**, *Construction of optimal* 4*-bit S-boxes by quasigroups of order* 4, Proc. of SECURWARE 2012, Rome, Italy, 2012, pp. 163–168.

[104] **H. Mihajloska, T. Yalcin and D. Gligoroski**, *How lightweight is the hardware implementation of quasigroup S-boxes*, Advances in Intelligent Systems and Computing - ICT Innovations 2012 207 , Springer Berlin Heidelberg, 2013, pp. 121–127.

[105] **M. Mihova, M. Siljanoska and S. Markovski**, *Tracing bit differences in strings transformed by linear quasigroups of order* 4, Proc. of the 9th Confer. for Informatics and Information Technology, Bitola, Macedonia, 2012, pp. 229–233.

[106] **A. Mileva**, *Cryptographic primitives with quasigroup transformations*, Doctoral dissertation, University Sts. Cyril and Methodius, Skopje, 2010.

[107] **A. Mileva**, *Analysis of some quasigroup transformations as Boolean functions*, Mathematica Balkanica **3-4** (2012).

[108] **A. Mileva**, *New developments in quasigroup-based cryptography*, Multidisciplinary Perspectives in Cryptology and Information Security. IGI-Global, 2014, pp. 286–317.

[109] **A. Mileva and S. Markovski**, *Correlation matrices and prop ratio tables for quasigroups of order* 4, Proc. of the 6th Intern. Confer. for Informatics and Information Technology, Ohrid, Macedonia, 2008, pp. 17–22.

[110] **A. Mileva and S. Markovski**, *Quasigroups string transformations and hash function design*, ICT Innovations 2009, Springer Berlin Heidelberg, 2010, pp. 367–376.

[111] **A. Mileva and S. Markovski**, *Shapeless quasigroups derived by Feistel orthomorphisms*, Glasnik Matematicki **47** (2012), 333–349.

[112] **A. Mileva and S. Markovski**, *Quasigroup representation of some Feistel and Generalized Feistel Ciphers*, Advances in Intelligent Systems and Computing - ICT Innovations 2012, 207, Springer, 2013, pp. 161–171.

[113] **M.S. Mohamed, J. Ding, J. Buchmann and F. Werner**, *Algebraic attack on the MQQ public key cryptosystem*, Proc. of 8th Intern. Confer. on Cryptology and Network Security, Springer Berlin Heidelberg, 2009, pp. 391–401.

[114] **I. Nikolić and D. Khovratovich**, *Free-start attacks on NaSHA*, http://ehash.iaik.tugraz.at/uploads/3/33/Free-start_attacks_on_Nasha.pdf, 2008.

[115] **P. Novotney and N. Ferguson**, *Detectable correlation in Edon-R.* IACR Cryptology ePrint Archive 2009: 378.

[116] **J. Patarin**, *Hidden fields equations* (HFE) *and isomorphisms of polynomials* (IP)*: two new families of asymmetric algorithms*, LNCS **1440** (1996), 33–48.

[117] **A. Petrescu**, *n-quasigroup cryptographic primitives: stream ciphers*, Studia Univ. Babes Bolyai, Informatica **55(2)** (2010), 27–34.

[118] **H.O. Pflugfelder**, *Quasigroups and Loops: Introduction.* Heldermann Verlag, Berlin, 1991.

[119] **A. Popovska-Mitrovikj, V. Bakeva and S. Markovski**, *On random error correcting codes based on quasigroups*, Quasigroups and Related Systems **19** (2011), 301–316.

[120] **R.L. Rivest**, *All-or-nothing encryption and the package transform*, LNCS **1267** (1997), 210–218.

[121] **R.L. Rivest**, *Permutation polynomials modulo* $2w$, Finite Fields and Their Appl. **7** (2001), 287–292

[122] **A. Sade**, *Quasigroups automorphes par le groupe cyclique*, Canadian J. Math. **9** (1957), 321–335.

[123] **S. Samardziska**, *Polynomial n-ary quasigroups of order w*, Master thesis, University Ss. Cyril and Methodius, Skopje, 2009.

[124] **S. Samardziska**, *ID based identification schemes using multivariate left quasigroups*, (submitted).

[125] **S. Samardziska, Y. Chen and D. Gligoroski**, *Algorithms for construction of multivariate quadratic quasigroups* ($MQQs$) *and their parastrophe operations in arbitrary Galois Fields*, J. Inform. Assurance and Security **7(3)** (2012), 164–172.

[126] **S. Samardziska and D. Gligoroski**, *Identity-based identification schemes using left multivariate quasigroups*, NIK-2011, Tapir Akad, Forlag, 2011, pp. 19–30.

[127] **S. Samardziska, S. Markovski and D. Gligoroski**, *Multivariate quasigroups defined by T-functions*, Proc. of the Second Intern. Confer. on Symbolic Computation and Cryptography, University of London, 2010, pp. 117–127.

[128] **D.G. Sarvate and J. Seberry**, *Encryption methods based on combinatorial designs*, Ars Combinatoria **21A** (1986), 237–246.

[129] **M.V.K. Satti**, *Quasi-group based crypto-system*, Master thesis, Louisiana State University, 2007.

[130] **M. Satti and S. Kak**, *Multilevel indexed quasigroup encryption for data and speech*, IEEE Transactions on Broadcasting **55(2)** (2009), 270–281.

[131] **R.H. Schulz**, *A note on check character systems using latin squares*, Discrete Math. **97** (1991), 371–375.

[132] **V.A. Shcherbacov**, *On some known possible applications of quasigroups in cryptology*, http://www.karlin.mff.cuni.cz/ drapal/krypto.pdf, 2003.

[133] **V.A. Shcherbacov**, *Quasigroups in cryptology*, Computer Sci. J. Moldova **17(2)** (2009), 193–228.

[134] **V.A. Shcherbacov**, *Quasigroups in cryptology*, arXiv:1007.3572.

[135] **V.A. Shcherbacov**, *Quasigroup based crypto-algorithms*, arXiv:1201.3016v1.

[136] **V.A. Shcherbacov**, *Quasigroup based hybrid of a code and a cipher*, http://ictinnovations.org/2012/htmls/papers/WebProceedings2012.pdf, p.411–417.

[137] **M. Simjanovska, M. Mihova and S. Markovski**, *Matrix presentation of quasigroups of order* 4, Proc. of the Tenth International Conference CIIT 2013, Bitola, 2013, pp. 192–196.

[138] **I. Slaminková and M. Vojvoda**, *Cryptanalysis of a hash function based on isotopy of quasigroups*, Tatra Mountains Math. Publ. **45** (2010), 137–149.

[139] **J.D.H. Smith**, *An introduction to quasigroups and their representations*, Chapman and Hall/ CRC, 2006.

[140] **V. Snášel, A. Abraham, J. Dvorsky, P. Krõmer and J. Platoš**, *Hash function based on large quasigroups*, LNCS **5544** (2009), 521–529.

[141] **V. Snášel, J. Dvorsky, E. Ochodkova, P. Krõmer, J. Platoš and A. Abraham**, *Evolving quasigroups by genetic algorithms*, Proc. of DATESO 2010, 2010, pp. 108–117.

[142] **D.R. Stinson**, *Cryptography: Theory and practice*, Second edition. Chapman and Hall / CRC, 2002.

[143] **S. Vaudenay**, *On the need for multipermutations: Cryptanalysis of $MD4$ and $SAFER$*, LNCS **1008** (1995), 286–297.

[144] **M. Vojvoda**, *Cryptanalysis of one hash function based on quasigroup*, Tatra Mountains Math. Publ. **29** (2004), 173–181.

[145] **M. Vojvoda, M. Sýs and M. Jókay**, *A note on algebraic properties of quasigroups in Edon80*, SASC 2007, Bochum, Germany, 2007.

[146] **C. Wolf and B. Preneel**, *Taxonomy of public key schemes based on the problem of multivariate quadratic equations*, IACR Cryptology ePrint Archive, Report 2005/077.

[147] **C. Wolf and B. Preneel**, *MQ*-IP: An identity-based identification scheme without number-theoretic assumptions*, ICAR Cryptology ePrint Archive, Report 2010/087.

[148] **Y. Xu**, *A cryptography application of conjugate quasigroups*, Proc. of the Intern. Confer. on Web Information Systems and Mining 2010, vol. II, Sanya, China, 2010, pp. 63–65.

Received May 5, 2015

Faculty of Computer Science and Engineering, Ss “Cyril and Methodius” University, Skopje, Macedonia
E-mail: smile.markovski@gmail.com