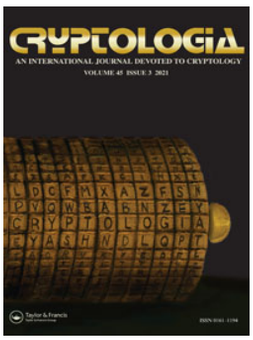

Cryptologia

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/ucry20

Quasigroups and their applications in cryptography

Dimpy Chauhan, Indivar Gupta & Rashmi Verma

To cite this article: Dimpy Chauhan, Indivar Gupta & Rashmi Verma (2021) Quasigroups and their applications in cryptography, Cryptologia, 45:3, 227-265, DOI: 10.1080/01611194.2020.1721615

To link to this article: https://doi.org/10.1080/01611194.2020.1721615

Published online: 01 May 2020.

CRYPTOLOGIA
2021, VOL. 45, NO. 3, 227–265
https://doi.org/10.1080/01611194.2020.1721615

Taylor & Francis
Taylor & Francis Group

REVIEW

Quasigroups and their applications in cryptography

Dimpy Chauhan, Indivar Gupta, and Rashmi Verma

ABSTRACT
Quasigroups have wide applications in coding theory and cryptography. We present a brief survey on quasigroups and discuss their applications in designing various cryptographic primitives including block cipher, stream cipher, hash function, public-key schemes, etc.

KEYWORDS
Block ciphers; cryptographic primitives; hash functions; Latin squares; quasigroup transformation; public-key algorithm; quasigroups; S-boxes; stream ciphers

1. Introduction

Toward the end of the eighteenth century, two seminal papers (Euler 1782, 1849) on pairs of mutually orthogonal Latin squares were published by Euler. It is closely related to the idea of a quasigroup which first appeared in combinatorics. The notion of quasigroup was introduced in 1929 by Suschkewitsch (Suschkewitsch 1929). The term "quasigroup" was first used by Moufang (1935), who considered certain quasigroups with a unity (which were later on named as loops). The structures, properties and existence of large number of quasigroups of certain order enable the quasigroup to be applied in many theories like telecommunications, experimental designs, coding theory, cryptography, etc. Various results related to applications of quasigroups in coding theory and cryptology have been discussed in Dénes and Keedwell (1974, 1991, 2001).

A quasigroup is an algebraic structure similar to a group but it differs from a group in several properties, particularly it is not necessarily associative. But an associative quasigroup forms a group. Hence every group is a quasigroup but a quasigroup need not be a group. Quasigroups are important to cryptography because of two reasons. Firstly, the closure and inversion properties of quasigroups make handy tools for creating cryptosystems. Further, the lack of associativity gives rise to one-way functions in an easier way. The use of the quasigroups provides permutations and substitution networks which are used in various cryptographic structures. These permutations and substitutions are generated more easily and require less device memory. They can even act "locally" on a single block of a plaintext. Reader may refer Belousov (1967, 1972, 1981), Laywine and

CONTACT Dimpy Chauhan dimpychauhan21jan@gmail.com Research scholar, Department of Mathematics, University of Delhi, Delhi-110007, India.

Mullen (1998), Pflugfelder (1990) and Shcherbacov (2003) for basics of quasigroup theory, and Belousov (1967, 1981), Pflugfelder (1990) and Smith (1974) for their detailed study. Applications of quasigroups have been discussed in Bruck (1971), Chein et al. (1990), Shcherbacov (2017) and Syrbu (2014).

The associative algebraic structures such as groups, rings and fields, have been used to construct most of the cryptographic primitives and to derive methods for error detection and correction. Dénes (1979, 2000) and Dénes and Keedwell (1974, 1991, 2001), have done a lot of work in cryptology using non-associative algebraic systems. Quasigroups and their combinatorial equivalent Latin squares are suitable for this purpose due to their structures, features and the fact that they lead to construction of particular and efficient primitives. Latin squares are the equivalent combinatorial structure of quasigroups and have many interesting areas that are worth studying viz. transversals of Latin squares, Mutually Orthogonal Latin Squares (MOLS) and Latin subsquares. For introduction to the theory of Latin square and its applications reader may refer to Belousov and Belyavskaya (1989), Laywine and Mullen (1998), Dénes and Keedwell (1974) and Keedwell and Dénes (2015). The orthogonal property of quasigroups and Latin squares make them appropriate for application in coding theory and cryptology (Dénes and Keedwell 1974). Latin squares have a variety of different practical applications viz. the construction of error correction telegraph codes, in designing statistical experiments, to code messages, generate magic squares, etc.

As discussed earlier quasigroups have been used to realize many cryptographic primitives. The security of the previously designed cryptographic primitives like NASHA was based on large number of quasigroups of the same order, secret quasigroup operations, large numbers of isotopies for a given carrier, etc. However the newer designs like EdonX base their security mostly on the difficulty of various factors such as solving systems of quasigroup equations, finding order of elements in a quasigroup and solving a system of multivariate quadratic functions, etc.

The aim of this article is to illustrate the use of quasigroups and quasigroup transformations in building suitable cryptographic primitives. Section 2 presents a mathematical background on quasigroups and their string transformations including characterization of some new types of quasigroup transformations. In Section 3, several well-known ways of constructing huge quasigroups have been studied. We also discuss the extended Feistel networks introduced by Markovski and Mileva (2009a). In Section 4, we describe the representation of a finite quasigroup as a vector valued Boolean function. Section 5 discusses S-boxes and their cryptographic properties. The construction of some cryptographic primitives based on quasigroups have been explained in Section 6. Conclusions are

finally drawn in Section 7. In this paper, we will not address details on cryptography for which the reader may reference books by Menezes, van Oorschot, and Vanstone (1997), Stinson and Paterson (2019), Katz and Lindell (2008), Beutelspacher (2002), Moldovyan (1998) and Moldovyan, Moldovyan, and Eremeev (2004).

2. Some basic concepts

In this section, we discuss basic mathematical background of quasigroups and their string transformations which is needed for the understanding of further topics. For more mathematical background refer Belousov (1967), Dénes and Keedwell (1974, 1991), Laywine and Mullen (1998) and Smith (1974).

Definition 1. Let (Q, A) be an n-ary groupoid with n-ary operation A. (Q, A) is called an n-ary quasigroup if in the equality $A(x_1, x_2, ..., x_n) = x_{n+1}$ on knowing any n elements of the set $\{x_1, x_2, ..., x_n, x_{n+1}\}$ uniquely specifies the remaining one element.

Definition 2. An n-ary groupoid (Q, A) is said to be a cancellative n-ary groupoid if it satisfies the cancelation law:

$$A(x_1, ..., x_i, a, x_{i+2}, ..., x_n) = A(x_1, ..., x_i, b, x_{i+2}, ..., x_n) \Rightarrow a = b$$

for all $x_i \in Q$ where $i = 1, 2, ..., n$.

Taking $n = 2$ in the Definition 1, n-ary quasigroup is called quasigroup and is defined as below.

Definition 3. Let $(Q, *)$ be a binary groupoid. $(Q, *)$ is called a quasigroup if there exist unique solutions $x, y \in Q$ of the equations $x * a = b$ and $a * y = b$ for all ordered pair $(a, b) \in Q^2$.

Note: From Definition 3 it can be said that every group is a quasigroup but a quasigroup differs from a group in several properties, particularly non-associativity. A quasigroup which satisfies associative law forms a group. Hence sometimes quasigroups are considered as "non-associative groups." Some examples of quasigroups (that are not groups) are $(\mathbb{Z}, -), (\mathbb{Q}\backslash\{0\}, \div)$ and $(\mathbb{R}\backslash\{0\}, \div)$. Any quasigroup having identity element is known as a loop. In this article we shall consider only finite quasigroups.

Definition 4. Let Q be a finite set with cardinality n. A Latin square L on Q is a square matrix of order n with elements from Q such that in any row or column of the matrix none of the element occurs twice.

A Latin square with border form a multiplication table, whereas a multiplication table of a finite quasigroup is just a Latin square. However a

quasigroup has more than one Latin square associated with it since along with its border elements its rows or/and columns can be permuted.

Definition 5. A Groupoid $(G, *)$ is called a solvable groupoid if for all ordered pair $(a, b) \in G^2$, the equations $x * a = b$ and $a * y = b$ have solutions $x, y \in G$.

Definition 6. A Groupoid $(G, *)$ is called a cancellative groupoid, if for every $c, x, y \in G$, $c * x = c * y \Rightarrow x = y$ and $x * c = y * c \Rightarrow x = y$.

For any given quasigroup $(Q, *)$, the operations $/, \backslash, ., //$ and $\backslash\backslash$ on the set Q are as follows:

$$
\begin{aligned}
&x/y = z \iff z * y = x \quad \text{(right division)},\\
&x\backslash y = z \iff x * z = y \quad \text{(left division)},\\
&x.y = z \iff y * x = z \quad \text{(opposite multiplication)},\\
&x//y = z \iff y/x = z \iff z * x = y \quad \text{(opposite right division)}\\
&\text{and } x\backslash\backslash y = z \iff y\backslash x = z \iff y * z = x \quad \text{(opposite left division)}.
\end{aligned}
$$

The set $\mathrm{Par}(*) = \{*, /, \backslash, ., //, \backslash\backslash\}$ is known as the set of *parastrophes* of quasigroup operation $*$ and $|\mathrm{Par}(*)| \leq 6$.

Definition 7. An algebraic quasigroup $(Q, *, \backslash, /)$ is said to be a type (2, 2, 2) algebra if it satisfies the identities:

$$
\begin{cases}
y = x * (x\backslash y)\\
y = x\backslash(x * y)\\
y = (y/x) * x\\
y = (y * x)/x.
\end{cases}
\tag{1}
$$

Definition 8. Let $(G, \cdot)$ be a groupoid and let a be a fixed element in G. The left and the right translations denoted by L_a and R_a are defined as $L_a(x) = a \cdot x$ and $R_a(x) = x \cdot a$ respectively, for each $a \in G$.

There exist one more translation known as middle translation. Let P_a be a middle translation of a quasigroup $(G, \cdot)$ then for all $x \in G$, $x \cdot P_a(x) = a$.

In Pflugfelder (1990), Pflugfelder has given the following definition of quasigroup:

Definition 9. A groupoid $(Q, *)$ is called a quasigroup if for all $a \in Q$, $L_a : Q \to Q, R_a : Q \to Q$ are bijections maps.

For all $a, b \in Q$, the equation $x * a = b$ has a unique solution x in Q $\iff R_a$ is the bijection map on Q.

For all $a, b \in Q$, the equation $a * y = b$ has a unique solution y in $Q \iff L_a$ is the bijection of the set Q.

Definition 10. n-ary groupoids $(Q, f_1), (Q, f_2), ..., (Q, f_n)$ are called orthogonal, if the following system of equation:

$$\begin{cases} f_1(x_1, x_2, ..., x_n) = a_1 \\ f_2(x_1, x_2, ..., x_n) = a_2 \\ ... \\ f_n(x_1, x_2, ..., x_n) = a_n \end{cases} \tag{2}$$

has unique solution for any fixed n-tuple $a_1, a_2, ..., a_n \in Q$.

2.1. Quasigroup string transformations

Markovski, Gligoroski, and Andova (1997) introduced quasigroup string transformations which were later investigated by several other researchers (Markovski 2003; Markovski and Bakeva 2001; Markovski, Gligoroski, and Bakeva 1999; Markovski and Kusakatov 2000, 2002–2003).

Let $(Q, *)$ be a finite quasigroup and $Q^+ = \{x_1 x_2 ... x_n \mid x_i \in Q, n \geq 1\}$ denotes the set of all finite string over Q (also denoted by Q^n and elements by $(x_1, x_2, ..., x_n)$). For all $l \in Q$, two functions $e_{l,*} : Q^+ \to Q^+$ and $d_{l,*} : Q^+ \to Q^+$ are defined as follows:

$$e_{l,*}(x_1 x_2 ... x_n) = (y_1 y_2 ... y_n) \iff y_j = \begin{cases} l * x_1, & j = 1 \\ y_{j-1} * x_j, & 2 \leq j \leq n \end{cases} \tag{3}$$

and

$$d_{l,*}(x_1 x_2 ... x_n) = (z_1 z_2 ... z_n) \iff z_j = \begin{cases} l * x_1, & j = 1 \\ x_{j-1} * x_j, & 2 \leq j \leq n. \end{cases} \tag{4}$$

These functions are called *elementary quasigroup transformation* or *e*- and *d*- transformation of Q^+ with leader l. Their graphical representation is shown in Figures 1 and 2.

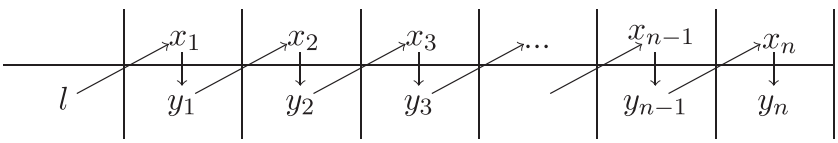

Figure 1. Graphical representation of $e_{l,*}$ function.

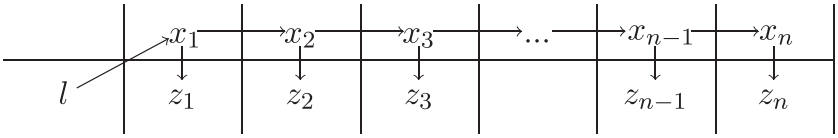

Figure 2. Graphical representation of $d_{l,*}$ function.

Table 1. A quasigroup $(Q, *)$ and its parastrophes $(Q, \backslash)$, $(Q, /)$.

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

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

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

Example 2.1. Let $Q = \mathbb{Z}_4 = \{0, 1, 2, 3\}$ and the quasigroup $(Q, *)$ and its parastrophes $(Q, \backslash)$, $(Q, /)$ be as shown in Table 1. Consider a string in Q^+, say $x = 1\ 2\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 2\ 2\ 3\ 0\ 1\ 0\ 0\ 0\ 3\ 1\ 1\ 0\ 0$ and 1 be a leader. Then by applying the transformation $e_{l,*}$ and $d_{l,*}$ we will get the following strings:

$e_{l,*}(x) =\ 2\ 2\ 1\ 2\ 3\ 2\ 1\ 2\ 1\ 0\ 3\ 0\ 1\ 1\ 0\ 0\ 3\ 2\ 1\ 1\ 2\ 3\ 2\ 1$

and $d_{l,*}(x) = 2\ 3\ 1\ 0\ 2\ 0\ 3\ 0\ 0\ 3\ 3\ 1\ 2\ 0\ 2\ 0\ 0\ 3\ 3\ 2\ 1\ 2\ 0\ 3.$

Also $d_{l,\backslash} = 3\ 1\ 3\ 2\ 3\ 0\ 1\ 2\ 0\ 1\ 1\ 3\ 2\ 1\ 2\ 2\ 0\ 1\ 1\ 0\ 1\ 3\ 0\ 1.$
Hence $d_{l,\backslash}(e_{l,*}(x)) = 1\ 2\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 2\ 2\ 3\ 0\ 1\ 0\ 0\ 0\ 3\ 1\ 1\ 0\ 0 = x$ and $e_{l,*}(d_{l,\backslash}(x)) = 1\ 2\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 2\ 2\ 3\ 0\ 1\ 0\ 0\ 0\ 3\ 1\ 1\ 0\ 0 = x$ i.e. $e_{l,*}$ and $d_{l,\backslash}$ are inverse transformations on Q^+.

From the fact that the algebra $(Q, *, \backslash, /)$ satisfies the identities (1), the following property holds:

Theorem 2.1. (Markovski, Gligoroski, and Bakeva 1999) *Let $(Q, *, \backslash, /)$ be a finite quasigroup. Then for each string $x \in Q^+$ and for each leader $l \in Q$ we have that $e_{l,*}$ and $d_{l,\backslash}$ are mutually inverse permutations of Q^+, i.e. $d_{l,\backslash}(e_{l,*}(x)) = x = e_{l,*}(d_{l,\backslash}(x))$.*

From Theorem 2.1 we can conclude that the transformations $e_{l,*}$ and $d_{l,\backslash}$ can be used for defining suitable encryption and decryption functions.

Let $x = x_1x_2...x_n \in Q^+$ and $l \in Q$. Similar string transformations $e'_{l,*} : Q^+ \to Q^+$ and $d'_{l,*} : Q^+ \to Q^+$ are defined as follows:

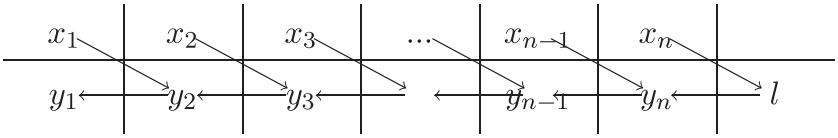

Figure 3. Graphical representation of $e'_{l,*}$ function.

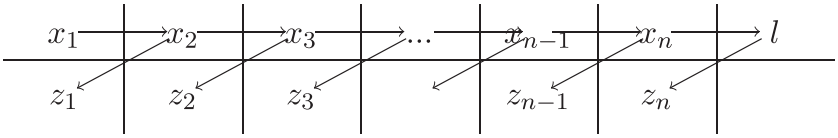

Figure 4. Graphical representation of $d'_{l,*}$ function.

$$e'_{l,*}(x_1x_2\ldots x_n) = (y_1y_2\ldots y_n) \Longleftrightarrow y_j = \begin{cases} x_n * l, & j = n \\ x_j * y_{j+1}, & 1 \le j \le n-1 \end{cases} \tag{5}$$

and

$$d'_{l,*}(x_1x_2\ldots x_n) = (z_1z_2\ldots z_n) \Longleftrightarrow z_j = \begin{cases} x_n * l, & j = n \\ x_j * x_{j+1}, & 1 \le j \le n-1. \end{cases} \tag{6}$$

Their graphical representation is shown in Figures 3 and 4.

Theorem 2.1 also holds for the transformations $e'_{l,*}$ and $d'_{l,\backslash}$. Hence these transformations can also be used for defining suitable encryption and decryption functions.

Markovski, Gligoroski, and Bakeva (1999) defined the composition of elementary quasigroup transformations known as *composite quasigroup transformation* as follows:

Let $*_1, *_2, \ldots, *_n$ be n quasigroup operations on Q and let for $i = 1, 2, \ldots, n$, $e_{l_i}, d_{l_i}, e'_{l_i}, d'_{l_i}$ be the transformations defined as in (3, 4, 5, 6) by choosing fixed leaders $l_1, l_2, \ldots, l_n \in Q$. Let t_{l_i} be any of the transformations $e_{l_i}, d_{l_i}, e'_{l_i}, d'_{l_i}$, then E, D, E', D' *and* T can be defined as:

$$E = E^{(n)}_{l_n,\ldots,l_1} = e_{l_n} \circ e_{l_{n-1}} \circ \cdots \circ e_{l_1}, \tag{7}$$

$$D = D^{(n)}_{l_n,\ldots,l_1} = d_{l_n} \circ d_{l_{n-1}} \circ \cdots \circ d_{l_1}, \tag{8}$$

$$E' = E'^{(n)}_{l_n,\ldots,l_1} = e'_{l_n} \circ e'_{l_{n-1}} \circ \cdots \circ e'_{l_1}, \tag{9}$$

$$D' = D'^{(n)}_{l_n,\ldots,l_1} = d'_{l_n} \circ d'_{l_{n-1}} \circ \cdots \circ d'_{l_1}, \tag{10}$$

and

$$T = T^{(n)}_{l_n,\ldots,l_1} = t_{l_n} \circ t_{l_{n-1}} \circ \cdots \circ t_{l_1} \tag{11}$$

There is an e-transformation in which the leaders are the elements of the input string taken in reverse order, namely reverse string transformation which was introduced by Gligoroski (2004) and is defined as follows:

Definition 11. Let n be a positive integer, $(Q, *)$ be a quasigroup and $x_j \in Q$, $1 \leq j \leq n$. Quasigroup reverse string transformation $\mathcal{R} : Q^n \to Q^n$ is defined as composition of e-transformations in following way:

$$\mathcal{R}(x_1, x_2, ..., x_n) = (e_{x_1} \circ e_{x_2} \circ ... \circ e_{x_n})(x_1, x_2, ..., x_n).$$

2.2. Some important quasigroup transformations

Definition 12. (Markovski and Mileva 2009b) A groupoid $(G, \cdot)$ is said to be a left quasigroup (a right quasigroup) if the equation $x \cdot a = b$ $(a \cdot y = b)$ have a unique solution $x(y)$ in G for every $a, b \in G$.

Proposition 1. (Markovski and Mileva 2009b) *Let $(G, +)$ be a group and let $(G, *)$ be a quasigroup. Then the operation $\bullet$ defined by $x \bullet y = (x + y) * y$ defines a left quasigroup $(G, \bullet)$.*

Proposition 2. (Markovski and Mileva 2009b) *Let $(G, +)$ be a group and let $(G, *)$ be a quasigroup. Then the operation $\diamond$ defined by $x \diamond y = x * (x + y)$ defines a right quasigroup $(G, \diamond)$.*

Some new types of quasigroup transformations have been defined for constructing cryptographic primitives. We will discuss some of these primitives in Section 6.

Definition 13. (Mileva and Markovski 2010) Let $(Q, *_1)$ and $(Q, *_2)$ be two orthogonal quasigroups. Then orthogonal quasigroup string transformation is a map $OT : Q^+ \to Q^+$ defined 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 $OT(x_1, x_2, ..., x_{t-2}, x_{t-1}) = (z_1, z_2, ..., z_{t-1})$ is defined for $t > 2$, then

$OT(x_1, x_2, ..., x_{t-1}, x_t) = (z_1, z_2, ..., z_{t-1} *_1 x_t, z_{t-1} *_2 x_t)$, where $x_i \in Q$.

In 2008, Markovski and Mileva defined quasigroup additive string transformation, quasigroup reverse additive string transformation and quasigroup main transformation which were used in the design of NaSHA hash family (Markovski and Mileva 2008). More details on NaSHA hash family is described in Section 6.5.

Let $G = \mathbb{Z}_{2^n}$, $(G, *)$ be a quasigroup and "+" denotes the addition modulo 2^n.

Definition 14. Quasigroup additive string transformation $\mathcal{A}_l : G^+ \to G^+$ with leader l is the transformation defined as follows:

$$\mathcal{A}_l(x_1x_2...x_t) = (z_1z_2...z_t) \iff z_j = \begin{cases} (l + x_1) * x_1, j = 1 \\ (z_{j-1} + x_j) * x_j, 2 \le j \le t \end{cases}$$

where $x_i, z_i \in G, t \ge 1$.

Definition 15. Quasigroup reverse additive string transformation $\mathcal{RA}_l : G^+ \to G^+$ with leader l is the transformation defined as follows:

$$\mathcal{RA}_l(x_1x_2...x_t) = (z_1z_2...z_t) \iff z_j = \begin{cases} x_j * (x_j + z_{j+1}), 1 \le j \le t-1 \\ x_t * (x_t + l), j = t \end{cases}$$

where $x_i, z_i \in G, t \ge 1$.

Let for fixed elements $l_1, l_2, ..., l_s \in G$, $\mathcal{A}_{l_i}$ and $\mathcal{RA}_{l_i}$ be transformations defined as above. Let for $i = 1, 2, ..., s$, m_{l_i} denotes any of the previous transformations $\mathcal{A}_{l_i}$, $\mathcal{RA}_{l_i}$. Define a transformation M as:

$$M = m_{l_1} \circ m_{l_2} \circ \cdots \circ m_{l_s}.$$

Let for each element $z \in G = \mathbb{Z}_{2^n}$, $\rho(z, \lfloor \frac{n}{2} \rfloor)$ denotes the element of G obtained by rotating the n-bit representation of z to the left by $\lfloor \frac{n}{2} \rfloor$ bits. For a given string $Z = (z_1z_2...z_t) \in G^t$, $\rho(Z)$ denotes the string given by:

$$\rho(Z) = \left(\rho\left(z_1, \left\lfloor \frac{n}{2} \right\rfloor\right)...\rho\left(z_t, \left\lfloor \frac{n}{2} \right\rfloor\right)\right) \in G^t.$$

A function $\rho(f) = \rho(f)(Z)$ was defined for $f = f(Z)$ as $\rho(f)(Z) = f(\rho(Z))$.

Definition 16. Quasigroup main transformation $\mathcal{MT} : G^+ \to G^+$ with complexity k is defined as:

For each $x_i \in G$,

$$\mathcal{MT}(x_1x_2...x_t) = \rho(\mathcal{RA}_{l_1})(\mathcal{A}_{l_2}(...(\rho(\mathcal{RA}_{l_{k-1}})(\mathcal{A}_{l_k}(x_1x_2...x_t)))...))$$

i.e. as composition of transformations of the type $\mathcal{A}_{l_i}$ followed by the function $\rho(\mathcal{RA}_{l_j})$, where the leaders l_i and l_j are suitably chosen as a function depending on variables $x_1, x_2, ..., x_t$. Here $\mathcal{MT} = \rho(\mathcal{RA}_{l_1}) \circ \mathcal{A}_{l_2} \circ ...\rho(\mathcal{RA}_{l_{k-1}}) \circ \mathcal{A}_{l_k}$, where $\circ$ denotes a composition operation.

Note:

$$\begin{aligned} \rho(\mathcal{RA}_{l_{k-1}}) \circ \mathcal{A}_{l_k}(x_1x_2...x_t) &= \mathcal{RA}_{l_{k-1}}(\rho(\mathcal{A}_{l_k}(x_1x_2...x_t)) \\ &= \mathcal{RA}_{l_{k-1}}(z_1z_2...z_t) \end{aligned}$$

and so on, where $\mathcal{A}_{l_k}(x_1x_2...x_t) = (z_1z_2...z_t)$.

3. Generation of huge quasigroups

With the help of Latin square, a quasigroup can be constructed. Small order quasigroup can be easily represented by its multiplicative table. But this cannot be done for quasigroup of huge order such as $2^{16}, 2^{64}, 2^{128}, 2^{256}, 2^{512}, \ldots$ (known as huge quasigroups) that are used in the construction of some cryptographic primitives. Some of the known constructions of huge quasigroups are discussed as follows.

3.1. Generation of huge quasigroups using extended Feistel networks

Definition 17. (Dénes and Keedwell 1991) A complete mapping of a group, quasigroup or loop $(G,+)$ is a bijection mapping $\phi : G \to G$ such that the mapping $\theta : G \to G$ defined by $\theta(x) = x + \phi(x)$ ($\theta = I + \phi$, where I is the identity mapping) is again a bijective mapping of G. The mapping θ is called orthomorphism associated to the complete mapping ϕ and group (quasigroup) G is admissible if there exist a complete mapping $\phi : G \to G$.

Extended Feistel networks $F_{a,b,c}$ are defined by Markovski and Mileva (2009a) as:

Let $(G,+)$ be an abelian group and let $f : G \to G$ be a function. Let $a, b, c \in G$ be constants. The extended Feistel network $F_{a,b,c} : G^2 \to G^2$ generated by f is defined as:

$$F_{a,b,c}(l,r) = (r + a, l + b + f(r + c)), \qquad \text{for every} \quad (l,r) \in G^2.$$

If f is a bijection mapping then $F_{a,b,c}$ is an orthomorphism of the group $(G^2,+)$ (in other words $F_{a,b,c}$ and $F_{a,b,c} - I$ are bijections), so by Sade's diagonal method (Sade 1957) a quasigroup $(G^2, *_{F_{a,b,c}})$ is defined as:

$$X *_{F_{a,b,c}} Y = F_{a,b,c}(X - Y) + Y$$

The parameters *a*,*b*,*c* of $F_{a,b,c}$ can be used for different purposes so this construction is suitable for many applications. A quasigroup of large order (i.e. huge quasigroup) can be constructed from a group of small order and using iteration. This can be done as follows:
Let f be a bijection on G from which a bijection map $f_1 = F_{a,b,c}$ is generated on G^2. By choosing constants $a_1, b_1, c_1 \in G^2$, a suitable extended Feistel network F_{a_1,b_1,c_1} can again be generated and a quasigroup $(G^4, *_{F_{a_1,b_1,c_1}})$ of order $|G|^4$ can be produced by using Sade's diagonal method. Continuing like this, a quasigroup having order $|G|^{2^k}$ gets generated after k steps.

Note that the only thing that is to be kept in memory is the starting bijection f. Some more constructions of quasigroups by using different types of Feistel networks are given by Mileva and Markovski (2012).

3.2. Generation of huge quasigroups using T-functions

Definition 18. (Klimov and Shamir 2002) T-function (T stands for triangular) is a mapping that associates an n-bit input to an n-bit output in which ith bit of the output can depend only on the input bits $0, 1, 2, \ldots, i$.

Most of the arithmetical operations and all the Boolean operators that are available on modern processors, and their compositions are examples of T-functions. The definition of T-function can be extended to a function from several n-bit inputs to several n-bit outputs. Circular rotations and right shifts are not T-functions.

Huge quasigroups have been constructed by using T-functions by Courtois et al. (2000) and one more way of achieving huge quasigroups is given by Samardziska, Markovski, and Gligoroski (2010) as follows:

Let $Q = \mathbb{Z}_{2^s}$ and let the elements of Q be represented in binary as bit strings of length s. Let $\mathbf{b}$ be a constant Boolean vector and $x_s, \ldots, x_1$ be Boolean variables. Let $\mathbf{A}_1 = [f_{ij}]_{s\times s}$ and $\mathbf{A}_2 = [g_{ij}]_{s\times s}$ with variables $x_s, \ldots, x_1$ be upper triangular matrices of linear Boolean expressions such that the following holds:

(i) $f_{ii} = 1$, $g_{ii} = 1$ and f_{is} are constants, $\forall\ i = 1, 2, \ldots, s$.
(ii) f_{ij} can only depend on the variables $x_{s-j}, \ldots, x_1\ \forall\ i < j < s$.
(iii) g_{ij} can only depend on the variables $x_s, \ldots, x_1$, $\forall\ i < j$.

Let the binary representation of the variables x and y over Q be $x = (x_s, \ldots, x_1)$ and $y = (y_s, \ldots, y_1)$. Then $(Q, *)$ is a quasigroup of order 2^s, where $*$ is defined as:

$$x * y = \mathbf{A}_1.(x_s, \ldots, x_1)^T + \mathbf{A}_2.(y_s, \ldots, y_1)^T + \mathbf{b}^T.$$

The parastrophe $(Q, \backslash)$ is defined as:

$$x\backslash y = \mathbf{A}_2^{-1}.((y_s, \ldots, y_1)^T - \mathbf{A}_1.(x_s, \ldots, x_1)^T - \mathbf{b}^T).$$

3.3. Generation of huge quasigroups using simple isotopies

Two huge quasigroups of order 2^{256} and 2^{512} that are used in the compression function of the hash function Edon-R given by Gligorovski et al. (2008) have been discussed in Section 6.5. Their operations are defined by isotopies of the abelian group $((\mathbb{Z}_{2^s})^8, +_8)$, where $s = 32$ and $s = 64$,

respectively. ($+_8$ denotes component-wise addition on two 8-dimensional vectors in $(\mathbb{Z}_{2^s})^8$). The quasigroup operation $*$ is defined as:

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

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

4. Quasigroups as vector valued Boolean functions

Let $\mathbb{F}_2$ denotes a Galois field with two elements. A map $f : \mathbb{F}_2^k \to \mathbb{F}_2^t, (t \geq 1)$ is known as a *vector valued Boolean function*. Any such function can be written as t k-ary *Boolean functions* which are nothing but functions $f_i : \mathbb{F}_2^k \to \mathbb{F}_2$, as follows:

$$f(x_1, ..., x_k) = (f_1(x_1, ..., x_k), f_2(x_1, ..., x_k), ..., f_t(x_1, ..., x_k)),$$

where

$$f_1(x_1, ..., x_k) = y_1, ..., f_t(x_1, ..., x_k) = y_t \iff f(x_1, ..., x_k) = (y_1, ..., y_t).$$

Every Boolean function of k variables $f_i(x_1, x_2, ..., x_k)$ can uniquely be represented in Algebraic Normal Form(ANF) as:

$$f_i(x_1, x_2, ..., x_k) = \sum_{I \subseteq \{1, 2, ..., k\}} \alpha_I \left(\prod_{i \in I} x_i \right)$$

where the coefficients $\alpha_I \in \mathbb{F}_2$ and the addition & multiplication are over $\mathbb{F}_2$. This representation gives the algebraic degree of the vector valued Boolean function which is the maximum of the degrees of its component polynomials i.e.

$$\deg(f) = max\{\deg(f_i) | i \in \{1, 2, ..., k\}\}. \tag{12}$$

Every quasigroup $(Q, *)$ of order n where $n = 2^d$ and $n \geq 2$ can be represented as a vector valued Boolean function (Gligoroski, Dimitrova, and Markovski 2009). In other words a quasigroup can be represented as a map $f : \mathbb{F}_2^{2d} \to \mathbb{F}_2^d$. Thus for each $x, y, z \in Q$, with binary representations $(x_1, x_2, ..., x_d)$, $(y_1, y_2, ..., y_d)$ and $(z_1, z_2, ..., z_d)$ respectively, the operation $x * y = z$ can be represented as:

$$\begin{aligned} f(x_1, x_2, ..., x_d, y_1, y_2, ..., y_d) &= (z_1, z_2, ..., z_d) \\ &= (f_1(x_1, ..., x_d, y_1, ..., y_d), ..., f_d(x_1, ..., x_d, y_1, ..., y_d)) \end{aligned}$$

where for each $i = 1, 2, ..., d$, $z_i = f_i(x_1, ..., x_d, y_1, ..., y_d)$ and $f_i : \mathbb{F}_2^{2d} \to \mathbb{F}_2$. f is a linear function if $deg(f) = 1$. Linear and affine function may also be defined as follows:

Definition 19. Let $(G, +)$ be a group. A function $f : G \to G$ is said to be an affine function if for each $x, y \in G$, $f(x+y) = f(x) + f(y) - f(0)$, where $0 \in G$ is the identity element. An affine function f with $f(0) = 0$ is a linear function.

According to the algebraic degree, quasigroups can be divided into two disjoint classes viz. the linear and the non-linear quasigroups (Dimitrova 2010; Gligoroski, Dimitrova, and Markovski 2009). *Linear quasigroups* are those quasigroups which have all of its component Boolean functions linear i.e. of degree 1 and has algebraic degree 1. Otherwise the quasigroups (having greater than 1 algebraic degree) belong to the class of *non-linear quasigroups*. For generating non-linear cryptographic primitives it is not a good practice to use a quasigroup with any linear component Boolean function. Non-linear quasigroups can further be classified into *weak non-linear* and *pure non-linear quasigroups*. A quasigroup is called weak non-linear quasigroup if it has at least one component Boolean function of degree 1 and at least one component Boolean function of degree greater than 1. A pure non-linear quasigroup is the one which has all its component Boolean functions of degree greater than 1.

Definition 20. (Gligoroski, Markovski, and Knapskog 2008b) 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 the polynomials f_i are of degree 2 (i.e. quadratic) and k of them are of degree 1 (i.e. linear) where $0 \leq k < d$.

5. S-boxes and their cryptographic properties

There does not exist any formal definition for S-boxes but they are defined as lookup tables and that can be interpreted as Boolean maps or vector valued Boolean functions. Any given S-box that maps n bits to m bits can be presented as a Boolean map $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$, where $\mathbb{F}_2$ is a Galois field with two elements. It can be represented by m Boolean functions of n variables.

Now we discuss some cryptographic properties of S-boxes. For detailed study the reader may reference Wu and Feng (2016). Let $\mathcal{F}_n$ be the set of all Boolean functions in n variables, $\mathcal{L}_n \subset \mathcal{F}_n$ be the set of linear Boolean functions and $\mathcal{A}_n \subset \mathcal{F}_n$ be the set of affine Boolean functions. Also $\mathcal{L}_n \subset \mathcal{A}_n \subset \mathcal{F}_n$.

A Boolean function is said to be *balanced* if its truth table has equal number of zeroes and ones. This is one of the criterion which a Boolean function should satisfy.

Let $f(x)$, $g(x) \in \mathcal{F}_n$, then the distance between $f(x)$ and $g(x)$ is the number of coordinates in which their truth tables differs and is denoted by $d(f, g)$.

The *non-linearity* of a Boolean function $f(x) \in \mathcal{F}_n$, denoted by N_f, is defined as the minimum distance between $f(x)$ and the set of all affine functions i.e.

$$N_f = min\{d(f(x), l(x)) : l(x) \in \mathcal{A}_n\}.$$

Let $f(x) \in \mathcal{F}_n$, then $f(x)$ is said to satisfy the *propagation criterion* if $f(x) \oplus f(x \oplus \alpha)$ is always a balanced Boolean function for any nonzero vector $\alpha \in \mathbb{F}_2^n$. A Boolean function $f(x) \in \mathcal{F}_n$ satisfies the *strict avalanche criterion* (SAC), if for all $\alpha \in \mathbb{F}_2^n$ such that $wt(\alpha) = 1$, $f(x) \oplus f(x \oplus \alpha)$ is always balanced (where $wt(\alpha)$ is the hamming weight of α). Further $f(x) \in \mathcal{F}_n$ is called *perfect non-linear* if $f(x) \oplus f(x \oplus \alpha)$ is always a balanced Boolean function for any nonzero vector $\alpha \in \mathbb{F}_2^n$. Note that SAC is the propagation criterion of order 1.

Let $x, y \in \mathbb{F}_2^n$ be such that $x = (x_0, x_1, ..., x_{n-1}), y = (y_0, y_1, ..., y_{n-1})$, then the canonical dot product of x and y can be written as $x.y = \sum_{i=0}^{n-1} x_i y_i$.

According to the results in Pommering (2005) on linearity of Boolean maps, the *linearity* of an S-box can be defined as:

$$Lin(S) = max\left\{\frac{1}{2^{2n}} S^W(u, v)^2 : u \in \mathbb{F}_2^n, v \in \mathbb{F}_2^m, (u, v) \neq 0\right\} \tag{13}$$

where u and v are the part of input and output values respectively of the S-box.

The *Walsh spectrum* S^W of this S-box is defined in Pommering (2005) as:

$$S^W(u, v) = \sum_{x \in \mathbb{F}_2^n} (-1)^{u.x + v.S(x)}. \tag{14}$$

The linearity of S-box is a property that gives a measure for the resistance of S against linear cryptanalysis. It has been proved that $Lin(S) \geq \frac{1}{2^n}$ (Chabaud and Vaudenay 1995).

The *differential potential* of an S-box is defined in Pommering (2005) as:

$$Diff(S) = max\left\{\frac{1}{2^n} \Delta_S(u, v)^2 : u \in \mathbb{F}_2^n, v \in \mathbb{F}_2^m, (u, v) \neq 0\right\} \tag{15}$$

where

$$\Delta_S(u, v) = |\{x \in \mathbb{F}_2^n : S(x \oplus u) = S(x) \oplus v\}|. \tag{16}$$

Diff(S) of an S-box is used to measure the resistance against differential cryptanalysis. For any S-box, $Diff(S) \geq \frac{1}{2^m}$.

6. Cryptographic primitives based on quasigroups

Mileva and Markovski (2012) defined Shapeless Quasigroup as follows:

Definition 21. A quasigroup $(Q, *)$ of order r is said to be shapeless iff it is non-associative, non-commutative, non-idempotent, it does not contain proper sub-quasigroups, it does not have neither left nor right unit, and there is no $k < 2r$ for which identities of the kinds are satisfied:

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

They proposed various constructions of huge shapeless quasigroups. The idea of shapeless quasigroup was extended to perfect quasigroup to improve security against linear and differential attacks.

Definition 22. (Mileva 2010) A quasigroup $(Q, *)$ of order r is said to be perfect if it is pure non-linear, non-correlated and restricted shapeless.

According to the properties of quasigroups, for every n, the set of all quasigroups Q_n can be classified into two disjoint classes viz. the class of fractal and the class of non-fractal quasigroups (Dimitrova and Markovski 2007; Markovski, Dimitrova, and Samardjiska 2010). Q_n can also be classified into two disjoint classes by taking into consideration how the period of the string $e^t_{l,*}(\beta)$ is growing for a periodic string β in a quasigroup. These two classes are the class of exponential and the class of linear quasigroups. If the period of the string $e^t_{l,*}(\beta)$ of a quasigroup is bounded below by an exponential function constant $c.2^{at}$, for some positive constants c and a, then the quasigroup is called exponential quasigroup otherwise it is a linear quasigroup. This has been discussed in more detail in Dimitrova and Markovski (2004), Markovski (2003) and Markovski, Gligoroski, and Kocarev (2005). In the subsequent section we discuss constructions of various cryptographic primitives based on quasigroups.

6.1. S-boxes defined by quasigroups

S-boxes, in almost all modern block ciphers are the only non-linear part and hence play a fundamental role in the design of a cryptosystem. In order to make the cipher secure against various types of attacks, the S-boxes should be chosen very carefully. Confusion of the input data is the main purpose of the S-boxes. S-boxes have a slight amount of data, therefore in order to satisfy cryptographic properties, the S-boxes construction should be made carefully.

Quasigroup also works as an S-box. For example, a quasigroup of order 4 can be considered as a 4×2 bits S-box with 2×2 bits as input and 2 bits

output. Consider a quasigroup as shown below and suppose we have a 4-bits input $B = 0110$ to the S-box then the output will be computed as $S(B) = 01 * 10 = 1 * 2 = 1 = 01$ $(=$ output) is the 2-bits output.

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

There are 576 quasigroups of order 4, out of which 144 are linear and 432 are non-linear. Only non-linear ones are used to construct quasigroup based S-boxes (known as Q-S-boxes).

Bogdanov et al. (2007) proposed an ultra-light block cipher PRESENT. The S-boxes in PRESENT is resultant of the complete search of 16! bijective 4×4 bit S-boxes. Only 16 different non-equivalence classes were obtained and all the S-box members that these classes contain are optimal with respect to linear and differential properties. Instead of the complete search of 16! bijections Mihajloska and Gligoroski (2012) designed a fast and elegant technique that construct cryptographically strong Q-S-boxes by using quasigroup of order 4. In their investigation they chose to work with 2, 4 different leaders and 4 rounds, and 8 different leaders and 8 rounds and proposed the algorithm given in Table 2.

The algorithm ends up with a 4×4 bit Q-S-box and in order to get optimal Q-S-box, the conditions of linearity and differential potential were also checked. At least 4 number of rounds were used, and using the described algorithm Q-S-boxes were generated in different ways that depend on the chosen number of rounds and leaders. Figure 5 shows the working of the algorithm for 4 rounds.

Table 2. Construction of one Q-S-box.

Algorithm for generation of Q-S-boxes
Input: A non-linear quasigroup $(Q, *)$ of order 4, a positive integer r-the number rounds, the leaders $l \in Q$ (usually, their number is the same as the number of rounds).
Output: Optimal Q-S box.
Step 1: Generate all possible input blocks of 4 bits in the lexicographic ordering (they are 2^4 in number);
Step 2:Take input blocks one by one, and for each of the 2^4 input do:
Step 2.1: On the input block apply e-transformation with leader l;
Step 2.2: Reverse the result obtained from 2.1 and again apply e-transformation with leader l;
Step 2.3: Repeat 2.1 and 2.2 r times;
Step 2.4: Save the result of the last round;
Step 3: Concatenate all saved results which generate permutation of order 16 or 4×4-bit Q-S-box;
Step 4: Check if linearity $= \frac{1}{4}$ and differential potential $= \frac{1}{4}$ (as per Equations (13) and (15)):
Step 4.1: If yes, put it in the set of optimal Q-S-boxes;
Step 4.2: If not, input l and go to step 1;
Step 5: Analyze the optimal set of newly obtained Q-S-boxes;

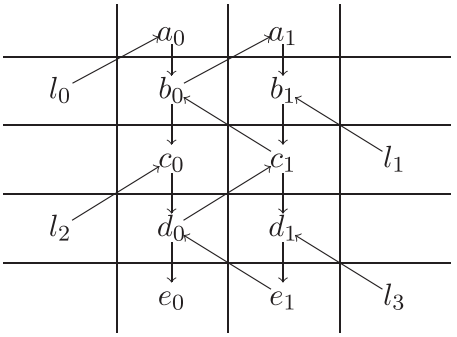

Figure 5. Graphical representation of 4 *e*-transformations with leaders l_0, l_1, l_2, l_3.

Table 3. Example of a Q-S-box given in its hexadecimal notation.

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

One representative of Q-S-boxes obtained from the algorithm in which all output bits have 3 algebraic degree is given in Table 3 (taken from Mihajloska and Gligoroski (2012)).

One more method for generating 4×4 bit S-box has been proposed by Yu and Xu (2017). In their method they worked on pure non-linear 3-quasigroups of order 4.

6.2. Block ciphers

A block cipher is simply a function that maps blocks of n-bit plaintext into blocks of n-bit ciphertext (n is known as the block length). By using a secret key, a block cipher encrypts a block of plaintext into a block of ciphertext having the same length. Let $\mathcal{P}$, $\mathcal{C}$ and $\mathcal{K}$ denote the spaces of plaintext, ciphertext and keys respectively. Let $M \in \mathcal{P}$ and $C \in \mathcal{C}$. An n-bit block cipher is an invertible mapping $E : \mathcal{P} \times \mathcal{K} \rightarrow \mathcal{C}$ (encryption function) such that for all $K \in \mathcal{K}$, $E(M, K) = C$ and $D : \mathcal{C} \times \mathcal{K} \rightarrow \mathcal{P}$ is the decryption function, denoted as $D(C, K)$ such that $D(E(M, K), K) = M$.

If $\mathcal{P} = \mathcal{C} = \mathcal{K} = \{0, 1\}^n$, then the quasigroup operation is considered as an encryption function E and its parastrophe as the decryption function D. Here we show the design of the block cipher BCDMPQ (Block Cipher Defined by Matrix Presentation of Quasigroups) given by Markovski et al. (2014).

The matrix presentation of quasigroups of order 4 (Simjanovska, Mihova, and Markovski 2013) is used in the design of BCDMPQ. A matrix representation of a given 4 order quasigroup $(Q, *)$ has the form:

$$x * y = m^T + Ax^T + By^T + CAx^T \circ CBy^T \tag{17}$$

for all $x = (x_1, x_2)$, $y = (y_1, y_2) \in Q$ (x_i, y_i are bits), where $m = \begin{bmatrix} m_1 & m_2 \end{bmatrix}$ is some constant Boolean vector from Q, $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ and $B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$ are nonsingular Boolean matrices and $C = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$. The operation "$\circ$" in the equation denotes the multiplication of two vectors component wise. Here the addition and multiplication are over the field $GF(2)$.

Out of 144 quasigroups of form (17), 128 are chosen and stored in memory as s_{num}, $m_1, m_2, a_{11}, a_{12}, a_{21}, a_{22}, b_{11}, b_{12}, b_{21}, b_{22}$, where s_{num} (a 7 bit number) denotes the number of quasigroup and m_1, m_2, $a_{11}, a_{12}, a_{21}, a_{22}$, $b_{11}, b_{12}, b_{21}, b_{22}$ are the bits that arise in (17). 16 quasigroups are used in the encryption and decryption algorithm in different steps, namely $Q_1, Q_2, ..., Q_8$, $T_1, ..., T_8$. The key of length 128 bits is shared as: 56 bits for quasigroups $Q_1, Q_2, ..., Q_8$ and $T_1, T_2, ..., T_8$ (7 bits for each quasigroup) and 16 bits for leaders $l_1, l_2, ..., l_8$ (two bits for each leader).

The design of BCDMPQ consists of 3 algorithms viz. Round key generation, Encryption and Decryption algorithm. In Table 4, we describe the algorithm for the generation of round key out of the secret key of which a fixed leader $l = 0 = [0, 0]$ and fixed shapeless quasigroup Q are taken. Let K be the secret symmetric key of 128 bits, *tmp* and *ltmp* are two auxiliary variables.

For any n, the block length of the message of BCDMPQ can be $8n$. But for the light version of the cipher, $n = 8$ is taken. Therefore, the plaintext will split into 64 bits blocks, and on each of these blocks, encryption

Table 4. Round key generation algorithm of BCDMPQ.

Round key generation algorithm
Input: The secret key $K = K_1K_2...K_{128}$, K_i are bits. **Output**: The round key $k = k_1k_2...k_{128}$, 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 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 do
tmp $\leftarrow (k_{j-1}, k_j)$;
$(k_{j-1}, k_j) = m^T + Al_{tmp}^T + Btmp^T + CAl_{tmp}^T \circ CBtmp^T$;
$l_{tmp} \leftarrow (k_{j-1}, k_j)$;

algorithm is applied (If the length of the message is not divisible by 64 then an appropriate padding will be used).

Detailed encryption and decryption algorithm is given in Tables 5 and 6, respectively.

In the encryption algorithm, there are mainly 2 steps. The matrices Q_1, Q_2, ..., Q_8 and $T_1, T_2, ..., T_8$ are respectively used in the first and second step. In the first step 64 bit block gets slitted into 8 smaller blocks having size 8 bits, then an e-transformation is applied on each of these small blocks (called mini-blocks) with a different quasigroup and a different leader (the quasigroup Q_i and the leader l_i are used in the ith mini-block). The final string of first step is used as an input in the consecutive step. In the next step, on each resulting string we apply e-transformation, repeating this 8 times with alternatively changing direction. The leader l_i and the quasigroup T_i are used in the ith transformation.

In decryption algorithm the parastrophe $(Q, \backslash)$ of the quasigroup $(Q, *)$ has been used. If $x * y = z$, then the matrix representation of $y = x\backslash z$ is:

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

In decryption algorithm, the inverse operation mentioned above is used. Firstly the e-transformation is reversed using the quasigroups $T_8, T_7, ..., T_1$

Table 5. Encryption algorithm of BCDMPQ.

Encryption algorithm
Input: The round key $k = k_1k_2...k_{128}$, and the plaintext message $a = a_1a_2...a_{64}$, where k_i, a_i are bits. **Output**: The ciphertext message $c = c_1c_2...c_{64}$, c_i is bits. **Initialization**: Put $l_i = (k_{2i-1}, k_{2i})$ for $i = 1, 2, ..., 8$. Initialize the matrices A_{Q_i} and B_{Q_i} and the vector m_{Q_i} for $i = 1, 2, ..., 8$. Look at 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_{T_i} and B_{T_i} and the vector m_{T_i} for $i = 1, 2, ..., 8$. Look at the quasigroup T_i using the sequence number binary presented as $(k_{7(i+8)-6}, k_{7(i+8)-5}, ..., k_{7(i+8)})$.

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

Table 6. Decryption algorithm of BCDMPQ.

Decryption algorithm
Input: The round key $k = k_1k_2...k_{128}$, and the ciphertext message $c = c_1c_2...c_{64}$, where k_i, c_i are bits. **Output**: The plaintext message $a = a_1a_2...a_{64}$, a_i is bits. **Initialization**: Put $l_i = (k_{2i-1}, k_{2i})$ for $i = 1, 2, ..., 8$. Initialize the matrices A_{Q_i} and B_{Q_i} and the vector m_{Q_i} for $i = 1, 2, ..., 8$. Look at 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_{T_i} and B_{T_i} and the vector m_{T_i} for $i = 1, 2, ..., 8$. Look at the quasigroup T_i using the sequence number binary presented as $(k_{7(i+8)-6}, k_{7(i+8)-5}, ..., k_{7(i+8)})$.

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 do
$tmp \leftarrow (a_{j-1}, a_j)$;
$(a_{j-1}, a_j) = B_{T_{i+4}}^{-1} m_{T_{i+4}}^T + B_{T_{i+4}}^{-1}(I + C)A_{T_{i+4}} l_{tmp}^T + B_{T_{i+4}}^{-1}(Cm_{T_{i+4}}^T \circ CA_{T_{i+4}} l_{tmp}^T) + B_{T_{i+4}}^{-1} tmp^T + B_{T_{i+4}}^{-1}(CA_{T_{i+4}} l_{tmp}^T \circ Ctmp^T)$;
$l_{tmp} \leftarrow (a_{j-1}, a_j)$;
$l_{tmp} \leftarrow l_i$;
for $j = 1$ to 63 do
$tmp \leftarrow (a_j, a_{j+1})$;
$(a_{j-1}, a_j) = B_{T_i}^{-1} m_{T_i}^T + B_{T_i}^{-1}(I + C)A_{T_i} l_{tmp}^T + B_{T_i}^{-1}(Cm_{T_i}^T \circ CA_{T_i} l_{tmp}^T) + B_{T_i}^{-1} tmp^T + B_{T_i}^{-1}(CA_{T_i} l_{tmp}^T \circ Ctmp^T)$;
$l_{tmp} \leftarrow (a_j, a_{j+1})$;
for $i = 1$ to 8 do
$l_{temp} \leftarrow l_i$;
for $j = 1$ to 7 do
$tmp \leftarrow (a_j, a_{j+1})$;
$(a_{j-1}, a_j) = B_{Q_i}^{-1} m_{Q_i}^T + B_{Q_i}^{-1}(I + C)A_{Q_i} l_{tmp}^T + B_{Q_i}^{-1}(Cm_{Q_i}^T \circ CA_{Q_i} l_{tmp}^T) + B_{Q_i}^{-1} tmp^T + B_{Q_i}^{-1}(CA_{Q_i} l_{tmp}^T \circ Ctmp^T)$;
$l_{tmp} \leftarrow (a_j, a_{j+1})$;

sequentially (i.e. in the reverse order of quasigroups), then the *e*-transformations of the mini-blocks are reversed using the quasigroups $Q_8, Q_7, ..., Q_1$. Only preliminary security investigations were done in the cipher BCDMPQ.

In 1995, Carter et al. proposed a design of block cipher based on quasigroups (Carter, Dawson, and Nielsen 1995). In this design, quasigroup operation defined by a Latin square replaced the XOR operation in a version of DES, and also claimed that reduced number of rounds can safely be studied. There is a family of tweakable block cipher by the name Alex'smile-(B,I,G) designed by Mileva (2010) which is based on quasigroup and works on 32-bit words.

6.3. Pseudo random number generators

It is impossible to generate (in practice) a truly random sequence, so we work with that sequence which looks random. Any sequence that passes all the statistical tests for randomness is called a random sequence. As there are infinitely many such tests so it is not possible to examine all these tests for randomness. So in many applications, instead of random sequences we use random looking sequences. Such sequences are known as Pseudo

Random Sequences (PRS). The deterministic algorithms which generate such a sequence are known as Pseudo Random Sequence Generator (PRSG) and Pseudo Random Number Generator (PRNG) (when we have number sequences).

The number generated by PRNG is not truly random because it is totally determined by an initial value, known as PRNG's seed. Several PRNG can be designed by using quasigroup transformations. The cryptographically secure (since key is used) stream ciphers can be used as PRNG. Pseudo random sequences can be generated by quasigroup PRNG (QPRNG) (Markovski, Gligoroski, and Kocarev 2005) from very biased sequences as well as from periodical sequences. The choice of quasigroup is significant in QPRNG. Exponential and shapeless quasigroup should be chosen so that the period of growth is as high as possible. An algorithm for simple QPRNG is given in Table 7.

6.4. Stream cipher

Stream cipher is a symmetric-key encryption scheme. They can be considered as very simple block cipher with one block length. According to the generation of keystream, the stream ciphers are mainly classified as synchronous and asynchronous. A synchronous stream cipher is the one in which keystream is generated independent of the plaintext and ciphertext. A asynchronous stream cipher is the one in which the keystream is generated by the key and a fixed number of ciphertext symbols. Totally asynchronous stream cipher is the one in which the keystream is produced from the key and the previous ciphertext symbols. Stream ciphers are faster than block ciphers and also have lower hardware complexity. It may be advantageous to use a stream cipher over a block cipher in those situations where the transmission errors are highly probable as stream ciphers have

Table 7. Algorithm for simple QPRNG.

Algorithm for simple QPRNG
Phase I. Initialization
1. Choose a positive integer $s \geq 4$;
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 $b_0b_1b_2...b_j \in A$
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;

limited or no error propagation. Some of the stream ciphers known to us are PANAMA, SEAL, RC4, etc.

One of the earliest attempt to construct a quasigroup based stream cipher was made by Sarvate and Seberry (1986) in 1986 using a set of mutually orthogonal Latin squares (MOLS) $\{L_1, \ldots, L_k\}$ of order n. A pair of different squares (L_c, L_d) is the secret key. Because of the orthogonality, decryption can be done easily. Another attempt was made by Kościelny (1996) in 1996 in which he suggested to use quasigroup $(Q, \circ)$ which is isomorphic to the cyclic group of order q or abelian loop of even order q or additive group of $GF(q)$. For generating quasigroup, reader can find various Maple 7 routines in Kościelny (2002), that is isomorphic to the interior of additive group and multiplicative group of a finite field $GF(q^m)$ of order q and their isotopies. He has also used the two conjugates, namely $(Q, \backslash)$ and $(Q, /)$ of $(Q, \circ)$, for the generation of stream cipher. Let $k_1k_2k_3\ldots$, $m_1m_2m_3\ldots$ and $c_1c_2c_3\ldots$ denote respectively the keystream, the stream of characters of the plaintext and the stream of characters of the ciphertext. The author has proposed 6 ways of enciphering and deciphering as:

$$\begin{aligned}
c_i &= m_i \circ k_i, \quad m_i = c_i/k_i \\
c_i &= k_i \circ m_i, \quad m_i = k_i\backslash c_i \\
c_i &= k_i/m_i, \quad m_i = c_i\backslash k_i \\
c_i &= m_i/ki, \quad m_i = c_i \circ k_i \\
c_i &= m_i\backslash k_i, \quad m_i = k_i/c_i \\
c_i &= k_i\backslash m_i, \quad m_i = k_i \circ c_i.
\end{aligned}$$

If $(Q, \circ)$ is non-associative quasigroup, then m_i can be mapped into six different characters c_i, that is an improvement in the stream ciphers built over the field $GF(2)$ with XOR operation. Another attempt was made by Markovski, Gligoroski, and Andova (1997) to generate quasigroup based asynchronous stream cipher. Later the same encoding method is used by Ochadková and Snášel (2001). In Markovski, Gligoroski, and Stojčevska (2000), encryption is based on quasigroup in which $T^{(n)}_{l_n,\ldots,l_1}$ - transformation is used for encryption and opposite transformation for decryption.

Petrescu (2007) used ternary quasigroups for encryption, which is an asynchronous stream cipher. Let (Q, α) be a quasigroup that is used as a seed, that is publicly know, and is an isotope carrier. It forms an algebra $(Q, \alpha, \alpha_1, \alpha_2, \alpha_3)$ with four ternary operations fulfilling the following identities:

$$\begin{aligned}
\alpha(\alpha_1(x_1), x_2, x_3) &= x_1, \quad \alpha_1(\alpha(x_1), x_2, x_3) = x_1 \\
\alpha(x_1, \alpha_2(x_2), x_3) &= x_2, \quad \alpha_2(x_1, \alpha(x_2), x_3) = x_2 \\
\alpha(x_1, x_2, \alpha_3(x_3)) &= x_3, \quad \alpha_3(x_1, x_2, \alpha(x_3)) = x_3.
\end{aligned}$$

Table 8. Encryption function $E_k(m_1m_2\ldots) = c_1c_2\ldots$.

$i = 1$	$i = 2$	$i = 3$
$c_1 = \beta(m_1, a_1, a_2)$ $c_2 = \beta(m_2, a_3, a_4)$ $j > 2$ $c_j = \beta(m_j, c_{j-2}, c_{j-1})$	$c_1 = \beta(a_1, m_1, a_2)$ $c_2 = \beta(a_3, m_2, a_4)$ $j > 2$ $c_j = \beta(m_j, c_{j-2}, c_{j-1})$	$c_1 = \beta(a_1, a_2, m_1)$ $c_2 = \beta(a_3, a_4, m_2)$ $j > 2$ $c_j = \beta(m_j, c_{j-2}, c_{j-1})$

Table 9. Decryption function $D_k(c_1c_2\ldots) = m_1m_2\ldots$.

$i = 1$	$i = 2$	$i = 3$
$m_1 = \beta_1(c_1, a_1, a_2)$ $m_2 = \beta_1(c_2, a_3, a_4)$ $j > 2$ $m_j = \beta_1(c_j, c_{j-2}, c_{j-1})$	$m_1 = \beta_2(a_1, c_1, a_2)$ $m_2 = \beta_2(a_3, c_2, a_4)$ $j > 2$ $m_j = \beta_2(c_j, c_{j-2}, c_{j-1})$	$m_1 = \beta_3(a_1, a_2, c_1)$ $m_2 = \beta_3(a_3, a_4, c_2)$ $j > 2$ $m_j = \beta_3(c_j, c_{j-2}, c_{j-1})$

Table 10. Quasigroups used in Edon80.

$\bullet_0$	0	1	2	3
0	0	2	1	3
1	2	1	3	0
2	1	3	0	2
3	3	0	2	1

$\bullet_1$	0	1	2	3
0	1	3	0	2
1	0	1	2	3
2	2	0	3	1
3	3	2	1	0

$\bullet_2$	0	1	2	3
0	2	1	0	3
1	1	2	3	0
2	3	0	2	1
3	0	3	1	2

$\bullet_3$	0	1	2	3
0	3	2	1	0
1	1	0	3	2
2	0	3	2	1
3	2	1	0	3

Let $\mathcal{K} = Q^4 \times \{1, 2, 3\}$ be the key space. Let $k = a_1a_2a_3a_4i$ represent the key, then another isotopic quasigroup (Q, β) is determined as:

$$\beta(x_1, x_2, x_3) = f_4(\alpha(f_1^{-1}(x_1), f_2^{-1}(x_2), f_3^{-1}(x_3))),$$

where $f_j = f_{a_j}$ are permutations on Q. Corresponding to each key k the encryption function $E_k(m_1m_2\ldots) = c_1c_2\ldots$ is given in Table 8 and the decryption function $D_k(c_1c_2\ldots) = m_1m_2\ldots$ is given in Table 9.

The author also proposed an implementation in which the seed quasigroup (Q, α) is defined as $\alpha(x_1, x_2, x_3) = (x_1 - x_2 + x_3) \mod 256$ and $f_a(x) = x + a \mod 256$.

Gligoroski et al. (2005a, 2008c) designed Edon80 in the year 2008 which is one of the eSTREAM finalists. This is a stream cipher based on quasigroup. In this cryptographic primitive, the most interesting thing is that it uses only 4 quasigroups and still is resistant to all types of attacks. In this design it has been claimed that out of 576 quasigroups of order 4, 64 are acceptable to be used in Edon80. The quasigroups with lexicographic order 61, 241, 350 and 564 (shown in Table 10) were chosen by the authors. Some of its algebraic properties are examined by Vojvoda, Sýs, and Jókay (2007).

Edon80 is a binary additive stream cipher with three modes of operation namely KeySetup, IVSetup and Keystream mode. Hell and Johansson (2008) suggested a related key attack on Edon80 with complexity of 2^{69} although the Edon80 authors disputed this complexity.

Table 11. Initialization of EdonX,Y,Z.

Initialization of EdonX,Y,Z
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 $(Q, *)$ 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 512s-bits words.
Phase 4. Transformation of K_{ex} with the given quasigroup $(Q, *)$ of order 2^s

4. For $i = 0$ to 511 do
begin
Set leader:= $K[i \bmod (n+2)]$;
$K_{ex} \leftarrow e_{leader,*}(K_{ex})$;
$K_{ex} \leftarrow RotateLeft(K_{ex})$;
end;

Phase 5. Transformation $(Q, \bullet) \leftarrow$ Isotope $(Q, *)$

5. $(Q, \bullet) \leftarrow (Q, *)$;
For $i = 0$ to 505 step 8 do
begin
Set $row_1 := K_{ex}[i]$;
Set $row_2 = K_{ex}[i+1]$;
$(Q, \bullet) \leftarrow SwapRows(Q, row_1, row_2)$;
Set $column_1 := K_{ex}[i+2]$;
Set $column_2 = K_{ex}[i+3]$;
$(Q, \bullet) \leftarrow SwapColumns(Q, column_1, column_2)$;
Set $\gamma := (K_{ex}[i+4], K_{ex}[i+6])$;
$(Q, \bullet) \leftarrow \gamma(Q, \bullet)$;
end;

Phase 6. Setting the working key $K = K_0 \| \ldots \| K_{m-1}$ (the last m s-bits words of K_{ex})
6. Set $K = K_0 \| K_1 \| \ldots \| K_{m-1} := K_{ex}[512 - m] \| \ldots \| K_{ex}[511]$.

Another eSTREAM that is designed by Matsumoto et al. (2007, 2008) is CryptMT v3 (Cryptographic Mersenne Twister). It is an unbroken phase 3 candidate that uses quasigroups.

6.4.1. EdonX, Y, Z

In this subsection, we will briefly discuss 3 different types of stream ciphers: EdonX which is a synchronous stream cipher, EdonY which is an asynchronous stream cipher and EdonZ which is a totally asynchronous stream cipher. For more details the reader may reference Lj, Markovski, and Gligoroski (2014).

The initialization phase is same for all the three ciphers EdonX,Y,Z and it is a major phase in their designs. Let K_{in} denotes the initial key shared secretly and it gets converted into the working key K. Both the keys and the messages consist of words having s-bit of any desired length $s \geq 4$. The quasigroup is defined on the set of s-bit words and has 2^s elements. One of the important feature of this Edon family is the flexibility of the choice of the key length. The algorithm for the initialization of EdonX,Y,Z family is described in Table 11.

Table 12. Quasigroup $(Q, *)$ and its isotopes $(Q, \bullet)$.

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

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

In the initialization algorithm K_{ex} is expanded key which is 512 s-bit word and $K_{in}[j], K_{ex}[j]$ and $K[j]$, respectively denotes the j^{th} s-bit words in K_{in}, K_{ex} and K, i.e. both $K[j]$ and K_j are same. The symbol $||$ denote the concatenation of s-bit words. $RotateLeft(K_{ex})$ cyclically rotates the values of the expanded key K_{ex} i.e. $K_{ex}[j] \leftarrow K_{ex}[j+1]; j = 0, 1, 2, ...510$ and $K_{511} \leftarrow K_{ex}[0]$. *SwapRows* and *SwapColumns* are the functions that swap the rows and columns of the quasigroup (i.e. in the Latin square) and γ is the permutation of length 2 (i.e. 2 cycle). In the phase 6 the working key is obtained by concatenation of last m s-bits words of K_{ex}.

Now we will give an example that works on the principles of the initialization algorithm to obtain a secret working key K and quasigroup $(Q, \bullet)$. For simplicity of the explanation, we will work on quasigroup of order 4 (i.e. Q= {0,1,2,3}) and take $m = n$. Accordingly, instead of s-bit, we work with 2-bit letters (i.e. 0, 1, 2 and 3). Moreover, in place of extended key of length 512 s-bit words we use 32 2-bit words. Let the initial quasigroup $(Q, *)$ be given in Table 12.

Phase 1. Let the initial secret key K_{in} = 0 2 1 3 (hence $n = 4$) and $s = 2$ is the length of the words.
Phase 2. $n = 4$ has the representation with 2-bit words as: 0 1 0 0 = 0 1 || 0 0. Hence n_1=1, n_2=0, K = 0 2 1 3 1 0.
Phase 3. K_{ex} = 0 2 1 3 1 0 0 2 1 3 1 0 0 2 1 3 1 0 0 2 1 3 1 0 0 2 1 3 1 0 0 2.
Phase 4. After transformation of we get: K_{ex}= 2 0 2 0 1 0 0 3 0 3 3 3 1 1 3 2 2 2 3 2 1 1 0 1 2 0 2 0 1 0 1 1.
Phase 5. After transformation we get secret working Quasigroup $(Q, \bullet)$ given in Table 12.
Phase 6. The secret working key is the last 4 letters of K_{ex} and hence K = 1 0 1 1.

6.4.1.1. EdonX. EdonX takes 4-bit(nibbles) variables as input, and hence for quasigroup transformation on the streams of data the quasigroup $Q = \{0, 1, ..., 15\}$ of order 2^4 is used. The working key K obtained through

Table 13. Encryption and decryption algorithm of EdonX.

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

initialization is stored in the internal variables K_i i.e. $K = K_0K_1...K_{m-1}$, where $m \geq 64$ and $K_i \in Q$. The encryption and decryption algorithm of EdonX are the same and are described in Table 13. In this algorithm two auxiliary variables X and T of 4-bit and an integer variable *count* and for each value of *count* we get an output which is the bitwise XOR of X and Inputnibble.

Now we will give an example that works on the principles of the EdonX encryption/description algorithm. We will work on the same example as taken in the initialization algorithm. After phase 1 we obtain $K = 1\ 0\ 1\ 1$ and $(Q, \bullet)$ as in Table 12. If the Inputnibble i.e. $M = 3\ 1\ 2\ 0\ 1\ 1\ 2\ 3\ 0\ 1$ ..., then the algorithm for EdonX give the output ciphertext string as $C = 0\ 1\ 3\ 1\ 2\ 3\ 3\ 1\ 2\ 2$ ....

The calculations for the decrypting phase are same as that of encryption, the only difference would be in the last two rows (i.e. input and output would be C and $M = X \oplus C$ respectively) because it is a binary additive stream cipher.

Some steps of EdonX encryption/decryption are given in Table 14.

It has been proved that EdonX is resistant to chosen plaintext/ciphertext attacks.

Edon X can also be used for generating pseudo-random numbers. For this we take the message (Inputnibble) $M = 0\ 0\ 0\ 0 \ldots$ consisting of zeros only then the output string would come out to be $C_i = X_i \oplus 0$, for $i = 0,1,2, \ldots$, then in this case the output string is nothing but the variable X obtained by the above described algorithm.

We obtain the following system of iterative functions from the above encryption/decryption algorithm:

Table 14. Some steps of encryption/decryption of EdonX.

	K	X	T	K	X	T	K	X	T	K	X	T	K	X	T	K	...
i		1	1		0	1		2	0		1	3		0	0		...
0	1	3	3	3	0	0	0	2	0	2	0	3	0	3	2	3	
1	0	0	3	0	3	2	3	3	2	3	0	3	0	0	2	0	
2	1	1	2	1	2	3	2	3	1	3	0	3	0	3	1	3	
3	1	3	1	1	0	3	3	1	1	1	1	2	2	3	3	3	
Output		0			1			3			1			2			

Table 15. Encryption and decryption algorithm of EdonY.

Encryption Algorithm	**Decryption Algorithm**
Input: Key K of length n and message M.	**Input**: Key K of length n and message C.
Output: Message C.	**Output**: Message M.
1. $count \leftarrow 0$; $p = \lfloor \frac{n}{2} \rfloor$;	1. $count \leftarrow 0$; $p = \lfloor \frac{n}{2} \rfloor$;
2. $K_0 \leftarrow K_0 \bullet (M_{count} \bullet K_{count+p \bmod n})$;	2. $X \leftarrow K_{n-1}$; $K_{n-1} \leftarrow C_{count}$;
3. For $i = 0$ to $n-1$ do begin $K_i \leftarrow K_i \bullet K_{i-1}$; end;	3. For $i = n-2$ down to 0 do begin $Y \leftarrow K_i$; $K_i \leftarrow X \backslash K_{i+1}$; $X \leftarrow Y$; end;
4. Output: $C_{count} = K_{n-1}$;	4. Output: $M_{count} = (X \backslash K_0)/K_{count+p \bmod n}$;
5. $count \leftarrow count + 1$;	5. $count \leftarrow count + 1$;
6. Go to 2;	6. Go to 2;

$$\begin{cases} K_{\lambda,0} = K_{\lambda-1,0} \bullet K_{\lambda-1,count \mod m} \\ K_{\lambda,1} = K_{\lambda-1,1} \bullet K_{\lambda,0} \\ K_{\lambda,2} = K_{\lambda-1,2} \bullet K_{\lambda,1} \\ \cdots\cdots\cdots \\ K_{\lambda,m-2} = K_{\lambda-1,m-2} \bullet K_{\lambda,m-3} \\ X_{\lambda,m-1} = K_{\lambda,m-1} = K_{\lambda-1,m-1} \bullet K_{\lambda,m-2} \\ K_{\lambda,m-1} = ((\ldots(K_{\lambda-1,count+p \mod m} * K_{\lambda,0}) * K_{\lambda,1}) \ldots * K_{\lambda,m-2}) * X_{\lambda,m-1}. \end{cases} \tag{18}$$

For the output string we just require the value of $X_{\lambda,m-1}$, for $\lambda = 0, 1, 2, \ldots$. Hence, the string $X_{0,m-1}, X_{1,m-1}, X_{2,m-1}, X_{3,m-1}, \ldots$ is C which can easily be obtained. The period and the nature of the string depends on the theory of discrete chaos systems(not yet developed).

6.4.1.2. EdonY. The initialization phase of EdonY is the same as that of EdonX. The next phase, encryption and decryption algorithm of EdonY are defined in Table 15, where $M = M_0M_1M_2\ldots$ and $C = C_0C_1C_2\ldots$ are the input plaintext and ciphertext strings. X and Y are 5-bits auxiliary variables

in the decryption algorithm. A public quasigroup $(Q, *)$ of order 32 defined on 5-bits letters are used (i.e. $Q = \{0, 1, 2, ..., 31\}$) in the construction of EdonY.

Theorem 6.1. *Let* $E = e_{l_1,*_1} \circ ... \circ e_{l_n,*_n}$ *and* $D = d_{l_n,\backslash_n} \circ ... d_{l_1,\backslash_1}$ *be transformations obtained with n quasigroup transformations* $*_1, ..., *_n$ *on Q, leaders* $l_1, ..., l_n$ *and corresponding parastrophes* $\backslash_1, ..., \backslash_n$. *Assume that* $E(b_1 b_2 ... b_k) = c_1 c_2 ... c_k$, $k > n$, *and* $d \neq c_i$ *for some fixed* i $(b_j, c_j, d \in Q)$. *Then for some* $d_1, ..., d_{n+1} \in Q$,

$$D(c_1 ... c_{i-1} d c_{i+1} ... c_k) = \begin{cases} b_1 b_2 ... b_{i-1} d_1 ... d_{n+1} b_{i+n+1} ... b_k, & k > i + n \\ b_1 b_2 ... b_{i-1} d_1 ... d_{k-i+1}, & k \leq i + n \end{cases}.$$

It can be seen from the above theorem that one error in the ciphertext C will lead to the $n+1$ error in the recovered plaintext M' i.e. there are $n+1$ consecutive letters in which the message M and M' differ. Hence EdonY is self-synchronized. The recovered plaintext will have an error of $r+n$ length, if there is a string of errors of length r in C.

Now we will give an example that works on the principles of the algorithm of EdonY by working on the same example as in the initialization algorithm. After initialization we obtain new working key $K = 1\ 0\ 1\ 1$ and new quasigroup $(Q, \bullet) \leftarrow$ Isotope $(Q, *)$. Now if we take plaintext $M = 2\ 1\ 3\ 0\ 2\ 1\ 1\ 0\ 3\ 0$ ..., then the encryption algorithm for EdonY gives the output ciphertext string as $C = 2\ 0\ 1\ 0\ 1\ 2\ 2\ 3\ 1\ 2$ ....

EdonY is resistant to chosen plaintext/ciphertext and dictionary attacks. The resistance to the known ciphertext attack follows from the next theorem.

Theorem 6.2. *Given a ciphertext C, for each quasigroup operation* $*$ *on* $Q = \{0, 1, ..., 31\}$ *and each key* $K = K_0 K_1 ... K_{n-1}$, *there is a plaintext M such that C is its ciphertext.*

6.4.1.3. EdonZ. EdonZ operates on nibbles i.e. takes 4-bit variables and hence for quasigroup transformation on the streams of data it uses quasigroups $(Q, *)$, $Q = \{0, 1, ..., 15\}$ of order 16. *T*, *X* and *temp* are used as a temporary 4-bit variables in the encryption and decryption algorithm of EdonZ described in the Table 16. EdonZ needs the left parastrophe of the quasigroup $(Q, \bullet)$ in its decryption algorithm where as EdonX being a binary additive stream cipher, does not use $(Q, \backslash)$.

EdonZ is used in defining of the Random codes based on quasigroups (RCBQ) with cryptographic properties (Gligoroski, Markovski, and Kocarev 2007; Popovska-Mitrovikj, Bakeva, and Markovski 2011). RCBQ are the

Table 16. Encryption and decryption algorithm of EdonZ.

Encryption algorithm	Decryption algorithm
Input: Key K of length n and message M. **Output**: Message C.	**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 $X \leftarrow K_i \bullet X$; $T \leftarrow T \oplus X$; $K_i \leftarrow X$; 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 $i = n-1$ down to 0 do $X \leftarrow temp \backslash X$; $T \leftarrow T \oplus X$; $temp \leftarrow K_{i-1}$; $K_{i-1} \leftarrow X$; 4. $K_{n-1} \leftarrow T$; 5. Output: X; 6. Go to 1;

Table 17. Quasigroup $(Q, \bullet)$ and its parastrophe $(Q, \backslash)$.

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

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

Table 18. Some steps of EdonZ encryption are given as below.

		M_0			M_1			M_2			M_3			
	K	X	T	K	X	T	K	X	T	K	X	T	K	...
i		0	0		0	0		2	0		1	0		...
0	1	1	1	1	1	1	1	3	3	3	3	3	3	
1	3	3	2	3	3	2	3	2	1	2	1	2	1	
2	2	1	3	1	0	2	0	0	3	0	1	3	1	
3	2	0	3	3	0	2	2	3	0	0	1	2	2	
Output		0			0			3			1			

error-correcting codes, that provide information security. These codes have been proposed by Gligoroski, Markovski, and Kocarev (2007).

Now we give an example that works on the principles of EdonZ. For simplicity, we will work on quasigroup of order 4. Let the length of the working key be 4. Let the working key be $K = 1\,3\,2\,2$ and the message $M = 0\ 0\ 2\ 1\ 0\ \ldots$. The quasigroup and its parastrophe are shown in Table 17. After encryption the output obtained is $C = 0\,0\,3\,1\ \ldots$

Some steps are given in Table 18 of EdonZ encryption.

6.5. Hash functions

A hash function is a mapping h in which at least the following two properties hold:

1. *Compression*: The mapping $h : \{0,1\}^* \rightarrow \{0,1\}^n$ i.e. h maps an input of any arbitrary finite bit length to a fixed size output.
2. *Easy to compute*: It is easy to compute $h(x)$, for any given h and an input x.

As defined here, hash function implies an unkeyed hash function. In cryptography and information security, hash functions are considered as the "Swiss army knife" because of their versatile application in countless protocols and algorithms. They are used in key derivation, checking data integrity, password based identification systems, digital signature schemes, commitment schemes, digital time stamping schemes, pseudo random functions, and many other security applications. Hash functions can be divided in cryptographic hash functions known as manipulation detection codes(MDCs) and keyed hash functions known as message authentication codes (MACs). An additional input of secret key having fixed length is used in MACs.

The beginning attempts (Dvorsky, Ochodkova, and Snašel 2001, 2002; Gligoroski and Knapskog 2008; Gligoroski, Markovski, and Bakeva 2003; Markovski, Gligoroski, and Bakeva 2001) for generating cryptographic hash functions using quasigroup transformations hardly have any implementations. Snášel et al. (2009) defined hash function on a message as one elementary e-transformation. Gligoroski, Markovski, and Kocarev (2006) described a generic hash function with reverse string transformation $\mathcal{R}$. Edon-F is another generic quasigroup based hash function, discussed in Markovski, Gligoroski, and Bakeva (2003) without implementation. As one more application of quasigroups, Gligoroski et al. (2005b) gave quasigroup folding which is a comparatively cheap and simple fix for all MD4 family hash functions. This new construction of hash functions has enhanced the security features, but is approximately 2 times slower.

Further we will discuss the candidates of National Institute of Standards and Technology(NIST) SHA-3 competition whose designs are based on the huge quasigroup transformations, namely Edon-$\mathcal{R}$ and NaSHA.

6.5.1. Edon-$\mathcal{R}$

Edon-$\mathcal{R}$ designed by Gligorovski et al. (2008) is a wide-pipe iterative hash function with Merkle-Damgård straightening. It was the fastest candidate of NIST SHA-3 competition.

The compress function of Edon-$\mathcal{R}$ consists of two strings, one having length $2N$ and another of length N, where N is the size of hash digest in w-bit words. The algorithm for Edon-$\mathcal{R}$ is given in the Table 19. For more detailed description the reader may reference Gligorovski et al. (2008).

Table 19. Algorithm for Edon-$\mathcal{R}$.

Algorithm for Edon-$\mathcal{R}$
Input: Quasigroup $(Q, *)$, N and M, where the quasigroup $(Q, *)$ is shapeless having order 2^w, $w \geq 4$, N is the hash digest size in w-bit words and M is the message to be hashed.
Output: A hash of length $w \times N$ bits.
Step 1. Padding: The message M is padded to M' so that the length of M' would become a multiple of N w-bit words.
Step 2. Initialization: $H_0 = (0 \ mod \ 2^w, 1 \ mod \ 2^w, \ldots, 2N-1 \ mod \ 2^w)$ is initial value.
Step 3. Working: The following iterative procedure is used to compute the hash value: for $i = 1$ to k do $H_i = \mathcal{R}(H_{i-1} \| M_i) \ mod \ 2^{2wN}$;
Output: Edon-$\mathcal{R}(M)$=$H_k \ mod \ 2^{wN}$

Two huge quasigroups of order 2^{256}, 2^{512} are used in the compression function $\mathcal{R}$ and for their algorithmic description reader may refer chapter 2 of Gligorovski et al. (2008).

Khovratovich, Nikolić, and Weinmann (2009) proposed various attacks on Edon-$\mathcal{R}$. With minor effort all 3 main attacks (preimage, second pre-image and collision) can be applied on Edon-$\mathcal{R}$, and this is done in free start attack scheme with variable initial chaining values. In these attacks the asymmetrical diffusion of the chaining values in the compression function is exploited. By partially inverting the compression function and fixing one part of the chaining value, a meet-in-the-middle attack on Edon-$\mathcal{R}$-n to find real pre-images was launched.

6.5.2. NaSHA

NaSHA is also a wide-pipe iterative hash function with standard MD-strait-ening. It was designed by Markovski and Mileva (2008) in the year 2008 and was also a first round candidate of NIST SHA-3 competition. For defining the cryptographic hash function NaSHA-(m, k, r), the quasigroup transformation $\mathcal{MT}$ is used. The parameters m denotes the length of the message digest, k is the number of elementary quasigroup string transformations that is of the type $\mathcal{A}$ *and* $\mathcal{RA}$ (i.e. complexity of $\mathcal{MT}$) and r is from the order 2^{2^r} of quasigroup used. Hence m and r are positive integers and k is a positive even integer. Table 20 briefly describes the algorithm of NaSHA-(m, k, r).

The implementation of NaSHA-$(m, 2, 6)$ hash family and Design rationales have been discussed in the papers (Markovski and Mileva 2008, 2009b) for $m \in \{224, 256, 384, 512\}$. This hash family uses Merkle-Damgård domain extender with standard Merkle-Damgård strengthening. Their design also incorporates the wide-pipe design of Lucks (2004, 2005) and Coron et al.'s (2005) ideas. In the algorithm, the compression function $\mathcal{MT}$ at each iterative step uses the massage blocks and chaining variable, each of $2n$-bit length, hence the string of $4n$ bits length are mapped to the string of $4n$ bits length and out of which, for the next iterative step, only

Table 20. Algorithm of NaSHA-(m, k, r).

Algorithm for NaSHA-$(\mathbf{m}, \mathbf{k}, \mathbf{r})$
Input: A positive even integer k and positive integers m and r such that $m > 2^r$, and a message M. **Output:** A hash value (message digest) NaSHA-$(m, k, r)(M)$ of m bits.
Step 1. Denote by n the smallest integer such that $m \leq 2^n$. (For example, $n = 8$ for $m = 224$ and $n = 9$ for $m = 384$.) **Step 2.** Pad the message M, so that the length of the padded message M' is a multiple of 2^{n+1}, $\|M'\| = 2^{n+1}N$ for some integer N. Separate M' in N 2^{n+1}-bit blocks, $M' = M_1 \| M_2 \| \ldots \| M_N$, $\|M_i\| = 2^{n+1}$. **Step 3.** Initialize the initial value H_0, which is a 2^{n+1}-bit word. **Step 4.** The first message block M_1 and the initial value H_0 separate to $q = 2^{n-r+1}$ 2^r-bits words: $M_1 = S_1 \| S_3 \| S_5 \| \ldots \| S_{2q-3} \| S_{2q-1}$, $H_0 = S_2 \| S_4 \| S_6 \| \ldots \| S_{2q-2} \| S_{2q}$, $(\|S_i\| = 2^r)$ and form the word $S^{(0)} = S_1 \| S_2 \| S_3 \| S_4 \| \ldots \| S_{2q-3} \| S_{2q-2} \| S_{2q-1} \| S_{2q}$. **Step 5.** Choose leaders l_i as functions that depend on $S_1, S_2, S_3, \ldots, S_{2q}$, and a suitable linear transformation $LinTr_{2^{n+2}}$. **Step 6.** Choose two quasigroups $(\{0,1\}^{2^r}, *_1)$ and $(\{0,1\}^{2^r}, *_2)$ (one for $\mathcal{A}$ and one for $\mathcal{RA}$ transformation) and compute the string of bits $S^{(N-1)}$ as follows: for $i = 1$ to $N - 1$ do $A_1 \| A_2 \| A_3 \| \ldots \| A_{2q} \leftarrow \mathcal{MT}(LinTr^{2q}_{2^{n+2}}(S^{(i-1)}))$; $B_1 \| B_2 \| B_3 \| \ldots \| B_{q-1} \| B_q \leftarrow M_{i+1}$; $S^{(i)} := B_1 \| A_2 \| B_2 \| A_4 \| \ldots \| B_{q-1} \| A_{2q-2} \| B_q \| A_{2q}$; end. **Step 7.** Choose two quasigroups $(\{0,1\}^{2^r}, *_1)$ and $(\{0,1\}^{2^r}, *_2)$ and compute $\mathcal{MT}(LinTr^{2q}_{2^{n+2}}(S^{(N-1)})) := A_1 \| A_2 \| A_3 \| \ldots \| A_{2q}$. Then NaSHA-$(m, k, r)(M) = A_4 \| A_8 \| \ldots \| A_{2q-4} \| A_{2q} \ (mod\ 2^m)$.

$2n$ bits are kept. It is important to note that the final digest value of the compression function is at most half of the chaining variable. Due to the above design rationales Gligoroski and Knapskog (2008) concluded that NaSHA-(m, k, r) hash family are resistant to generic attack including Joux's multicollision attack (Joux 2004), length extension attack, Dean's fixed point attack (Dean 1999), Kelsey and Kohno's herding attack (Kelsey and Kohno 2006), Kelsey and Schneier's long message 2^{nd} preimage attack (Kelsey and Schneier 2005) and 2^{nd} collision attack.

The free-start collision of NaSHA family was proposed by Ji et al. (2008). They also proposed differential collision attack on NaSHA-512 with complexity equal to 2^{192} which was confirmed with unknown probability by Markovski et al. (2009). Free-start collision attacks on NaSHA with complexity of 2^{32} and free-start preimage attack on NaSHA-n with complexity of $2^{n/2}$ are given by Nikolić and Knovratovich (2008).

6.6. Public-key algorithm

Let $\{E_e : e \in \mathcal{K}\}$ and $\{D_d : d \in \mathcal{K}\}$ denote respectively the sets of encryption and decryption transformations in an encryption scheme. If for each pair of keys (e, d) of encryption/decryption, the key e is publicly known (called public key), whereas the other key d is kept secret (called private key) then the scheme is called a *public-key encryption scheme*. For the secure scheme, the estimation of d from e is computationally infeasible. A

public-key encryption scheme consists of the following three algorithms, namely (i) Key generation algorithm, (ii) Encryption algorithm and (iii) Decryption algorithm. In this algorithm, the encryption of the message is done by using public key. It is slower than the symmetric key algorithm, hence usually they are used for key management and key agreement protocols in two communicating parties, whereas actually the communication is done by using some symmetric key algorithm.

Gligoroski, Markovski, and Knapskog (2008a, 2008b) proposed a trapdoor one-way function with the help of multivariate quadratic quasigroups (MQQs) for building public key cryptosystems. Subsequently, a signature MQQ-SIG (Gligorovski et al. 2011) and an encryption scheme MQQ-ENC was proposed based on MQQs (Gligoroski and Samardjiska 2012). These cryptographic primitives belong to the multivariate public-key cryptosystems (MQ) and their security is based on the hardness of solving the system of quadratic polynomial equations over finite fields, which is known to be $\mathcal{NP}$-hard problem. The encryption speed of the above cryptographic primitives is similar to other MQ schemes, whereas the speed of the decryption is 500–1000 times faster than the public-key schemes. By proposing a modified version of the MutanXL algorithm, Mohamed, Ding, and Buchmann (2008) broke this public key cryptosystem. A recovery attack using good keys is given in Faugére et al. (2015) against MQQ-ENC and MQQ-SIG.

7. Conclusions and future work

In this article, firstly we have given a brief description on quasigroups & their transformations and discussed various ways of constructing huge quasigroups. Next, we described the representation of a finite quasigroup as a vector valued Boolean function. Finally, the use of these quasigroups and their string transformations to construct various cryptographic primitives were presented in this article. The cryptographic primitives covered in the article include S-boxes, block ciphers, stream ciphers, pseudo random number generators, hash functions and public-key algorithms.

A lot of research is being done to design cryptographic primitives using quasigroup. Apart from the primitives discussed in the paper, several other primitives have also been proposed in the literature (Bakhtiari, Safavi-Naini, and Pieprzyk 1997; Dénes and Keedwell 1992; Shcherbacov 2007) such as construction of MAC, quasigroup-based secret sharing schemes, zero knowledge protocols, generating the NLPN-sequences, Identity based cryptosystem, etc.

From our discussion we can conclude that the research community is gradually realizing the power of applications of quasigroups in constructing

cryptographic primitives. In fact, quasigroups could be a good instrumental tool to cater cryptographic requirements as quasigroups based cryptographic primitives are relatively easy to implement and require less silicon area. Construction of efficient and secure cryptographic primitives based on quasigroup and quasigroup transformation is still an open research problem and will be taken as a future task.

About the authors

Dimpy Chauhan is a research scholar at the Department of Mathematics, University of Delhi, India. She has received her Masters in Mathematics from Panjab University, India. Her research interests are in algebra and its applications, especially cryptography and coding theory.

Indivar Gupta completed his Ph.D. from the IIT Delhi, India. He has been working as a scientist in Scientific Analysis Group, DRDO since 2000, and has research contributions in various areas related to cryptography and information security. Presently, his areas of research include computational algebra, number theory, cryptography, information security, and high-performance computing.

Rashmi Verma is an assistant professor at the Department of Mathematics, Mata Sundri College for Women, University of Delhi, India. She completed her M.Phil. and Ph.D. from the Department of Mathematics, University of Delhi, India. Her research interest is in algebraic coding theory.

Funding

This study was supported by the University Grants Commission (Ref. No: 20/12/2015(ii)EU-V).

References

Bakhtiari, S., R. Safavi-Naini, and J. Pieprzyk. 1997. A message authentication code based on Latin squares. Australian Conference on Information Security and Privacy (ACISP '97), Springer-Verlag, LNCS 1270, 194–203.

Belousov, V. D. 1967. *Foundations of the theory of quasigroups and loops*. Moscow: Nauka (in Russian).

Belousov, V. D. 1972. *n-Ary quasigroups*. Kishinev: Shtiinta (in Russian).

Belousov, V. D. 1981. *Elements of quasigroup theory: A special course*. Kishinev: Kishinev State University Printing House (in Russian).

Belousov, V. D., and G. B. Belyavskaya. 1989. *Latin squares, quasigroups and their applications*. Kishinev: Shtiinta (in Russian).

Beutelspacher, A. 2002. *Cryptology: An introduction to the science of encoding, concealing and hiding*. Wiesbaden: Vieweg (in German).

Bogdanov, A., L. R. Knudsen, G. Le, C. Paar, A. Poschmann, R. M.J.B, Y. Seurin, and C. Vikkelsoe. 2007. PRESENT: An ultra-lightweight block cipher. Proceedings of CHES. Springer-Verlag.

Bruck, R. H. 1971. *A survey of binary systems*. New York: Springer Verlag. 3rd printing, corrected edition.

Carter, G., E. Dawson, and L. Nielsen. 1995. Desv: A Latin square variation of des. Proceedings of the Workshop on Selected Areas in Cryptography, Ottawa, Canada, 144–58.

Chabaud, F., and S. Vaudenay. 1995. Links between differential and linear cryptanalysis. Advances in Cryptology - EUROCRYPT'94, Workshop on the Theory and Application of Cryptographic Techniques, vol. 950, 356–65. Springer-Verlag.

Chein, O., H. O. Pflugfelder, and J. D. H. Smith. 1990. *Quasigroups and loops: Theory and applications*. Berlin: Heldermann Verlag.

Coron, J. S., Y. Dodis, C. Malinaud, and P. Puniya. 2005. Merkle-damgård revisited: How to construct a hash function. Advances in Cryptology - CRYPTO 2005, LNCS 3621, 430–48.

Courtois, N., A. Klimov, J. Patarin, and A. Shamir. 2000. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. LNCS 1807.

Dean, R. D. 1999. Formal aspects of mobile code security. PhD thesis, Princeton University.

Dénes, J. 1979. Latin squares and non-binary encoding. Proceedings of the Conference on Information Theory, CNRS, Paris, 215–21.

Dénes, J. 2000. On Latin squares and a digital encrypting communication system. *Pure Mathematics and Applications* 11 (4):559–63.

Dénes, J., and A. D. Keedwell. 1974. *Latin squares and their applications*. Cambridge, MA: Academic Press.

Dénes, J., and A. D. Keedwell. 1991. *Latin squares: New developments in the theory and applications*. Amsterdam: Elsevier.

Dénes, J., and A. D. Keedwell. 1992. A new authentication scheme based on Latin squares. *Discrete Mathematics* 106–107:157–61. A collection of contributions in honour of Jack van Lint. doi: 10.1016/0012-365X(92)90543-O.

Dénes, J., and A. D. Keedwell. 2001. Some applications of non-associative algebraic systems in cryptology. *Pure Mathematics and Applications* 12 (2):147–95.

Dimitrova, V. 2010. Quasigroup processing string, their Boolean representations and application in cryptography and coding theory. PhD thesis, Ss. Cyril and Methodius University, Skopje, S. Markovski (promotor).

Dimitrova, V., and S. Markovski. 2004. On Quasigroup pseudo random sequence generators. Proceedings of the 1st Balkan Conference in Informatics, Thessaloniki, Greece, 235–43.

Dimitrova, V., and S. Markovski. 2007. Classification of quasigroups by image patterns. Proceedings of the Fifth International Conference for Informatics and Information Technology, Bitola, Macedonia, 152–60.

Dvorsky, J., E. Ochodkova, and V. Snašel. 2001. Hash function based on quasigroups (Czech). Proceedings of Mikuláška kryptobesídká, Praha, 27–36.

Dvorsky, J., E. Ochodkova, and V. Snašel. 2002. Hash function based on large quasigroups (Czech). Proceedings of Velikonocni kryptologie, Brno, 1–8.

Euler, L. 1782. Recherches sur une nouvelle espéce de quarrés magiques. *Verh. Zeeuwsch. Genootsch. Wetensch. Vlissengen* 9:85–239. see also Euler, L., Opera Omnia., Series I: Opera mathematica. Vol. VII: algebraicae ad theoriam combinationum et probabilitatum pertinentes (Latin), Teubner, Leipzig and Berlin, 1923, pp. 291–392. JFM 49:0007.

Euler, L. 1849. Recherches sur une esp'ece de carrés magiques. In *Commentationes Arithmeticae Collectae*. Vol. II, 302–61. St. Petersburg: St. Petersburg Academy of Sciences. See also reprint made by Kraus Reprint, Nendeln/Liechtenstein, 1969, Vol. I, 1xxxix + 584 pp.; Vol. II: ix + 651 pp. MR0250831 (40:4063).

Faugére, J.-C., D. Gligoroski, L. Perret, S. Samardjiska, and E. Thomae. 2015. A polynomial-time key-recovery attack on MQQ cryptosystems. *Public-key cryptography–PKC 2015*, 150–74. Lecture Notes in Computer Science 9020. Heidelberg: Springer.

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

Gligorovski, D., R. S. Ødegård, M. Mihova, S. J. Knapskog, K. Lj, A. Drapal, and V. Klima. 2008. Cryptographic hash function EDON-R. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/documents/EdonRUpdate.zip.

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

Gligoroski, D., and S. J. Knapskog. 2008. Edon-R(256,384,512) – An efficient implementation of Edon-R family of cryptographic hash functions. *Commentationes Mathematicae Universitatis Carolinae* 49 (2):219–39.

Gligoroski, D., and S. Samardjiska. 2012. The multivariate probabilistic encryption scheme MQQ-ENC. IACR Cryptology ePrint Archive, Report 2012/328.

Gligoroski, D., S. Markovski, and L. Kocarev. 2006. Edon-R, an infinite family of cryptographic hash functions. *International Journal of Network Security* 8:293–300.

Gligoroski, D., S. Markovski, and L. Kocarev. 2007. Error-correcting codes based on quasigroups. Proceedings of 16th International Conference on Computer Communications and Networks - ICCCN 2007, Honolulu, 165–72.

Gligoroski, D., S. Markovski, L. Kocarev, and M. Gusev. 2005a. The stream cipher edon80. Submission to eSTREAM project. http://www.ecrypt.eu.org/stream/edon80p3.html.

Gligoroski, D., S. Markovski, and S. J. Knapskog. 2005b. A fix of the MD4 family of hash functions - Quasigroup fold. NIST Cryptographic Hash Workshop, Gaithersburg, MD. http://csrc.nist.gov/groups/ST/hash/documents/Gligoroski_MD4Fix.pdf.

Gligoroski, D., S. Markovski, and S. J. Knapskog. 2008a. A public key block cipher based on multivariate quadratic quasigrops. Cryptology ePrint Archive, Report 2008/320.

Gligoroski, D., S. Markovski, and S. J. Knapskog. 2008b. Multivariate quadratic trapdoor functions based on multivariate quadratic quasigroups. Proceedings of the American Conference on Applied Mathematics, Harvard, MA, 44–9.

Gligoroski, D., S. Markovski, and S. J. Knapskog. 2008c. The stream cipher edon80. In *New stream cipher designs: The eSTREAM finalists*, 152–69. Berlin: Springer-Verlag.

Gligoroski, D., S. Markovski, and V. Bakeva. 2003. On infinite class of strongly collision resistant hash functions "Edon-F" with variable length of output. Proceedings of 1st International Conference on Mathematics and Informatics for Industry, Thessaloniki, 302–8.

Gligoroski, D., V. Dimitrova, and S. Markovski. 2009. Quasigroups as Boolean functions, their equation systems and Gröbner bases. In *Gröbner bases, coding, and cryptography*, eds. M. Sala, T. Mora, L. Perret, S. Sakata, and C. Traverso. Berlin: Springer-Verlag.

Hell, M., and T. Johansson. 2008. A key recovery attack on edon80. Advances in Cryptology ASIACRYPT 2007, LNCS 4833, 568–81.

Ji, L., X. Liangyu, and G. Xu. 2008. Collision attack on nasha–512. Cryptology ePrint Archive, Report 2008/519.

Joux, A. 2004. Multi-collisions in iterated hash functions, Applications to cascades constructions. Advances in Cryptology - CRYPTO 2004, LNCS 3152, 306–16.

Katz, J., and Y. Lindell. 2008. *Introduction to modern cryptography, CRC Press Series on Discrete Mathematics and its Applications*. Boca Raton: CRC Press.

Keedwell, D., and J. Dénes. 2015. *Latin squares and their applications*. 2nd ed. Amsterdam: Elsevier.

Kelsey, J., and B. Schneier. 2005. Second preimages on *n*-bit hash functions for much less than 2*n* work. Advances in Cryptology - CRYPTO 2005, LNCS 3494, 474–90.

Kelsey, J., and T. Kohno. 2006. Herding hash functions and the nostradamus attack. Advances in Cryptology - EUROCRYPT 2006, LNCS 4004, 183–200.

Khovratovich, D., I. Nikolić, and R.-P. Weinmann. 2009. Cryptanalysis of Edon-R. University of Luxembourg.

Klimov, A., and A. Shamir. 2002. A new class of invertible mappings. In Workshop on Cryptographic Hardware and Embedded Systems (CHES).

Kościelny, C. 1996. A method of constructing quasigroup-based streamciphers. *International Journal of Applied Mathematics and Computer Science* 6:109–21.

Kościelny, C. 2002. Generating quasigroups for cryptographic applications. *International Journal of Applied Mathematics and Computer Science* 12 (4):559–69.

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

Lj, K., S. Markovski, and D. Gligoroski. 2014. Cryptographic primitives, error coding, and pseudo-random number improvement methods using quasigroups, European Patent. Patent No. EP 1 800 432 B1, Bulletin 2014/28, Dated 09.07.2014.

Lucks, S. 2004. Design principles for iterated hash functions. Cryptology ePrint Archive, Report 2004/253.

Lucks, S. 2005. A failure-friendly design principle for hash functions. ASIACRYPT 2005, LNCS 3788, 474–94.

Markovski, S. 2003. Quasigroup string processing and applications in cryptography. Proceedings 1-st International Conference on Mathematics and Informatics for industry MII 2003, 14–16 April, Thessaloniki, 278–90.

Markovski, S., and A. Mileva. 2008. NaSHA. Submission to NIST.

Markovski, S., and A. Mileva. 2009a. Generating huge quasigroups from small non- linear bijections via extended Feistel function. *Quasigroups and Related Systems* 17:97–106.

Markovski, S., and A. Mileva. 2009b. Nasha - cryptographic hash functions. NIST The First SHA-3 Candidate Conference, Leuven, Belgium, 25–28 February.

Markovski, S., A. Mileva, V. Dimitrova, and D. Gligoroski. 2009. On a conditional collision attack on nasha-512. Cryptology ePrint Archive, Report 2009/034.

Markovski, S., D. Gligoroski, and B. Stojčevska. 2000. Securetwo-wayon-line communications by using quasigroup enchipering with almost public key. *Novi Sad Journal of Mathematics* 30 (2):43–9.

Markovski, S., D. Gligoroski, and L. Kocarev. 2005. Unbiased random sequences from quasigroup string transformations. *LNCS* 3557:163–80.

Markovski, S., D. Gligoroski, and S. Andova. 1997. Using quasigroups for one-one secure encoding. Proceedings of VIII Conference on Logic and Computer Science LIRA97, Novi Sad, 157–62.

Markovski, S., D. Gligoroski, and V. Bakeva. 1999. Quasigroup string processing Part 1. Contributions: Sec. Math. Tech. Sci. MANU 20, 13–28. doi: 10.20903/csnmbs.masa.2006.27.1-2.5.

Markovski, S., D. Gligoroski, and V. Bakeva. 2001. Quasigroups and hash functions. In *Discrete mathematics and applications*, 43–50. Res. Math. Comput. Sci. 6. Blagoevgrad: South-West University.

Markovski, S., D. Gligoroski, and V. Bakeva. 2003. On infinite class of strongly collision resistant hash functions "edon–f" with variable length of output. Proceedings of the 1st International Conference on Mathematics and Informatics for Industry, Thessaloniki, 302–8.

Markovski, S., and V. Bakeva. 2001. Quasigroup string processing: Part 4. Contributions: Sec. Math. Tech. Sci. MANU 22. doi: 10.20903/csnmbs.masa.2006.27.1-2.5.

Markovski, S., and V. Kusakatov. 2000. Quasigroup string processing: Part 2. Contributions. Sec. Math. Tech. Sci., MANU 21, 15–32. doi: 10.20903/csnmbs.masa.2006.27.1-2.5.

Markovski, S., and V. Kusakatov. 2002–2003. Quasigroup string processing: Part 3. Contributions, Sec. Math. Tech. Sci., MANU 23–24, 7–27. doi: 10.20903/csnmbs.masa.2006.27.1-2.5.

Markovski, S., V. Dimitrova, and S. Samardjiska. 2010. Identity sieves for quasigroups. *Quasigroups Related Systems* 18 (2):149–63.

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

Matsumoto, M.,. M. Saito, T. Nishimura, and M. Hagita. 2007. A fast stream cipher with huge state space and quasigroup filter for software. Selected Area in Cryptography. *LNCS* 4876:246–63.

Matsumoto, M.,. M. Saito, T. Nishimura, and M. Hagita. 2008. Cryptmt3 stream cipher. New Stream Cipher Designs, LNCS 4986, 7–19.

Menezes, A. J., P. C. van Oorschot, and S. A. Vanstone. 1997. *Handbook of applied cryptography, CRC Press Series on Discrete Mathematics and its Applications*. Boca Raton: CRC Press.

Mihajloska, H., and D. Gligoroski. 2012. Construction of optimal 4-bit S-boxes by quasigroups of order 4. Proceedings of SECURWARE 2012, Rome, Italy, 163–8.

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

Mileva, A., and S. Markovski. 2010. Quasigroups string transformations and hash function design. ICT Innovations 2009, Springer Berlin Heidelberg, 367–76.

Mileva, A., and S. Markovski. 2012. Shapeless quasigroups derived by Feistel orthomorphisms. *Glasnik Matematicki* 47 (2):333–49. doi: 10.3336/gm.47.2.09.

Mohamed, M. S. E., J. Ding, and J. Buchmann. 2008. Algebraic cryptanalysis of MQQ public key cryptosystem by mutantXL. Cryptology ePrint Archive, Report 2008/451.

Moldovyan, N. A. 1998. *Problems and methods of cryptology*. St.-Petersburg: St.-Petersburg University Press (in Russian).

Moldovyan, N. A., A. A. Moldovyan, and M. E. Eremeev. 2004. *Cryptology: From primitives to the synthesis of algorithms*. St.-Petersburg: BKhV-Peterburg (in Russian).

Moufang, R. 1935. Zur Struktur von Alternativkörpern. *Mathematische Annalen* 110 (1): 416–30. doi: 10.1007/BF01448037.

Nikolić, I., and D. Knovratovich. 2008. Free–start attacks on NaSHA.

Ochadková, E., and V. Snášel. 2001. Using quasigroups for secure encoding of file system. Abstract of Talk on Conference Security and Protection of information, Brno.

Petrescu, A. 2007. Applications of quasigroups in cryptography. Proc of Inter Ing 2007.

Pflugfelder, H.O. 1990. *Quasigroups and loops: Introduction*. Berlin: Heldermann Verlag,

Pommering, K. 2005. Fourier analysis of Boolean maps: A tutorial.

Popovska-Mitrovikj, A., V. Bakeva, and S. Markovski. 2011. On random error correcting codes based on quasigroups. *Quasigroups and Related Systems* 19:301–16.

Sade, A. 1957. Quasigroups automorphes par le groupe cyclique. *Canadian Journal of Mathematics* 9:321–35.

Samardziska, S.,. S. Markovski, and D. Gligoroski. 2010. Multivariate quasigroups defined by T-functions. Proceedings of the Second International Conference on Symbolic Computation and Cryptography, University of London.

Sarvate, D. G., and J. Seberry. 1986. Encryption methods based on combinatorial designs. *Ars Combinatoria* 21A:237–46.

Shcherbacov, V. 2007. On some known possible applications of quasigroups in cryptology. SMIK.

Shcherbacov, V. 2017. *Elements of quasigroup theory and applications. Monographs and Research Notes in Mathematics*, xxi–576. Boca Raton: CRC Press.

Shcherbacov, V.A. 2003. Elements of quasigroup theory and some its applications in code theory and cryptology.

Simjanovska, M., M. Mihova, and S. Markovski. 2013. Matrix presentation of quasigroups of order 4. Proceedings of the Tenth International Conference CIIT 2013, Bitola, 192–6.

Smith, J. D. H. 1974. *An introduction to quasigroups and their representations*. Cambridge, MA: Academic Press.

Snášel, V., A. Abraham, J. Dvorsky, P. Krőmer, and J. Platš. 2009. Hash function based on large quasigroups. *LNCS* 5544:521–9.

Stinson, D. R., and M. B. Paterson. 2019. *Cryptography*. 4th ed. Textbooks in Mathematics. Boca Raton: CRC Press.

Suschkewitsch, A. K. 1929. On a generalization of the associative law. *Transactions of the American Mathematical Society* 31 (1):204–14. doi: 10.1090/S0002-9947-1929-1501476-0.

Syrbu, P.N. 2014. *Teoria quasigrupurilor. Introducere*. Chisinau: Universitatea de Stat din Moldova. (in Romanian).

Vojvoda, M., M. Sýs, and M. Jókay. 2007. A note on algebraic properties of quasigroups in edon80. SASC. Bochum, Germany.

Wu, C.-K., and D. Feng. 2016. *Boolean functions and their applications in cryptography. Advances in Computer Science and Technology*. Heidelberg: Springer.

Yu, Z., and Y. Xu. 2017. Optimal S-boxes based on 3-quasigroups of order 4. 6th International Conference on Advanced Materials and Computer Science (ICAMCS 2017).