Edon–$\mathcal{R}(256, 384, 512)$ – an Efficient Implementation of Edon–$\mathcal{R}$ Family of Cryptographic Hash Functions

Danilo Gligoroski and Svein Johan Knapskog

Centre for Quantifiable Quality of Service in Communication Systems,
Norwegian University of Science and Technology,
O.S.Bragstads plass 2E, N-7491 Trondheim, NORWAY
{Danilo.Gligoroski,Svein.J.Knapskog}@q2s.ntnu.no

Abstract. We have designed three fast implementations of recently proposed family of hash functions Edon–$\mathcal{R}$. They produce message digests of length 256, 384 and 512 bits. We have defined huge quasigroups of orders 2^{256}, 2^{384} and 2^{512} by using only bitwise operations on 32 bit values (additions modulo 2^{32}, XORs and left rotations) and achieved processing speeds of the Reference C code of 16 cycles/byte, 25.75 cycles/byte and 33.63 cycles/byte on x86 (Intel and AMD microprocessors). In this paper we give their full description, as well as an initial security analysis.

Key words: hash function, Edon–$\mathcal{R}$, quasigroup

1 Introduction

On Second NIST Hash Workshop a family of hash functions Edon–$\mathcal{R}$ was proposed [1]. The initial design was by general quasigroups of relatively small order (up to 256), and the approach was without concrete realization of those hash functions. No concrete measurements about the speed of those hash functions were given, although the authors admitted that computational speed of their design is slow.

In this paper we will describe three concrete realizations of Edon–$\mathcal{R}$ that will produce hash outputs of 256, 384 and 512 bits. We have used bitwise operations on 32 bit values (additions modulo 2^{32}, XORs and left rotations) to construct quasigroups of huge order (2^{256}, 2^{384} and 2^{512}) and then we have used those quasigroups as a basis for implementing the compression function of Edon–$\mathcal{R}$. We will show that the designed quasigroups of huge order lack some of the laws that are satisfied in groups such as commutativity and associativity. That is similar to the approach in the original proposal for the Edon–$\mathcal{R}$ family of cryptographic hash functions. Thus, we are relying our claims about the security of our concrete realization of Edon–$\mathcal{R}$ hash functions on the difficulty of solving general quasigroup equations.

The organization of the paper is as follows: In Section 2 we give some basic mathematical definitions, a definition of a general compression function of Edon–$\mathcal{R}$ with only three blocks, and a definition of three huge quasigroups of orders 2^{256}, 2^{384} and 2^{512}, in Section 3 we define three hash functions Edon–$\mathcal{R}(256, 384, 512)$, in Section 4 we give some implementation characteristics, in Section 5 we give an initial security analysis of the proposed hash functions, in Section 6 we give a design rationale and we conclude the paper by Section 7.

2 Mathematical preliminaries and notation

In this section we need to repeat some parts of the definition of the class of one-way candidate functions $\mathcal{R}_1$ recently defined in [1, 2]. For that purpose we will need also several brief definitions for quasigroups and quasigroup string transformations.

A quasigroup $(Q, *)$ is an algebraic structure consisting of a nonempty set Q and a binary operation $* : Q^2 \rightarrow Q$ with the property each of the equations $a * x = b$ and $y * a = b$ to have unique solutions x and y in Q. Closely related combinatorial structures to finite quasigroups are the Latin squares, since the main body of the multiplication table of a quasigroup is just a Latin square. More detailed information about theory of quasigroups, quasigroup string processing, Latin squares and hash functions you can find in [3–6].

For the description of the algorithm we will use the following definitions:

Definition 1. ([2] **Quasigroup reverse string transformation** $\mathcal{R}_1 : Q^r \to Q^r$)
Let r be a positive integer, let $(Q, *)$ be a quasigroup and $a_j, b_j \in Q$. For each fixed $m \in Q$ define first the transformation $Q_m : Q^r \to Q^r$ by

$$Q_m(a_0, a_1, \ldots, a_{r-1}) = (b_0, b_1, \ldots, b_{r-1}) \Longleftrightarrow b_i := \begin{cases} m * a_0, & i = 0 \\ b_{i-1} * a_i, & 1 \le i \le r-1. \end{cases}$$

Then define $\mathcal{R}_1$ as composition of transformations of kind Q_m, for suitable choices of the indexes m, as follows:

$$\mathcal{R}_1(a_0, a_1, \ldots, a_{r-1}) := Q_{a_0}(Q_{a_1} \ldots (Q_{a_{r-1}}(a_0, a_1, \ldots, a_{r-1}))).$$

	a_0	a_1	a_2
a_2	$x_0^{(1)}$	$x_1^{(1)}$	$x_2^{(1)}$
a_1	$x_0^{(2)}$	$x_1^{(2)}$	$x_2^{(2)}$
a_0	b_0	b_1	b_2

a.

	x_0	x_1	x_2
x_2	$x_0^{(1)}$	$x_1^{(1)}$	$x_2^{(1)}$
x_1	$x_0^{(2)}$	$x_1^{(2)}$	$x_2^{(2)}$
x_0	b_0	b_1	b_2

b.

Table 1. a. Schematic presentation of the function $\mathcal{R}_1$ for $r = 3$, **b.** Conjectured one-wayness of $\mathcal{R}_1$ comes from the difficulty to solve a system of three equations where b_0, b_1 and b_2 are given, and $a_0 = x_0$, $a_1 = x_1$ and $a_2 = x_2$ are indeterminate variables.

It was conjectured in [1, 2] that $\mathcal{R}_1$ is one-way function (under some assumptions about the underlying quasigroup $(Q, *)$) and that the complexity of its inverting is exponential i.e. that inverting $\mathcal{R}_1$ has a complexity $O(|Q|^{\frac{r}{3}})$, where $|Q|$ is the size of the set Q.

In our construction of Edon–$\mathcal{R}(n), n = 256, 384, 512$, we will use the function $\mathcal{R}_1$ with $r = 3$. The transformation can be schematically presented by the Table 1a.

The conjectured one-wayness of $\mathcal{R}_1$ can be explained by Table 1b. Namely, let us take that only the values b_0, b_1 and b_2 are given. Then, in order to find pre-image values $a_0 = x_0$, $a_1 = x_1$ and $a_2 = x_2$ we can use the Definition 1 and we will obtain the following equalities for the elements of Table 1b:
$x_0^{(1)} = x_2 * x_0$; $x_1^{(1)} = (x_2 * x_0) * x_1$; $x_2^{(1)} = ((x_2 * x_0) * x_1) * x_2$; $x_0^{(2)} = x_1 * (x_2 * x_0)$; $x_1^{(2)} = (x_1 * (x_2 * x_0)) * ((x_2 * x_0) * x_1)$; $x_2^{(2)} = \big((x_1 * (x_2 * x_0)) * ((x_2 * x_0) * x_1)\big) * \big(((x_2 * x_0) * x_1) * x_2\big)$.

From them, we can obtain the following system of quasigroup equations with indeterminate x_0, x_1, x_2:

$$\begin{cases} b_0 = x_0 * (x_1 * (x_2 * x_0)) \\ b_1 = b_0 * \big((x_1 * (x_2 * x_0)) * ((x_2 * x_0) * x_1)\big) \\ b_2 = b_1 * \Big(\big((x_1 * (x_2 * x_0)) * ((x_2 * x_0) * x_1)\big) * \big(((x_2 * x_0) * x_1) * x_2\big)\Big). \end{cases}$$

One can show that for any given $a_0 = x_0 \in Q$ either there are values of $a_1 = x_1$ and $a_2 = x_2$ as a solution or there is no solution. However, if the quasigroup operation is non-commutative and non-associative, and if the size of the quasigroup is very big (for example 2^{256}, 2^{384} or 2^{512}) then solving this simple system of three quasigroup equations is hard. Actually there is no known efficient method for solving such systems of quasigroup equations.

Of coarse, one inefficient method for solving that system would be to try every possible value for $a_0 = x_0 \in Q$ until obtaining other two indeterminates $a_1 = x_1$ and $a_2 = x_2$. That brute force method would require in average $\frac{1}{2}|Q|$ attempts to guess $a_0 = x_0 \in Q$ before solving the system.

2.1 Definition of quasigroups of huge order

In this section we will describe the construction of quasigroups of huge orders (2^{256}, 2^{384} and 2^{512}). We will use the following notation: Q is a set of cardinality 2^n, and elements $x \in Q$ will be represented in their bitwise form as n-bit words

$$x \equiv (\overline{x}_0, \overline{x}_1, \ldots, \overline{x}_{n-2}, \overline{x}_{n-1}) \equiv \overline{x}_0 \cdot 2^{n-1} + \overline{x}_1 \cdot 2^{n-2} + \ldots + \overline{x}_{n-2} \cdot 2 + \overline{x}_{n-1}$$

where $\overline{x}_i \in \{0, 1\}$.

Let us start first with the following simple (and obvious) proposition:

Proposition 1. *For any finite set Q of cardinality 2^n the operation "Bitwise eXlusive OR" $\oplus_n : Q^2 \rightarrow Q$ is quasigroup operation.* □

Let $\pi_1, \pi_2, \pi_3 : Q \rightarrow Q$ be three permutations on the set Q.

Proposition 2. *For any finite set Q of cardinality 2^n the operation $* : Q^2 \rightarrow Q$ defined as:*

$$a * b \equiv \pi_1(\pi_2(a) \oplus_n \pi_3(b))$$

is a quasigroup operation. □

Proposition 3. *If permutations π_2 and π_3 are not equal, then the quasigroup $(Q, *)$ is non-commutative.* □

Let us denote by $Q_{256} = \{0, 1\}^{256}$, $Q_{384} = \{0, 1\}^{384}$ and $Q_{512} = \{0, 1\}^{512}$ the corresponding sets of 256–bit, 384–bit and 512–bit words. Since our intention is to define Edon–$\mathcal{R}$ by bitwise operations on 32 bit values, we will introduce the following convention: elements $X \in Q_{256}$ will be represented as $X = (X_0, X_1, \ldots, X_7)$, elements $X \in Q_{384}$ will be represented as $X = (X_0, X_1, \ldots, X_{11})$, and elements $X \in Q_{512}$ will be represented as $X = (X_0, X_1, \ldots, X_{15})$, where X_i are 32–bit words.

Further, let us denote by $ROTL(Y, k)$ left rotation of a 32–bit word Y by k positions, by $Y \oplus Z$ ordinary bitwise XOR operations between two 32–bit words Y and Z, and by $Y + Z$ addition modulo 2^{32}.

We will give the formal definitions for the following permutations: $\pi_{1,256}, \pi_{2,256}, \pi_{3,256}, \pi_{1,384}, \pi_{2,384}, \pi_{3,384}, \pi_{1,512}, \pi_{2,512}, \pi_{3,512}$ where the corresponding three digit index (256, 384 or 512) denotes the cardinality of the set Q over which they are defined.

Definition 2. *Transformation $\pi_{1,256} : Q_{256} \rightarrow Q_{256}$ is defined as:*

$$\pi_{1,256}(X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7) = (X_5, X_6, X_7, X_0, X_1, X_2, X_3, X_4)$$

Lemma 1. *Transformation $\pi_{1,256}$ is permutation.* □

Definition 3. *Transformation $\pi_{2,256} : Q_{256} \rightarrow Q_{256}$ is defined as:*

$$\pi_{2,256}(X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7) = (Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Y_7)$$

where

$$\begin{cases} T_0 = ROTL((X_1 + X_2 + X_4 + X_6 + X_7), 1); \\ T_1 = ROTL((X_0 + X_1 + X_3 + X_4 + X_7), 3); \\ T_2 = ROTL((X_0 + X_1 + X_2 + X_6 + X_7), 4); \\ T_3 = ROTL((X_1 + X_3 + X_4 + X_5 + X_6), 5); \\ T_4 = ROTL((X_0 + X_3 + X_4 + X_5 + X_6), 7); \\ T_5 = ROTL((X_0 + X_2 + X_4 + X_5 + X_7), 8); \\ T_6 = ROTL((X_0 + X_1 + X_2 + X_3 + X_5), 10); \\ T_7 = ROTL((X_2 + X_3 + X_5 + X_6 + X_7), 13); \end{cases} \quad and \quad \begin{cases} Y_0 = ROTL((T_0 \oplus T_3 \oplus T_5), 1); \\ Y_1 = ROTL((T_2 \oplus T_5 \oplus T_6), 4); \\ Y_2 = ROTL((T_3 \oplus T_4 \oplus T_5), 8); \\ Y_3 = ROTL((T_0 \oplus T_2 \oplus T_7), 9); \\ Y_4 = ROTL((T_1 \oplus T_2 \oplus T_7), 10); \\ Y_5 = ROTL((T_1 \oplus T_3 \oplus T_6), 12); \\ Y_6 = ROTL((T_4 \oplus T_6 \oplus T_7), 13); \\ Y_7 = ROTL((T_0 \oplus T_1 \oplus T_4), 14); \end{cases}$$

Lemma 2. *Transformation $\pi_{2,256}$ is permutation.*

Proof. It is elementary exercise to check that the matrix $A_{1,1} = \begin{pmatrix} 0&1&1&0&1&0&1&1\\ 1&1&0&1&1&0&0&1\\ 1&1&1&0&0&0&1&1\\ 0&1&0&1&1&1&1&0\\ 1&0&0&1&1&1&1&0\\ 1&0&1&0&1&1&0&1\\ 1&1&1&1&0&1&0&0\\ 0&0&1&1&0&1&1&1 \end{pmatrix}$ which correspond to the additions for obtaining temporal variables T_i is nonsingular in $(\mathbb{Z}_{2^{32}}, +)$. Thus the operations of additions are permutations over Q_{256}.

Similarly, the matrix $A_{1,2} = \begin{pmatrix} 1&0&0&1&0&1&0&0\\ 0&0&1&0&0&1&1&0\\ 0&0&0&1&1&1&0&0\\ 1&0&1&0&0&0&0&1\\ 0&1&1&0&0&0&0&1\\ 0&1&0&1&0&0&1&0\\ 0&0&0&0&1&0&1&1\\ 1&1&0&0&1&0&0&0 \end{pmatrix}$ which correspond to the bitwise XoRs for obtaining final values Y_i is nonsingular in $GF(2)$, so the operations of XoRs are permutations over Q_{256}.

Since the left rotations are also permutations, by composition of all permutations we get that the transformation $\pi_{2,256}$ is permutation. □

Definition 4. *Transformation $\pi_{3,256} : Q_{256} \to Q_{256}$ is defined as:*

$$\pi_{3,256}(X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7) = (Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Y_7)$$

where

$$\begin{cases} T_0 = ROTL((X_1 + X_4 + X_5 + X_6 + X_7), 2);\\ T_1 = ROTL((X_1 + X_2 + X_4 + X_6 + X_7), 5);\\ T_2 = ROTL((X_0 + X_1 + X_3 + X_6 + X_7), 6);\\ T_3 = ROTL((X_0 + X_2 + X_3 + X_5 + X_6), 7);\\ T_4 = ROTL((X_2 + X_3 + X_4 + X_5 + X_7), 8);\\ T_5 = ROTL((X_0 + X_3 + X_4 + X_5 + X_6), 10);\\ T_6 = ROTL((X_0 + X_1 + X_2 + X_3 + X_4), 11);\\ T_7 = ROTL((X_0 + X_1 + X_2 + X_5 + X_7), 14); \end{cases} \quad and \quad \begin{cases} Y_0 = ROTL((T_0 \oplus T_2 \oplus T_3), 3);\\ Y_1 = ROTL((T_0 \oplus T_3 \oplus T_5), 4);\\ Y_2 = ROTL((T_2 \oplus T_4 \oplus T_5), 6);\\ Y_3 = ROTL((T_1 \oplus T_4 \oplus T_7), 8);\\ Y_4 = ROTL((T_0 \oplus T_1 \oplus T_6), 9);\\ Y_5 = ROTL((T_1 \oplus T_2 \oplus T_7), 11);\\ Y_6 = ROTL((T_5 \oplus T_6 \oplus T_7), 12);\\ Y_7 = ROTL((T_3 \oplus T_4 \oplus T_6), 13); \end{cases}$$

Without proof (since it is similar to the proof for $\pi_{2,256}$) we give the following lemma:

Lemma 3. *Transformation $\pi_{3,256}$ is permutation.* □

Theorem 1. *Operation $*_{256} : Q_{256}^2 \to Q_{256}$ defined as:*

$$a *_{256} b = \pi_{1,256}(\pi_{2,256}(a) \quad \oplus_{256} \quad \pi_{3,256}(b))$$

is a non-commutative and non-associative quasigroup operation.

Proof. The proof that the operation $*_{256}$ is quasigroup operation follows immediately from the previous propositions and lemmas. The only non-obvious part is that the operation is non-associative. However, it can be easily checked that for example

$$(1 *_{256} 2) *_{256} 3 \neq 1 *_{256} (2 *_{256} 3)$$

where 1, 2 and 3 are represented as 256–bit words. □

Definition 5. *Transformation* $\pi_{1,384} : Q_{384} \to Q_{384}$ *is defined as:*

$$\pi_{1,384}(X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9, X_{10}, X_{11}) = (X_7, X_8, X_9, X_{10}, X_{11}, X_0, X_1, X_2, X_3, X_4, X_5, X_6)$$

Lemma 4. *Transformation* $\pi_{1,384}$ *is permutation.* □

Definition 6. *Transformation* $\pi_{2,384} : Q_{384} \to Q_{384}$ *is defined as:*

$$\pi_{2,384}(X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9, X_{10}, X_{11}) = (Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Y_7, Y_8, Y_9, Y_{10}, Y_{11})$$

where

$$\begin{cases} T_0 = ROTL((X_1 + X_2 + X_3 + X_7 + X_8 + X_{10} + X_{11}), 1); \\ T_1 = ROTL((X_0 + X_1 + X_4 + X_5 + X_6 + X_8 + X_{10}), 3); \\ T_2 = ROTL((X_0 + X_2 + X_4 + X_5 + X_9 + X_{10} + X_{11}), 4); \\ T_3 = ROTL((X_1 + X_3 + X_4 + X_5 + X_6 + X_7 + X_{11}), 5); \\ T_4 = ROTL((X_0 + X_1 + X_3 + X_7 + X_8 + X_9 + X_{11}), 7); \\ T_5 = ROTL((X_0 + X_2 + X_4 + X_5 + X_6 + X_7 + X_8), 8); \\ T_6 = ROTL((X_0 + X_1 + X_2 + X_3 + X_4 + X_9 + X_{10}), 10); \\ T_7 = ROTL((X_0 + X_2 + X_5 + X_6 + X_8 + X_9 + X_{11}), 13); \\ T_8 = X_3 + X_4 + X_5 + X_7 + X_8 + X_9 + X_{10}; \\ T_9 = X_0 + X_3 + X_4 + X_6 + X_7 + X_{10} + X_{11}; \\ T_{10} = X_1 + X_2 + X_5 + X_6 + X_7 + X_9 + X_{10}; \\ T_{11} = X_1 + X_2 + X_3 + X_6 + X_8 + X_9 + X_{11}; \end{cases} \quad and \quad \begin{cases} Y_0 = ROTL((T_0 \oplus T_4 \oplus T_5 \oplus T_6 \oplus T_9), 1); \\ Y_1 = ROTL((T_2 \oplus T_3 \oplus T_7 \oplus T_9 \oplus T_{11}), 4); \\ Y_2 = ROTL((T_1 \oplus T_3 \oplus T_6 \oplus T_7 \oplus T_8), 8); \\ Y_3 = ROTL((T_0 \oplus T_2 \oplus T_8 \oplus T_9 \oplus T_{10}), 9); \\ Y_4 = ROTL((T_2 \oplus T_4 \oplus T_5 \oplus T_6 \oplus T_{10}), 10); \\ Y_5 = ROTL((T_1 \oplus T_3 \oplus T_9 \oplus T_{10} \oplus T_{11}), 12); \\ Y_6 = ROTL((T_5 \oplus T_6 \oplus T_7 \oplus T_8 \oplus T_{11}), 13); \\ Y_7 = ROTL((T_1 \oplus T_3 \oplus T_4 \oplus T_7 \oplus T_{10}), 14); \\ Y_8 = T_0 \oplus T_1 \oplus T_2 \oplus T_6 \oplus T_{11}; \\ Y_9 = T_1 \oplus T_2 \oplus T_5 \oplus T_8 \oplus T_9; \\ Y_{10} = T_0 \oplus T_3 \oplus T_4 \oplus T_8 \oplus T_{11}; \\ Y_{11} = T_0 \oplus T_4 \oplus T_5 \oplus T_7 \oplus T_{10}; \end{cases}$$

Lemma 5. *Transformation* $\pi_{2,384}$ *is permutation.* □

Definition 7. *Transformation* $\pi_{3,384} : Q_{384} \to Q_{384}$ *is defined as:*

$$\pi_{3,384}(X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9, X_{10}, X_{11}) = (Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Y_7, Y_8, Y_9, Y_{10}, Y_{11})$$

where

$$\begin{cases} T_0 = ROTL((X_4 + X_5 + X_6 + X_7 + X_9 + X_{10} + X_{11}), 2); \\ T_1 = ROTL((X_1 + X_2 + X_3 + X_7 + X_8 + X_{10} + X_{11}), 5); \\ T_2 = ROTL((X_0 + X_1 + X_2 + X_3 + X_7 + X_9 + X_{10}), 6); \\ T_3 = ROTL((X_0 + X_2 + X_4 + X_5 + X_7 + X_8 + X_9), 7); \\ T_4 = ROTL((X_1 + X_3 + X_4 + X_6 + X_7 + X_9 + X_{11}), 8); \\ T_5 = ROTL((X_0 + X_1 + X_2 + X_5 + X_6 + X_9 + X_{10}), 10); \\ T_6 = ROTL((X_0 + X_1 + X_4 + X_6 + X_8 + X_{10} + X_{11}), 11); \\ T_7 = ROTL((X_0 + X_2 + X_3 + X_4 + X_8 + X_9 + X_{11}), 14); \\ T_8 = X_0 + X_1 + X_5 + X_6 + X_7 + X_8 + X_{11}; \\ T_9 = X_0 + X_3 + X_5 + X_6 + X_8 + X_9 + X_{10}; \\ T_{10} = X_2 + X_3 + X_4 + X_5 + X_6 + X_7 + X_8; \\ T_{11} = X_1 + X_2 + X_3 + X_4 + X_5 + X_{10} + X_{11}; \end{cases} \quad and \quad \begin{cases} Y_0 = ROTL((T_0 \oplus T_1 \oplus T_2 \oplus T_3 \oplus T_8), 3); \\ Y_1 = ROTL((T_0 \oplus T_4 \oplus T_5 \oplus T_6 \oplus T_9), 4); \\ Y_2 = ROTL((T_4 \oplus T_5 \oplus T_6 \oplus T_8 \oplus T_{11}), 6); \\ Y_3 = ROTL((T_1 \oplus T_3 \oplus T_6 \oplus T_{10} \oplus T_{11}), 8); \\ Y_4 = ROTL((T_0 \oplus T_2 \oplus T_5 \oplus T_8 \oplus T_{10}), 9); \\ Y_5 = ROTL((T_3 \oplus T_4 \oplus T_7 \oplus T_8 \oplus T_{11}), 11); \\ Y_6 = ROTL((T_2 \oplus T_3 \oplus T_5 \oplus T_7 \oplus T_9), 12); \\ Y_7 = ROTL((T_1 \oplus T_5 \oplus T_6 \oplus T_7 \oplus T_{10}), 13); \\ Y_8 = T_2 \oplus T_3 \oplus T_4 \oplus T_9 \oplus T_{10}; \\ Y_9 = T_1 \oplus T_2 \oplus T_4 \oplus T_7 \oplus T_{11}; \\ Y_{10} = T_0 \oplus T_1 \oplus T_9 \oplus T_{10} \oplus T_{11}; \\ Y_{11} = T_0 \oplus T_6 \oplus T_7 \oplus T_8 \oplus T_9; \end{cases}$$

Lemma 6. *Transformation* $\pi_{3,384}$ *is permutation.* □

Theorem 2. *Operation* $*_{384} : Q_{384}^2 \to Q_{384}$ *defined as:*

$$a *_{384} b = \pi_{1,384}(\pi_{2,384}(a) \quad \oplus_{384} \quad \pi_{3,384}(b))$$

is a non-commutative and non-associative quasigroup operation. □

Definition 8. *Transformation* $\pi_{1,512} : Q_{384} \to Q_{384}$ *is defined as:*

$$\pi_{1,384}(X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9, X_{10}, X_{11}, X_{12}, X_{13}, X_{14}, X_{15}) = \\ (X_9, X_{10}, X_{11}, X_{12}, X_{13}, X_{14}, X_{15}, X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8)$$

Lemma 7. *Transformation $\pi_{1,512}$ is permutation.* □

Definition 9. *Transformation $\pi_{2,512} : Q_{512} \to Q_{512}$ is defined as:*

$$\begin{aligned}\pi_{2,512}(&X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9, X_{10}, X_{11}, X_{12}, X_{13}, X_{14}, X_{15}) = \\ &(Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Y_7, Y_8, Y_9, Y_{10}, Y_{11}, Y_{12}, Y_{13}, Y_{14}, Y_{15})\end{aligned}$$

where

$$\left\{\begin{array}{l}
T_0 = ROTL((X_0 + X_2 + X_3 + X_4 + X_6 + X_9 + X_{10} + X_{13} + X_{15}), 1);\\
T_1 = ROTL((X_0 + X_1 + X_3 + X_4 + X_6 + X_8 + X_{10} + X_{13} + X_{15}), 3);\\
T_2 = ROTL((X_0 + X_1 + X_3 + X_6 + X_7 + X_{10} + X_{11} + X_{12} + X_{14}), 4);\\
T_3 = ROTL((X_1 + X_3 + X_4 + X_6 + X_8 + X_{10} + X_{11} + X_{14} + X_{15}), 5);\\
T_4 = ROTL((X_0 + X_1 + X_2 + X_4 + X_7 + X_8 + X_9 + X_{11} + X_{13}), 7);\\
T_5 = ROTL((X_1 + X_2 + X_4 + X_5 + X_7 + X_8 + X_9 + X_{10} + X_{12}), 8);\\
T_6 = ROTL((X_0 + X_5 + X_7 + X_9 + X_{10} + X_{11} + X_{12} + X_{13} + X_{14}), 10);\\
T_7 = ROTL((X_0 + X_3 + X_5 + X_8 + X_9 + X_{11} + X_{12} + X_{13} + X_{14}), 13);\\
T_8 = X_0 + X_1 + X_2 + X_3 + X_4 + X_5 + X_9 + X_{13} + X_{15};\\
T_9 = X_0 + X_1 + X_2 + X_3 + X_5 + X_6 + X_8 + X_9 + X_{15};\\
T_{10} = X_1 + X_2 + X_6 + X_8 + X_{11} + X_{12} + X_{13} + X_{14} + X_{15};\\
T_{11} = X_3 + X_5 + X_7 + X_{10} + X_{11} + X_{12} + X_{13} + X_{14} + X_{15};\\
T_{12} = X_2 + X_3 + X_5 + X_6 + X_7 + X_9 + X_{12} + X_{13} + X_{14};\\
T_{13} = X_0 + X_4 + X_5 + X_6 + X_7 + X_9 + X_{11} + X_{12} + X_{15};\\
T_{14} = X_2 + X_4 + X_5 + X_6 + X_7 + X_8 + X_{10} + X_{11} + X_{14};\\
T_{15} = X_1 + X_2 + X_4 + X_7 + X_8 + X_{10} + X_{12} + X_{14} + X_{15};
\end{array}\right. \quad and \quad \left\{\begin{array}{l}
Y_0 = ROTL((T_1 \oplus T_5 \oplus T_7 \oplus T_8 \oplus T_{11} \oplus T_{12} \oplus T_{14}), 1);\\
Y_1 = ROTL((T_2 \oplus T_5 \oplus T_7 \oplus T_9 \oplus T_{11} \oplus T_{12} \oplus T_{14}), 4);\\
Y_2 = ROTL((T_2 \oplus T_4 \oplus T_5 \oplus T_8 \oplus T_9 \oplus T_{13} \oplus T_{15}), 8);\\
Y_3 = ROTL((T_0 \oplus T_2 \oplus T_5 \oplus T_7 \oplus T_9 \oplus T_{12} \oplus T_{13}), 9);\\
Y_4 = ROTL((T_3 \oplus T_5 \oplus T_6 \oplus T_{10} \oplus T_{12} \oplus T_{14} \oplus T_{15}), 10);\\
Y_5 = ROTL((T_0 \oplus T_3 \oplus T_6 \oplus T_{11} \oplus T_{13} \oplus T_{14} \oplus T_{15}), 12);\\
Y_6 = ROTL((T_1 \oplus T_2 \oplus T_3 \oplus T_4 \oplus T_6 \oplus T_8 \oplus T_{15}), 13);\\
Y_7 = ROTL((T_1 \oplus T_2 \oplus T_4 \oplus T_6 \oplus T_7 \oplus T_{10} \oplus T_{15}), 14);\\
Y_8 = T_6 \oplus T_7 \oplus T_8 \oplus T_{10} \oplus T_{11} \oplus T_{12} \oplus T_{14};\\
Y_9 = T_4 \oplus T_7 \oplus T_{10} \oplus T_{11} \oplus T_{12} \oplus T_{13} \oplus T_{14};\\
Y_{10} = T_0 \oplus T_3 \oplus T_4 \oplus T_5 \oplus T_7 \oplus T_9 \oplus T_{10};\\
Y_{11} = T_0 \oplus T_1 \oplus T_2 \oplus T_4 \oplus T_6 \oplus T_8 \oplus T_9;\\
Y_{12} = T_0 \oplus T_1 \oplus T_4 \oplus T_8 \oplus T_{10} \oplus T_{11} \oplus T_{15};\\
Y_{13} = T_1 \oplus T_2 \oplus T_3 \oplus T_8 \oplus T_{10} \oplus T_{13} \oplus T_{14};\\
Y_{14} = T_0 \oplus T_1 \oplus T_3 \oplus T_9 \oplus T_{12} \oplus T_{13} \oplus T_{15};\\
Y_{15} = T_0 \oplus T_3 \oplus T_5 \oplus T_6 \oplus T_9 \oplus T_{11} \oplus T_{13};
\end{array}\right.$$

Lemma 8. *Transformation $\pi_{2,512}$ is permutation.* □

Definition 10. *Transformation $\pi_{3,512} : Q_{512} \to Q_{512}$ is defined as:*

$$\begin{aligned}\pi_{3,512}(&X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9, X_{10}, X_{11}, X_{12}, X_{13}, X_{14}, X_{15}) = \\ &(Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Y_7, Y_8, Y_9, Y_{10}, Y_{11}, Y_{12}, Y_{13}, Y_{14}, Y_{15})\end{aligned}$$

where

$$\left\{\begin{array}{l}
T_0 = ROTL((X_1 + X_2 + X_3 + X_4 + X_8 + X_{10} + X_{11} + X_{13} + X_{14}), 2);\\
T_1 = ROTL((X_0 + X_1 + X_3 + X_6 + X_7 + X_{11} + X_{13} + X_{14} + X_{15}), 5);\\
T_2 = ROTL((X_0 + X_3 + X_5 + X_6 + X_8 + X_{10} + X_{12} + X_{14} + X_{15}), 6);\\
T_3 = ROTL((X_0 + X_2 + X_4 + X_6 + X_9 + X_{10} + X_{11} + X_{12} + X_{15}), 7);\\
T_4 = ROTL((X_0 + X_3 + X_4 + X_5 + X_7 + X_9 + X_{10} + X_{12} + X_{13}), 8);\\
T_5 = ROTL((X_2 + X_4 + X_5 + X_6 + X_8 + X_{10} + X_{12} + X_{14} + X_{15}), 10);\\
T_6 = ROTL((X_1 + X_3 + X_4 + X_7 + X_8 + X_9 + X_{11} + X_{12} + X_{14}), 11);\\
T_7 = ROTL((X_2 + X_4 + X_5 + X_7 + X_9 + X_{11} + X_{13} + X_{14} + X_{15}), 14);\\
T_8 = X_1 + X_5 + X_6 + X_7 + X_9 + X_{10} + X_{11} + X_{13} + X_{14};\\
T_9 = X_0 + X_1 + X_2 + X_3 + X_7 + X_8 + X_9 + X_{10} + X_{15};\\
T_{10} = X_0 + X_1 + X_2 + X_3 + X_5 + X_6 + X_{11} + X_{12} + X_{14};\\
T_{11} = X_0 + X_1 + X_5 + X_6 + X_7 + X_8 + X_{11} + X_{12} + X_{13};\\
T_{12} = X_1 + X_3 + X_4 + X_5 + X_8 + X_{10} + X_{12} + X_{13} + X_{15};\\
T_{13} = X_0 + X_1 + X_2 + X_4 + X_6 + X_7 + X_8 + X_9 + X_{10};\\
T_{14} = X_0 + X_2 + X_7 + X_8 + X_9 + X_{12} + X_{13} + X_{14} + X_{15};\\
T_{15} = X_2 + X_3 + X_4 + X_5 + X_6 + X_9 + X_{11} + X_{13} + X_{15};
\end{array}\right. \quad and \quad \left\{\begin{array}{l}
Y_0 = ROTL((T_0 \oplus T_5 \oplus T_6 \oplus T_7 \oplus T_9 \oplus T_{12} \oplus T_{15}), 3);\\
Y_1 = ROTL((T_2 \oplus T_4 \oplus T_5 \oplus T_8 \oplus T_9 \oplus T_{10} \oplus T_{12}), 4);\\
Y_2 = ROTL((T_1 \oplus T_2 \oplus T_4 \oplus T_7 \oplus T_9 \oplus T_{11} \oplus T_{13}), 6);\\
Y_3 = ROTL((T_1 \oplus T_3 \oplus T_5 \oplus T_7 \oplus T_8 \oplus T_{13} \oplus T_{14}), 8);\\
Y_4 = ROTL((T_1 \oplus T_2 \oplus T_6 \oplus T_8 \oplus T_{11} \oplus T_{14} \oplus T_{15}), 9);\\
Y_5 = ROTL((T_0 \oplus T_1 \oplus T_3 \oplus T_7 \oplus T_9 \oplus T_{11} \oplus T_{13}), 11);\\
Y_6 = ROTL((T_0 \oplus T_2 \oplus T_5 \oplus T_6 \oplus T_{10} \oplus T_{13} \oplus T_{15}), 12);\\
Y_7 = ROTL((T_0 \oplus T_1 \oplus T_3 \oplus T_6 \oplus T_8 \oplus T_{10} \oplus T_{12}), 13);\\
Y_8 = T_0 \oplus T_2 \oplus T_3 \oplus T_4 \oplus T_8 \oplus T_{12} \oplus T_{15};\\
Y_9 = T_4 \oplus T_5 \oplus T_6 \oplus T_{11} \oplus T_{12} \oplus T_{13} \oplus T_{14};\\
Y_{10} = T_4 \oplus T_7 \oplus T_8 \oplus T_9 \oplus T_{10} \oplus T_{13} \oplus T_{15};\\
Y_{11} = T_2 \oplus T_3 \oplus T_4 \oplus T_9 \oplus T_{10} \oplus T_{14} \oplus T_{15};\\
Y_{12} = T_0 \oplus T_2 \oplus T_6 \oplus T_7 \oplus T_9 \oplus T_{11} \oplus T_{14};\\
Y_{13} = T_3 \oplus T_5 \oplus T_{11} \oplus T_{12} \oplus T_{13} \oplus T_{14} \oplus T_{15};\\
Y_{14} = T_1 \oplus T_3 \oplus T_4 \oplus T_5 \oplus T_6 \oplus T_{10} \oplus T_{11};\\
Y_{15} = T_0 \oplus T_1 \oplus T_7 \oplus T_8 \oplus T_{10} \oplus T_{12} \oplus T_{14};
\end{array}\right.$$

Lemma 9. *Transformation $\pi_{3,512}$ is permutation.* □

Theorem 3. *Operation $*_{512} : Q_{512}^2 \to Q_{512}$ defined as:*

$$a *_{512} b = \pi_{1,512}(\pi_{2,512}(a) \quad \oplus_{512} \quad \pi_{3,512}(b))$$

is a non-commutative and non-associative quasigroup operation. □

Having defined three quasigroup operations $*_{256}$, $*_{384}$ and $*_{512}$ we will define three one-way functions $\mathcal{R}_{1,256}$, $\mathcal{R}_{1,384}$ and $\mathcal{R}_{1,512}$ as follows:

Definition 11.
1. *$\mathcal{R}_{1,256} : Q_{256}^3 \to Q_{256}^3 \equiv \mathcal{R}_1$ where $\mathcal{R}_1$ is defined as in Definition 1 over Q_{256} with the quasigroup operation $*_{256}$.*
2. *$\mathcal{R}_{1,384} : Q_{384}^3 \to Q_{384}^3 \equiv \mathcal{R}_1$ where $\mathcal{R}_1$ is defined as in Definition 1 over Q_{384} with the quasigroup operation $*_{384}$.*
3. *$\mathcal{R}_{1,512} : Q_{512}^3 \to Q_{512}^3 \equiv \mathcal{R}_1$ where $\mathcal{R}_1$ is defined as in Definition 1 over Q_{512} with the quasigroup operation $*_{512}$.*

3 Edon–$\mathcal{R}(256, 384, 512)$ hash algorithm

Having one-way quasigroup functions $\mathcal{R}_{1,256}$, $\mathcal{R}_{1,384}$ and $\mathcal{R}_{1,512}$, we now define three hash algorithms Edon–$\mathcal{R}$(256), Edon–$\mathcal{R}$(384) and Edon–$\mathcal{R}$(512) that map a messages M of arbitrary length of l bits ($l \leq 2^{128}$) into a hash value of 256, 384 or 512 bits.

3.1 Padding

Padding of the messages M of arbitrary length of l bits is done by the standard Merkle-Damgård strengthening. Let us shortly denote all three hash functions as Edon–$\mathcal{R}(n)$ where the parameter n can take the values 256, 384 or 512.

The padding of a message M that is long l bits by Edon–$\mathcal{R}(n)$ is done by the following procedure:

1. Append the bit 1 at the end of the message.
2. Append the smallest amount l_1 of zero bits, such that $l + 1 + l_1 + 128 \equiv 0 \ (\texttt{mod}\ n)$.
3. Represent the original length l of the message M as an 128–bit number and append it at the end of the message. The length of the appended message M' becomes multiple of n bits. Let represent the appended message as $M' = M_1 M_2 \dots M_N$ where M_i is n–bit long block.

3.2 Initial predetermined values

The definition of Edon–$\mathcal{R}(n)$ hash function includes one initial string H_0 of length $2n$ bits. That initial string is given as follows (represented in hexadecimal as concatenation of 32-bits chunks):

1. For $n = 256$, $H_0 =$ `0x01020304, 0x05060708, 0x090A0B0C, 0x0D0E0F10, 0x11121314, 0x15161718, 0x191A1B1C, 0x1D1E1F20, 0x21222324, 0x25262728, 0x292A2B2C, 0x2D2E2F30, 0x31323334, 0x35363738, x393A3B3C, 0x3D3E3F40.`
2. For $n = 384$, $H_0 =$ `0x01020304, 0x05060708, 0x090A0B0C, 0x0D0E0F10, 0x11121314, 0x15161718, 0x191A1B1C, 0x1D1E1F20, 0x21222324, 0x25262728, 0x292A2B2C, 0x2D2E2F30, 0x31323334, 0x35363738, 0x393A3B3C, 0x3D3E3F40, 0x41424344, 0x45464748, 0x494A4B4C, 0x4D4E4F50, 0x51525354, 0x55565758, 0x595A5B5C, 0x5D5E5F60.`
3. For $n = 512$, $H_0 =$ `0x01020304, 0x05060708, 0x090A0B0C, 0x0D0E0F10, 0x11121314, 0x15161718, 0x191A1B1C, 0x1D1E1F20, 0x21222324, 0x25262728, 0x292A2B2C, 0x2D2E2F30, 0x31323334, 0x35363738, 0x393A3B3C, 0x3D3E3F40, 0x41424344, 0x45464748, 0x494A4B4C, 0x4D4E4F50, 0x51525354, 0x55565758, 0x595A5B5C, 0x5D5E5F60, 0x61626364, 0x65666768, 0x696A6B6C, 0x6D6E6F70, 0x71727374, 0x75767778, 0x797A7B7C, 0x7D7E7F80.`

The initial values are obtained by concatenation of the 8–bit representation of the numbers $1, 2, \dots, 128$.

3.3 Edon–$\mathcal{R}(n)$ hash function

Input: n and M, where: n is 256, 384 or 512, and M is the message to be hashed.
Output: A hash of length n bits.

1. Pad the message M, so the length of the padded message M' is multiple of n–bit words i.e. $|M'| = N \times n$.
2. Initialize H_0.
3. Compute the hash with the following iterative procedure:

$$\text{For } i = 1 \text{ to } N \text{ do}$$
$$H_i = \mathcal{R}_{1,n}(H_{i-1} || M_i) \ \texttt{mod}\ 2^{2n};$$

Output:

$$\textit{Edon-}\mathcal{R}(n)(M) = H_N \ \texttt{mod}\ 2^n$$

Since the one-way functions $\mathcal{R}_{1,n}$ are considered as transformations $\{0,1\}^{3n} \rightarrow \{0,1\}^{3n}$ for obtaining the intermediate value H_i, we apply the operation $\texttt{mod}\ 2^{2n}$ that takes the last two n–bit words from the result of $\mathcal{R}_{1,n}$. Finally, since the requested output from the hash function is n bits, we take just the last n–bit word from the H_N and that is denoted as the operation $\texttt{mod}\ 2^n$.

4 Implementation characteristics of Edon–$\mathcal{R}(256, 384, 512)$

We have initial implementation of all three functions Edon–$\mathcal{R}(256, 384, 512)$ in C. We have run tests compiling both on Microsoft Visual C++ 6.0 and GNU C for x86 processors in 32–bit mode. Microsoft compiler gave around 30% – 40% faster code. However, in both cases we did not use 64 or 128 bit SSE and SSE2 registers as well as their SIMD capabilities. The initial processing speeds obtained by Microsoft Visual C++ 6.0 compiler (optimized for speed) are given in the Table 2.

n	cycles/byte
256	16.01
384	25.75
512	33.63

Table 2. Speed of the Reference C code for Edon–$\mathcal{R}(n)$ on x86 platforms in 32–bit mode.

We project that significant improvements (at least twofold increasing) in the speed can be achieved by using SIMD instructions and capabilities of modern CPUs.

On the other hand, measuring of the performances of Edon–$\mathcal{R}(256, 384, 512)$ on 8–bit platforms still have to be done, but we hope that the speeds will be relatively fast due to the fact that we are using only basic 32–bit operations such as addition modulo 2^{32}, eXlusive OR and rotations.

By careful analysis of the order of operations performed in Edon–$\mathcal{R}(256, 384, 512)$ one can notice that there are two types of parallelism of operations:

1. Operations inside the permutations π_2 and π_3 can be executed in parallel.
2. Pipelining of quasigroup operations: after the first quasigroup operation in the firs row, two quasigroup operations can be performed in parallel (one on the first row and one on the second row), and then similarly three quasigroup operations (in all three rows) can be performed in parallel.

This property can lead to hardware implementation of Edon–$\mathcal{R}(256, 384, 512)$ that can achieve even higher speeds.

5 Initial security analysis of the algorithm

This is just initial security analysis, and more thorough analysis will be done in the forthcoming period by the authors as well as by the cryptographic community.

The design of Edon–$\mathcal{R}(n)$ is based on Merkle-Damgård iterating principles [7–9]. In the light of latest attacks with multi-collisions, the design of Edon–$\mathcal{R}$ has incorporated the suggestions of Lucks [10] and Coron et al. [11]. Namely, by setting the internal memory of the iterated compression function to be double of the output length, weaknesses against generic attacks of Joux [12], and Kelsy and Schneier [13] are eliminated.

Doubling of the internal memory in our design is done by the fact that in every iterative step of its compression function, the strings of length $3n$ bits are mapped to strings of length $3n$ bits and then only the last significant $2n$ bits are kept for the next iterative step.

5.1 Infeasibility of going backward and infeasibility of finding free start collisions

According to the conjectured one-wayness of the function $\mathcal{R}_1$, iterating backward Edon–$\mathcal{R}(n)$ is infeasible. The conjecture is based on the infeasibility of solving nonlinear quasigroup equations in non-commutative and non-associative quasigroups. From this it follows that the workload for finding preimages and second-preimages for any hash function of the family Edon–$\mathcal{R}(n)$ is 2^n hash computations.

Moreover, inverting one-way function $\mathcal{R}_1$ would imply that finding free start collisions is feasible for the whole function Edon–$\mathcal{R}(n)$. Consequently, we base our conjecture that it is infeasible to find free start collisions for Edon–$\mathcal{R}(n)$ on the infeasibility of inverting the one-way function $\mathcal{R}_1$.

We will elaborate our claims more deeply by the following discussion:

Definition 12. *Let $(a_0, a_1, x_1), (b_0, b_1, x_2) \in Q_n \times Q_n \times Q_n$. If $\mathcal{R}_1(a_0, a_1, x_1) = (c_0, c_1, y)$ and $\mathcal{R}_1(b_0, b_1, x_2) = (d_0, d_1, y)$ then we say that the pair $((a_0, a_1, x_1), (b_0, b_1, x_2))$ is a free start collision for Edon–$\mathcal{R}(n)$.*

The free start collision situation is described in the Table 3.

	a_0	a_1	x_1
x_1	$x_0^{(1)}$	$x_1^{(1)}$	$x_2^{(1)}$
a_1	$x_0^{(2)}$	$x_1^{(2)}$	$x_2^{(2)}$
a_0	c_0	c_1	y

a.

	b_0	b_1	x_2
x_2	$y_0^{(1)}$	$y_1^{(1)}$	$y_2^{(1)}$
b_1	$y_0^{(2)}$	$y_1^{(2)}$	$y_2^{(2)}$
b_0	d_0	d_1	y

b.

Table 3. a. Schematic presentation of the function $\mathcal{R}_1(a_0, a_1, x_1) = (c_0, c_1, y)$, **b.** Schematic presentation of the function $\mathcal{R}_1(b_0, b_1, x_2) = (d_0, d_1, y)$.

In this moment we see two ways how to find free start collisions for Edon–$\mathcal{R}(n)$:

1. Generate a random $y \in Q_n$. Construct vectors (c_0, c_1, y) and (d_0, d_1, y) where $c_0, c_1, d_0, d_1 \in Q_n$ are randomly chosen. Try to find $\mathcal{R}_1^{-1}(c_0, c_1, y)$ and $\mathcal{R}_1^{-1}(d_0, d_1, y)$.
2. Generate a random (a_0, a_1, x_1) and compute $\mathcal{R}_1(a_0, a_1, x_1) = (c_0, c_1, y)$. Construct vector (d_0, d_1, y) where $d_0, d_1 \in Q_n$ are randomly chosen. Try to find $\mathcal{R}_1^{-1}(d_0, d_1, y)$.

Both ways need inversion of $\mathcal{R}_1$ and as we already said we see that as an infeasible task.

5.2 Testing avalanche properties of Edon–$\mathcal{R}(n)$

We have examined the avalanche propagation of the initial one bit differences of the compression function of Edon–$\mathcal{R}(n)$ during their evolution in all 9 quasigroup operations $*_n, (n = 256, 384, 512)$. We have used two experimental settings:

1. Examining the propagation of the initial 1–bit difference in a message consisting of all zeroes
2. Examining the propagation of the initial 1–bit difference in a randomly generated messages of n–bits.

The results for $n = 256$ are shown in the following Table 4. Notice that the level of Hamming distance equal to $\frac{1}{2}n = 128$ which would be expected in theoretical models of ideal random functions is achieved after applying quasigroup operations that lie on the down-right half of the tables (in bold).

The results for $n = 384$ are shown in the Table 5. Notice that the level of Hamming distance equal to $\frac{1}{2}n = 192$ which would be expected in theoretical models of ideal random functions is achieved after applying quasigroup operations that lie on the down-right half of the tables (in bold), but some close figures are obtained also after the second quasigroup operation (in italic).

The results for $n = 512$ are shown in the Table 6. Notice that the level of Hamming distance equal to $\frac{1}{2}n = 256$ which would be expected in theoretical models of ideal random functions is achieved after applying quasigroup operations that lie on the down-right half of the tables (in bold), but some close figures are obtained also after the second quasigroup operation (in italic).

One possible explanation about the reasons why Edon–$\mathcal{R}(384)$ and Edon–$\mathcal{R}(512)$ come slightly faster to the level of ideal random function than Edon–$\mathcal{R}(256)$ may lie in the fact that permutations π_2 and π_3 for $n = 384, 512$ are defined by bigger Latin squares of order 12×12 and 16×16 (see the Section 6). Thus they are more complex then corresponding permutations π_2 and π_3 for $n = 256$.

$Min = 15$ $Avr = 15$ $Max = 15$	$Min = 86$ $Avr = 108.44$ $Max = 133$	$Min = 107$ **Avr=127.43** $Max = 153$
$Min = 80$ $Avr = 110.84$ $Max = 142$	$Min = 103$ **Avr=128.17** $Max = 160$	$Min = 100$ **Avr=127.43** $Max = 151$
$Min = 103$ **Avr=127.54** $Max = 148$	$Min = 102$ **Avr=127.25** $Max = 146$	$Min = 105$ **Avr=127.86** $Max = 148$

a.

$Min = 15$ $Avr = 26.59$ $Max = 74$	$Min = 76$ $Avr = 113.68$ $Max = 149$	$Min = 102$ **Avr=128.11** $Max = 154$
$Min = 73$ $Avr = 115.93$ $Max = 155$	$Min = 103$ **Avr=128.09** $Max = 158$	$Min = 95$ **Avr=127.75** $Max = 155$
$Min = 101$ **Avr=128.07** $Max = 153$	$Min = 100$ **Avr=128.01** $Max = 154$	$Min = 95$ **Avr=127.67** $Max = 155$

b.

Table 4. a. Avalanche propagation of the Hamming distance between two 256–bit words M_1 and M_2 that initially differs in one bit and where $M_1 = 0$ (minimum, average and maximum) **b.** Avalanche propagation of the Hamming distance between two 256–bit words M_1 and M_2 that initially differs in one bit (minimum, average and maximum)

$Min = 23$ $Avr = 30.33$ $Max = 35$	$Min = 162$ *Avr=190.28* $Max = 255$	$Min = 166$ **Avr=190.89** $Max = 219$
$Min = 162$ *Avr=190.87* $Max = 218$	$Min = 166$ **Avr=192.17** $Max = 218$	$Min = 160$ **Avr=192.40** $Max = 222$
$Min = 162$ **Avr=191.40** $Max = 225$	$Min = 168$ **Avr=192.11** $Max = 223$	$Min = 160$ **Avr=192.15** $Max = 221$

a.

$Min = 23$ $Avr = 52.54$ $Max = 103$	$Min = 157$ *Avr=191.69* $Max = 227$	$Min = 163$ **Avr=192.31** $Max = 222$
$Min = 166$ *Avr=192.17* $Max = 225$	$Min = 164$ **Avr=191.41** $Max = 222$	$Min = 166$ **Avr=191.88** $Max = 222$
$Min = 166$ **Avr=192.68** $Max = 217$	$Min = 160$ **Avr=191.90** $Max = 216$	$Min = 167$ **Avr=191.99** $Max = 218$

b.

Table 5. a. Avalanche propagation of the Hamming distance between two 384–bit words M_1 and M_2 that initially differs in one bit and where $M_1 = 0$ (minimum, average and maximum) **b.** Avalanche propagation of the Hamming distance between two 384–bit words M_1 and M_2 that initially differs in one bit (minimum, average and maximum)

6 Design rationale

6.1 Choosing basic 32–bit operations

We have decided to choose 32–bit operations of addition modulo 2^{32}, XOR-ing and left rotations as an optimum choice that can be efficiently implemented both on low-end 8–bit and 16–bit processors, as well as on modern 32–bit and 64–bit CPUs. In the past several cryptographic primitives have been designed following the same rationale as well, such as: Salasa20 [14], The Tiny Encryption Algorithm [15], or IDEA [16] - to name a few.

6.2 Choosing permutations π_1, π_2 and π_3

Our goal was to design a structure that is non-commutative and non-associative quasigroup of huge orders ($2^{256}, 2^{384}$ and 2^{512}) in order to apply the principles of the hash family Edon–$\mathcal{R}$. We have found a way how to construct such a structure by applying some basic permutations π_1, π_2 and π_3 on the sets $\{0,1\}^{256}, \{0,1\}^{384}$ and $\{0,1\}^{512}$.

The permutations $\pi_{1,256}$, $\pi_{1,384}$ and $\pi_{1,512}$ are simple rotations on 256, 384 or 512–bit words. They can be effectively realized just by appropriate referencing of the 32–bit variables (after performing permutations π_2 and π_3). While the permutations π_2 and π_3 do the work of diffusion and nonlinear mixing separately on the first and the second argument of the quasigroup operations, after their outputs are XORed, the permutations π_1 introduce additional diffusion on the whole n–bit word. That diffusion

$Min = 27$ $Avr = 39.50$ $Max = 51$	$Min = 199$ *Avr=252.46* $Max = 289$	$Min = 222$ **Avr=256.031** $Max = 296$
$Min = 220$ *Avr=254.93* $Max = 293$	$Min = 222$ **Avr=255.25** $Max = 283$	$Min = 227$ **Avr=257.01** $Max = 288$
$Min = 224$ **Avr=256.36** $Max = 287$	$Min = 222$ **Avr=255.54** $Max = 290$	$Min = 227$ **Avr=255.89** $Max = 295$

a.

$Min = 27$ $Avr = 73.00$ $Max = 142$	$Min = 209$ *Avr=254.54* $Max = 288$	$Min = 222$ **Avr=255.34** $Max = 288$
$Min = 214$ *Avr=255.49* $Max = 287$	$Min = 226$ **Avr=255.85** $Max = 290$	$Min = 226$ **Avr=256.50** $Max = 287$
$Min = 217$ **Avr=255.35** $Max = 286$	$Min = 225$ **Avr=256.38** $Max = 288$	$Min = 221$ **Avr=256.402** $Max = 297$

b.

Table 6. a. Avalanche propagation of the Hamming distance between two 512–bit words M_1 and M_2 that initially differs in one bit and where $M_1 = 0$ (minimum, average and maximum) **b.** Avalanche propagation of the Hamming distance between two 512–bit words M_1 and M_2 that initially differs in one bit (minimum, average and maximum)

then have influence on the next application of the quasigroup operation $*_n$ (since we apply three such operations in every row).

For the choice of the permutations π_1 and π_2 we had plenty of possibilities. However, since our design is based on quasigroups, it was natural choice to use Latin squares in the construction of those permutations. Actually there is a long history of using Latin squares in the randomized experimental design (see for example [17]) as well as in cryptography [18–22].

Since for the permutations $\pi_{2,256}$ and $\pi_{3,256}$ we wanted bijectively to mix eight 32–bit variables we have used the following 8×8 Latin squares:

$$L_1 = \begin{pmatrix} 2\,1\,7\,6\,3\,4\,0\,5 \\ 4\,3\,2\,5\,0\,7\,1\,6 \\ 7\,0\,1\,4\,6\,2\,5\,3 \\ 6\,7\,0\,1\,4\,5\,3\,2 \\ 1\,4\,6\,3\,5\,0\,2\,7 \\ \hline 0\,6\,5\,2\,1\,3\,7\,4 \\ 5\,2\,3\,0\,7\,6\,4\,1 \\ 3\,5\,4\,7\,2\,1\,6\,0 \end{pmatrix} = \begin{pmatrix} L_{1,1} \\ L_{1,2} \end{pmatrix} \qquad L_2 = \begin{pmatrix} 5\,7\,0\,3\,4\,6\,1\,2 \\ 6\,2\,1\,0\,7\,3\,4\,5 \\ 7\,1\,3\,6\,5\,4\,2\,0 \\ 4\,6\,7\,5\,2\,0\,3\,1 \\ 1\,4\,6\,2\,3\,5\,0\,7 \\ \hline 2\,5\,4\,1\,0\,7\,6\,3 \\ 3\,0\,5\,4\,1\,2\,7\,6 \\ 0\,3\,2\,7\,6\,1\,5\,4 \end{pmatrix} = \begin{pmatrix} L_{2,1} \\ L_{2,2} \end{pmatrix}$$

Then we have split L_1 and L_2 on two (upper and lower) Latin rectangles $L_{1,1}$, $L_{1,2}$, $L_{2,1}$ and $L_{2,2}$. We used the columns of upper rectangles as index sets (block designs) for the variables that are bijectively transformed by addition modulo 2^{32} and the columns of lower rectangles as index sets (block designs) for the variables that are bijectively transformed by XORing of 32–bit variables. More concretely:

$$L_{1,1} \Rightarrow A_{1,1} = \begin{pmatrix} 0\,1\,1\,0\,1\,0\,1\,1 \\ 1\,1\,0\,1\,1\,0\,0\,1 \\ 1\,1\,1\,0\,0\,0\,1\,1 \\ 0\,1\,0\,1\,1\,1\,1\,0 \\ 1\,0\,0\,1\,1\,1\,1\,0 \\ 1\,0\,1\,0\,1\,1\,0\,1 \\ 1\,1\,1\,1\,0\,1\,0\,0 \\ 0\,0\,1\,1\,0\,1\,1\,1 \end{pmatrix} \qquad L_{1,2} \Rightarrow A_{1,2} = \begin{pmatrix} 1\,0\,0\,1\,0\,1\,0\,0 \\ 0\,0\,1\,0\,0\,1\,1\,0 \\ 0\,0\,0\,1\,1\,1\,0\,0 \\ 1\,0\,1\,0\,0\,0\,0\,1 \\ 0\,1\,1\,0\,0\,0\,0\,1 \\ 0\,1\,0\,1\,0\,0\,1\,0 \\ 0\,0\,0\,0\,1\,0\,1\,1 \\ 1\,1\,0\,0\,1\,0\,0\,0 \end{pmatrix}$$

As we mentioned in Section 2.1 matrix $A_{1,1}$ is nonsingular in $(\mathbb{Z}_{2^{32}}, +)$ and matrix $A_{1,2}$ is nonsingular in $GF(2)$. Similarly from Latin rectangles $L_{2,1}$ and $L_{2,2}$ we got the nonsingular matrices $A_{2,1}$ and $A_{2,2}$.

$$L_{2,1} \Rightarrow A_{2,1} = \begin{pmatrix} 0&1&0&0&1&1&1&1 \\ 0&1&1&0&1&0&1&1 \\ 1&1&0&1&0&0&1&1 \\ 1&0&1&1&0&1&1&0 \\ 0&0&1&1&1&1&0&1 \\ 1&0&0&1&1&1&1&0 \\ 1&1&1&1&1&0&0&0 \\ 1&1&1&0&0&1&0&1 \end{pmatrix} \qquad L_{2,2} \Rightarrow A_{2,2} = \begin{pmatrix} 1&0&1&1&0&0&0&0 \\ 1&0&0&1&0&1&0&0 \\ 0&0&1&0&1&1&0&0 \\ 0&1&0&0&1&0&0&1 \\ 1&1&0&0&0&0&1&0 \\ 0&1&1&0&0&0&0&1 \\ 0&0&0&0&0&1&1&1 \\ 0&0&0&1&1&0&1&0 \end{pmatrix}$$

Analogously, in what follows, without further explanation we will give corresponding Latin squares L_3 and L_4 of order 12×12 for Edon–$\mathcal{R}(384)$ and L_5 and L_6 of order 16×16 for Edon–$\mathcal{R}(512)$.

$$L_3 = \left(\begin{array}{cccccccccccc} 11&0&9&6&3&4&10&8&5&7&1&2 \\ 3&10&2&11&8&7&1&6&4&0&5&9 \\ 1&5&0&7&9&8&4&11&10&3&2&6 \\ 2&4&10&1&7&5&0&9&8&11&6&3 \\ 10&1&11&5&0&6&3&2&9&4&7&8 \\ 7&8&5&4&1&2&9&0&3&6&10&11 \\ 8&6&4&3&11&0&2&5&7&10&9&1 \\ \hline 0&9&6&10&5&3&7&1&2&8&11&4 \\ 6&2&3&8&10&1&5&4&11&9&0&7 \\ 4&11&7&2&6&9&8&10&0&1&3&5 \\ 5&7&1&9&4&10&11&3&6&2&8&0 \\ 9&3&8&0&2&11&6&7&1&5&4&10 \end{array}\right) \qquad L_4 = \left(\begin{array}{cccccccccccc} 11&10&9&5&7&0&4&8&1&6&2&3 \\ 4&7&0&8&11&2&10&9&6&5&3&1 \\ 9&1&3&2&4&5&6&0&8&10&7&11 \\ 5&11&1&9&6&10&8&3&7&0&4&2 \\ 6&8&2&7&3&1&11&4&0&9&5&10 \\ 7&3&10&4&1&9&0&2&11&8&6&5 \\ 10&2&7&0&9&6&1&11&5&3&8&4 \\ \hline 2&9&11&1&8&7&3&5&10&4&0&6 \\ 8&0&4&6&5&11&9&10&3&2&1&7 \\ 3&6&5&10&0&8&2&1&4&7&11&9 \\ 1&5&8&3&10&4&7&6&2&11&9&0 \\ 0&4&6&11&2&3&5&7&9&1&10&8 \end{array}\right)$$

$$L_5 = \left(\begin{array}{cccccccccccccccc} 4&10&11&1&2&5&7&3&13&0&8&14&9&12&6&15 \\ 0&15&1&10&8&7&13&12&9&3&14&11&6&5&2&4 \\ 15&3&6&4&1&9&10&14&0&2&11&12&13&7&5&8 \\ 6&1&3&11&0&2&14&8&5&9&15&13&7&4&10&12 \\ 10&4&0&6&9&8&12&13&1&5&2&15&3&11&14&7 \\ 13&0&14&3&4&10&9&11&15&8&1&5&12&6&7&2 \\ 2&13&7&8&11&12&5&9&3&15&6&10&14&0&4&1 \\ 3&6&10&15&13&4&11&0&2&1&12&7&5&9&8&14 \\ 9&8&12&14&7&1&0&5&4&6&13&3&2&15&11&10 \\ \hline 5&9&15&2&12&14&8&6&11&4&7&1&10&13&3&0 \\ 14&5&13&9&10&15&6&7&8&11&4&0&1&2&12&3 \\ 1&11&5&13&14&0&2&4&7&12&3&6&8&10&15&9 \\ 8&12&2&7&5&11&3&10&14&13&9&4&15&1&0&6 \\ 11&7&8&5&3&6&1&15&12&10&0&2&4&14&9&13 \\ 12&14&9&0&15&13&4&2&6&7&10&8&11&3&1&5 \\ 7&2&4&12&6&3&15&1&10&14&5&9&0&8&13&11 \end{array}\right) \qquad L_6 = \left(\begin{array}{cccccccccccccccc} 3&14&8&12&4&15&7&11&6&10&0&5&1&2&13&9 \\ 1&3&5&0&10&4&9&7&11&2&14&12&13&6&8&15 \\ 2&11&6&9&12&5&8&14&10&3&1&13&15&7&0&4 \\ 4&13&10&11&9&14&3&15&1&7&2&6&8&0&12&5 \\ 11&0&15&10&7&6&14&4&13&1&12&8&5&9&2&3 \\ 8&15&12&6&0&2&4&13&5&9&3&7&10&1&14&11 \\ 13&7&0&2&3&10&1&9&14&8&5&11&12&4&15&6 \\ 10&1&14&4&5&12&11&2&9&15&6&0&3&8&7&13 \\ 14&6&3&15&13&8&12&5&7&0&11&1&4&10&9&2 \\ \hline 7&9&11&3&1&13&2&6&15&4&8&14&0&12&5&10 \\ 12&10&7&5&2&3&13&8&0&11&9&4&14&15&6&1 \\ 9&5&13&8&11&7&6&0&4&12&15&10&2&3&1&14 \\ 0&4&2&14&15&1&5&12&8&6&10&3&9&13&11&7 \\ 5&8&4&1&6&9&0&10&2&13&7&15&11&14&3&12 \\ 6&12&9&13&14&0&15&1&3&5&4&2&7&11&10&8 \\ 15&2&1&7&8&11&10&3&12&14&13&9&6&5&4&0 \end{array}\right)$$

7 Conclusions

We have designed a concrete realization of the family of hash functions Edon–$\mathcal{R}$ with message digests of 256, 384 and 512 bits by defining huge non-commutative and non-associative quasigroups of orders 2^{256}, 2^{384} and 2^{512}. The definition of quasigroups involve 32–bit operations of addition modulo 2^{32}, bitwise XORing and left rotations. Those operations are very fast on most modern microprocessors but they can be also efficiently realized on low-end 8–bit and 16–bit processors. By our reference C code implementation on x86 platforms we have achieved processing speeds of 16 cycles/byte, 25.75 cycles/byte and 33.63 cycles/byte.

In the forthcoming period we will do additional security analysis and we will try to develop some optimized implementations for different platforms.

References

1. D. Gligoroski, S. Markovski and L. Kocarev, "Edon–$\mathcal{R}$, An Infinite Family of Cryptographic Hash Functions", Second NIST Cryptographic Hash Workshop, University of California - Santa Barbara, August, 2006 `http://www.csrc.nist.gov/pki/HashWorkshop/2006/Papers/GLIGOROSKI_EdonR-ver06.pdf`
2. D. Gligoroski, "On a Family of Minimal Candidate One-Way Functions and One-Way Permutations", in print, International Journal of Network Security, ISSN 1816-3548, (see also ePrint archive `http://eprint.iacr.org/2005/352.pdf` for an early version of the paper)
3. V.D. Belousov, *Osnovi teorii kvazigrup i lup.* (1967) "Nauka", Moskva.
4. S. Markovski, D. Gligoroski, V. Bakeva, "Quasigroup String Processing: Part 1", Contributions, Sec. Math. Tech. Sci., MANU **XX**, 1-2 (1999) 13–28
5. S. Markovski, D. Gligoroski, V. Bakeva, "Quasigroup and Hash Functions", Disc. Math. and Appl, Sl.Shtrakov and K. Denecke ed., Proceedings of the 6th ICDMA, Bansko 2001, pp. 43–50
6. B.D. McKay, E. Rogoyski, "Latin squares of order 10", Electronic J. Comb. **2** (1995) `http://ejc.math.gatech.edu:8080/Journal/journalhome.html`
7. I. B. Damgård, "Collision free hash functions and public key signature schemes", Advances in CryptologyEUROCRYPT 87 (LNCS 304), 1988 203216.
8. I. B. Damgård, "A design principle for hash functions", Advances in CryptologyCRYPTO 89 LNCS 435, G. Brassard, Ed., Springer-Verlag, 1990, pp. 416-427.
9. R. Merkle, "One way hash functions and DES," Advances in Cryptology-Crypto'89, LNCS 435, G. Brassard, Ed., Springer-Verlag, 1990, pp. 428–446.
10. S. Lucks, "Design Principles for Iterated Hash Functions", Cryptology ePrint Archive, report 2004/ 253.
11. J.-S. Coron, Y. Dodis, C. Malinaud, and P. Puniya, "Merkle-Damgård revisisted: How to construct a hash function", In V. Shoup, editor, Advances in Cryptology CRYPTO 2005, LNCS 3621, Springer-Verlag, Berlin, Germany, 2005.
12. A. Joux, "Multicollisions in iterated hash functions. Application to cascaded constructions", In M. Franklin, editor, Advances in Cryptology CRYPTO 2004, LNCS 3152, Springer-Verlag, Berlin, Germany, 2004, pp. 306-316.
13. J. Kelsey and B. Schneier, "Second preimages on n-bit hash functions for much less than 2^n work" In R. Cramer, editor, Advances in Cryptology EUROCRYPT 2005, LNCS 3494, Springer-Verlag, Berlin, Germany, 2005, pp. 474-490.
14. D. Bernstein, "Salsa20", eSTREAM – ECRYPT Stream Cipher Project, Report 2005/025, http://www.ecrypt.eu.org/stream
15. D. J. Wheeler, R. M. Needham, "TEA, a tiny encryption algorithm", Fast software encryption: second international workshop, Leuven, Belgium, December 1994, Proceedings, LNCS 1008, Springer-Verlag, Berlin, (1995), 363–366.
16. X. Lai, J. L. Massey, S. Murphy, "Markov ciphers and differential cryptanalysis", in Advances in cryptology – EUROCRYPT '91: Proceedings of the workshop on the theory and application of cryptographic techniques, Brighton, April, 1991, LNCS 547, Springer-Verlag, Berlin, (1991), 17–38.
17. DesignTheory.org, http://designtheory.org/
18. C. E. Shannon, "Communication theory of secrecy systems", Bell Sys. Tech. J. 28 (1949), pp. 657–715
19. C. P. Schnorr and S. Vaudenay, "Black Box Cryptanalysis of hash networks based on multipermutations", In Advances of Cryptology - EUROCRYPT'94, Springer, Berlin 1995.

20. G. Carter, E. Dawson, and L. Nielsen, “A latin square version of DES”, In Proc. Workshop of Selected Areas in Cryptography, Ottawa, Canada, 1995.
21. J. Cooper, D. Donovan and J. Seberry, “Secret sharing schemes arising from Latin Squares”, Bull. Inst. Combin. Appl., Vol 4, 1994, pp. 33–43
22. J. Dènes and A. D. Keedwell, “A new authentication scheme based on latin squares”, Discrete Math., Vol. 106-107 (1992) pp. 157–161