Edon–$\mathcal{R}$, An Infinite Family of Cryptographic Hash Functions

D. Gligoroski[1,*] and S. Markovski[2] and L. Kocarev[3]

[1] Centre for Quantifiable Quality of Service in Communication Systems, Norwegian University of Science and Technology, O.S.Bragstads plass 2E, N-7491 Trondheim, NORWAY, e-mail: gligoroski@yahoo.com
[2] University - "Ss Cyril and Methodius", Faculty of Natural Sciences and Mathematics, Institute of Informatics, P.O.Box 162, 1000 Skopje, Republic of MACEDONIA, e-mail: smile@ii.edu.mk
[3] University of California San Diego, Institute for Nonlinear Science, 9500 Gilman Drive, La Jolla, CA 92093-0402, USA, e-mail: lkocarev@ucsd.edu

Abstract

We propose a new infinite family of cryptographic hash functions, Edon–$\mathcal{R}$, based on a recently defined candidate one-way function. Edon–$\mathcal{R}$ is a class of hash functions with variable output lengths. It is defined using quasigroups and quasigroup string transformations.

Key words: hash function, one-way function, quasigroup

1 Introduction

Cryptographic hash functions are used in a wide range of applications including message integrity, authentication, digital signature and public key encryption. A cryptographic hash function takes an input of arbitrary size and produces an output, also called the hash value, of a fixed, predetermined size. The important properties of a cryptographic hash function are the collision-resistance and the preimage-resistance. Most hash functions are iterative and are built using the Merkle-Damgård design of hash functions [20, 5]. Although the mathematical knowledge about practical construction of cryptographic hash functions was increasing with slow rate during the last 15–20 years, we can locate at least four breakthroughs in their theoretical understanding. Those are given in the following list:

1. The first result from 1989 is that of Merkle [20] and Damgård [5] where they proved that for the iterated hash function h to be collision resistant it is sufficient that its compression function f is a collision resistant function.

2. The second theoretical result from 1992 is due to Lai and Massey in [14] where they proved that finding second preimages of some iterated hash function h (having a compression function f in its design) and by fixed initial IV in less than 2^n operations is equivalent of finding second preimages of the compression function f with arbitrary IV in less than 2^n operations.

3. The third important theoretical result is that of Joux [12] in 2004 were he showed that the workload for finding second-preimage collisions with equal length for iterated one-way hash functions is about $\log(k) \times 2^{n/2}$, where k is the number of computed hash values.

4. The fourth theoretical result from 2005 is that of Kelsy and Schneier in [13] where they showed that the workload for finding second-preimages that are expandable messages with different length is about $k \times 2^{n/2+1}$.

Concerning the Merkle-Damgård design, Coron at al. in [3] and Lucks in [15] made several suggestions how to strengthen that design while Gauravaram, Millan and Neito in a recent ePrint paper [6] gave an interesting discussion on the possibilities that Merkle-Damgård design for MDx family was, in fact, not properly implemented.

Although general in their approach, the above mentioned breakthroughs in the analysis of iterated hash functions are connected with the latest successes in the cryptanalysis of MD4 family of hash functions [28]. This family of hash functions was being in use around 15 years and it is one of the most used

*Corresponding author

cryptographic primitives. Recently, Wang at al. [32] pointed out some weaknesses of SHA-1 hash function. Many researchers immediately proposed SHA-1 to be replaced by SHA-256 (which outputs 256 bits instead of 160 bits of SHA-1). Other candidates from SHA family include SHA-n, where $n \in 224, 256, 384, 512$. We can mention here several other members of MDx family: RIPEMD-160 [26], as well as RIPEMD-256 and RIPEMD-320 [27].

To have in a reserve hash functions with bigger output was a strategy that was adopted from several international standardization organizations (including NIST and ISO). However, the costs of transition from one function to another are enormous, and the time for introducing a new hash function is estimated from 5 to 10 years.

In this paper we support the concept of designing cryptographic hash functions with variable output lengths. One such hash function with variable hash lengths, but restricted to 128, 160, 192, 224 or 256 output bits is HAVAL, proposed in 1993 by Zheng, Pieprzyk and Seberry [33]. However, the principles of the design of HAVAL are similar to those of MDx family of hash functions and collisions were found for 128 bit version of the function by Wang at al. in [31].

Another issue that have to be addressed about cryptographic hash function is their design in the light of ever increasing computing power. Attacks with complexity 2^{64} are already accessible using distributed internet computing, while attacks with complexity 2^{80} may also be feasible after 5–10 years. Therefore, hash functions with output size less than 160 bits are not a good choice for long term security. The ever-growing computing power is forcing designers to redesign secure hash functions, which is not at all trivial designing process. Instead of redesigning and creating new secure hash functions with larger (but still fix) digest size, it would be much simple to design one or several hash functions with variable length of output. As an additional argument for our standing we can mention the analogy with RSA algorithm [29] and its usage during the last 20 years. As the computing power was increasing, the length of prime numbers used in RSA was increasing, but the basic algorithm is still unchanged.

The design of a cryptographically secure hash function is a sufficient motivation and challenging problem by itself, but we had additional motive in the light of the last visions and revisions on Merkle-Damgård design. Actually, the number of cryptographically secure hash functions is not very large – see for example the excellent review of Preneel [24] or visit the web page [25] where you can find a good and updated review of almost all known secure hash functions.

From the point of view of used mathematical techniques, our design can be treated as one of those designs which does not rely on ad-hock techniques using XOR-ing and rotating, but tries to relate the claims of their security on some mathematically hard problem. Our design is based on theory of quasigroups and quasigroup string transformations [16, 9] and recently introduced one-way candidate functions [8]. The main point is the hardness of solving nonlinear systems of quasigroup equations, since quasigroups are algebraic structures with one binary operation, which do not satisfy the usual algebraic laws used in solving equations (the commutative law, the associative law, the idempotent law, having zeros or units, and so on).

A similar approach have been taken by Damgård in 1988 [4] and Gibson in 1991 [7], who designed hash functions based on intractability of discrete log problem, as well as recently by Contini, Lenstra and Steinfeld in 2005 [2], who proposed a hash function which cryptographic strength relies on the hardness of the number factorization problem.

The organization of the paper is as follows: In Section 2 we give some basic mathematical definitions and a definition of a one-way function, in Section 3 we define the hash function Edon–$\mathcal{R}$, in Section 4 we analyze the cryptographic properties of the proposed hash function, and we conclude the paper by Section 5.

2 Definition of a one-way function $\mathcal{R}_1 : Q^r \to Q^r$

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 [8] as well as the parts of the conjectured computational difficulties for their inversion. 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 $* : Q^2 \to Q$ with the property each of the equations $a * x = b$ and $y * a = b$ to have unique solutions x and y in Q. We deal with finite quasigroups only. 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 [1, 17, 18, 19].

In our design we are using finite quasigroups $(Q, *)$ of order $|Q| = n$, where $n \geq 2$ and $n = 2^w$. That means that the hash functions use $|Q|^2 = n^2$, w-bit words of internal memory for storing the quasigroup. For example if we take $w = 8$ then $n = 2^w = 256$ and we will work with a set of 256 8-bit words (bytes), i.e., we will work with the set $Q = \{0, 1, \ldots, 255\}$ which in fact is representing the ASCII set. In such a way, we will need $256^2 = 65536$ bytes, i.e., 65Kb of internal memory for storing the quasigroup, which is not small amount if we want to implement it in some embedded microprocessor. However, the design allows us to use quasigroups of any order which is a power of 2, i.e. 8, 16, 32, 64, ..., but the corresponding operations inside the hash functions will not be on bytes (8-bit words) but on different w-bit words. Then the needed internal memory for storing the quasigroup would be 24 bytes, 128 bytes, 640 bytes, 3 Kb, and so on.

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

Definition 1 (**[8] 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

$$Q_m(a_0, a_1, \ldots, a_{r-1}) = (b_0, b_1, \ldots, b_{r-1})$$

$$\iff b_i := \begin{cases} m * a_0, & i = 0 \\ b_{i-1} * a_i, & 1 \leq i \leq r-1. \end{cases}$$

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

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

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

$$\underbrace{x * (\cdots * (x}_{k} * y)) = y, \; y = ((y * \underbrace{x) * \ldots) * x}_{k}. \quad (1)$$

Shapeless quasigroups can be effectively constructed by using, for example, the Hall's algorithm [11] for construction of systems of different representatives of a family of finite sets. The reason why it is possible to construct shapeless quasigroups is the huge number of quasigroups (for example, there are around 2^{430} quasigroups of order 16 and much more than 2^{192672} quasigroups of order 256). Basically, our claims that the family of hash functions Edon–$\mathcal{R}$ that we will define in the next section is infinite relies upon the fact that the number of finite quasigroups (and shapeless quasigroups) is infinite and it grows with double exponential rate with the order of the quasigroup (see for example [19]).

We want to note that linear quasigroups that are mentioned in [8] and that are not suitable for design of quasigroup one-way functions $\mathcal{R}_1$ actually does not satisfy the criterion of being shapeless, concretely the criterion described by the equation (1).

Our statement that $\mathcal{R}_1$ is one-way function, i.e., for a given string $A \in Q^r$ it is easy to compute $\mathcal{R}_1(A)$ and it is computationally infeasible, for a given $B \in Q^r$, to find an $A \in Q^r$ such that $B = \mathcal{R}_1(A)$, is based on the following hypothesis:

Hypothesis *There is no effective algorithm for solving a system of non-linear quasigroup equations in a shapeless quasigroup.*

In this moment there is not enough mathematical knowledge for solving systems of quasigroup equations in order to prove or disprove the above hypothesis. However, our quasigroup string transformations can be seen as a special type of cellular automata operations. The predictability of cellular automata was investigated by Moore at al. in [21, 22] in cases when the obtained quasigroups have some of the properties that shapeless quasigroups don't. Moreover Goldmann and Russell [10] have shown that solving system of equations in non-abelian groups is NP-complete and Moore, Tesson and Thérien in [23] have shown NP-completeness for even more general algebraic structures, i.e., monoids that are not product of Abelian group and commutative idempotent monoid. Based on all this discussion we state the following conjecture:

Conjecture 1 *In a shapeless quasigroup $(Q, *)$, the function $\mathcal{R}_1 : Q^r \to Q^r$ is one-way function.*

From the previous discussion we can say that having a shapeless quasigroup, i.e., a quasigroup that does not have any mathematical property that will help us to reduce the complexity of the equations that have to be solved, it seems that only method for solving systems of quasigroup equations is by combinatorial exhaustive search and reading from the lookup table that defines the quasigroup. Next, we give arguments that support the Conjecture 1 from the lookup table point of view, by the following Theorem (similar but slightly more concise then it is in [8]):

Theorem 1 *If the quasigroup $(Q,*)$ of order n is shapeless, then the number of computations based only on the lookup table that defines the quasigroup $(Q,*)$ for finding a preimage of the function $\mathcal{R}_1 : Q^r \to Q^r$ is $O(n^{\lfloor \frac{r}{3} \rfloor})$.*

Proof It is clear that $\mathcal{R}_1(A)$ can be effectively computed for any string $A = (a_0, a_1, \ldots, a_{r-1}) \in Q^r$ in $O(r^2)$ operations. For the inverse task, given a string $B = (b_0, b_1, \ldots, b_{r-1}) \in Q^r$, we have to find a string $A = (x_0, x_1, \ldots, x_{r-1})$ such that

$$B = \mathcal{R}_1(A) = Q_{x_0}(Q_{x_1} \ldots (Q_{x_{r-1}}(x_0, x_1, \ldots, x_{r-1})))).$$

Further on, let denote $Q_{x_{r-1}}(x_0, x_1, \ldots, x_{r-1}) = (x_0^{(1)}, x_1^{(1)}, \ldots, x_{r-1}^{(1)})$ and

$$Q_{x_{r-i}}(x_0^{(i-1)}, x_1^{(i-1)}, \ldots, x_{r-1}^{(i-1)}) = (x_0^{(i)}, x_1^{(i)}, \ldots, x_{r-1}^{(i)})$$

for $i = 2, 3, \ldots r$. Then

$$B = Q_{x_0}(x_0^{(r-1)}, \ldots, x_{r-1}^{(r-1)}) = (x_0^{(r)}, \ldots, x_{r-1}^{(r)}),$$

i.e., $b_j = x_j^{(r)}$ for $j = 0, 1, \ldots, r-1$. In order for clearer presentation of these computations, we use the Table 1a, where $x_0^{(1)} = x_{r-1} * x_0$, $x_0^{(i+1)} = x_{r-i-1} * x_0^{(i)}$ for $i = 2, \ldots, r$, $x_j^{(1)} = x_{j-1}^{(1)} * x_j$ for $j = 1, \ldots, r-1$, $x_{j+1}^{(i)} = x_j^{(i)} * x_{j+1}^{(i-1)}$ for $i = 2, \ldots, r$, $j = 0, \ldots, r-2$. It can be seen from Table 1a and from the definition of the transformations Q_m that we can find the solutions of the equations $b_i * x_{i+1}^{(r-1)} = b_{i+1}$ in indeterminate $x_{i+1}^{(r-1)}$, $i = 0, \ldots, r-2$, and let denote them by $b_{i+1}^{(r-1)}$. After that we can find the solutions $b_{i+1}^{(r-2)}$ of the equations $b_i^{(r-1)} * x_{i+1}^{(r-2)} = b_{i+1}^{(r-1)}$ in indeterminate $x_{i+1}^{(r-2)}$ for $i = 1, \ldots, r-2$, and so on. After $r-2$ steps we have the solution $b_{r-1}^{(1)}$ of the equation $b_{r-2}^{(2)} * x_{r-1}^{(1)} = b_{r-1}^{(2)}$. Then Table 1a transforms to Table 1b. We can see that no value for the indeterminate $x_0, \ldots, x_{r-1}$ can be obtained.

	x_0	x_1	x_2	$\ldots$	x_{r-2}	x_{r-1}
x_{r-1}	$x_0^{(1)}$	$x_1^{(1)}$	$x_2^{(1)}$	$\ldots$	$x_{r-2}^{(1)}$	$x_{r-1}^{(1)}$
x_{r-2}	$x_0^{(2)}$	$x_1^{(2)}$	$x_2^{(2)}$	$\ldots$	$x_{r-2}^{(2)}$	$x_{r-1}^{(2)}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
x_2	$x_0^{(r-2)}$	$x_1^{(r-2)}$	$x_2^{(r-2)}$	$\ldots$	$x_{r-2}^{(r-2)}$	$x_{r-1}^{(r-2)}$
x_1	$x_0^{(r-1)}$	$x_1^{(r-1)}$	$x_2^{(r-1)}$	$\ldots$	$x_{r-2}^{(r-1)}$	$x_{r-1}^{(r-1)}$
x_0	b_0	b_1	b_2	$\ldots$	b_{r-2}	b_{r-1}

Table 1a.

Let choose a value a_0 for x_0. Then, as we explained above, we can determine from Table 1b the values for $x_0^{(r-1)}$, $x_1^{(r-2)}$, $\ldots$, $x_{r-2}^{(1)}$ and x_{r-1} and let denote them as $a_0^{(r-1)}$, $a_1^{(r-2)}$, $\ldots$, $a_{r-2}^{(1)}$ and a_{r-1}. Then we can also compute the value $x_0^{(1)} = a_{r-1} * a_0$ and let denote it by $c_0^{(1)}$. After that we can choose a value a_1 for x_1 and we can transform the Table 1b with new known values as it is shown in Table 1c.

After $s = \lfloor \frac{r}{3} \rfloor$ chooses of the values for the indeterminate $x_0, x_1, \ldots, x_s$ we have to check if several equalities of kind $c_s^{(s)} * c_{s+1}^{(s-1)} = a_{s+1}^{(s)}$, $c_{s-1}^{(s+1)} * c_s^{(s)} = a_s^{(s+1)}$, $c_{s+1}^{(s-1)} * c_{s+2}^{(s-2)} = a_{s+2}^{(s-1)}, \ldots$ are satisfied. It follows that we have to make $|Q|^{\lfloor r/3 \rfloor} = n^{\lfloor r/3 \rfloor}$ guesses for exact completing of the Table 1a, i.e, for finding a pre-image $A = (a_0, a_1, \ldots, a_{r-1})$ of the given string $B = (b_0, b_1, \ldots, b_{r-1})$ such that $B = \mathcal{R}(A)$.

	x_0	x_1	x_2	$\ldots$	x_{r-2}	x_{r-1}
x_{r-1}	$x_0^{(1)}$	$x_1^{(1)}$	$x_2^{(1)}$	$\ldots$	$x_{r-2}^{(1)}$	$b_{r-1}^{(1)}$
x_{r-2}	$x_0^{(2)}$	$x_1^{(2)}$	$x_2^{(2)}$	$\ldots$	$b_{r-2}^{(2)}$	$b_{r-1}^{(2)}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
x_2	$x_0^{(r-2)}$	$x_1^{(r-2)}$	$b_2^{(r-2)}$	$\ldots$	$b_{r-2}^{(r-2)}$	$b_{r-1}^{(r-2)}$
x_1	$x_0^{(r-1)}$	$b_1^{(r-1)}$	$b_2^{(r-1)}$	$\ldots$	$b_{r-2}^{(r-1)}$	$b_{r-1}^{(r-1)}$
x_0	b_0	b_1	b_2	$\ldots$	b_{r-2}	b_{r-1}

Table 1b.

	a_0	a_1	a_2	$\ldots$	x_{r-4}	x_{r-3}	a_{r-2}	a_{r-1}
a_{r-1}	$c_0^{(1)}$	$c_1^{(1)}$	$x_2^{(1)}$	$\ldots$	$x_{r-4}^{(1)}$	$a_{r-3}^{(1)}$	$a_{r-2}^{(1)}$	$b_{r-1}^{(1)}$
a_{r-2}	$c_0^{(2)}$	$c_1^{(2)}$	$x_2^{(2)}$	$\ldots$	$a_{r-4}^{(2)}$	$a_{r-3}^{(2)}$	$b_{r-2}^{(2)}$	$b_{r-1}^{(2)}$
x_{r-3}	$x_0^{(3)}$	$x_1^{(3)}$	$x_2^{(3)}$	$\ldots$	$a_{r-4}^{(3)}$	$b_{r-3}^{(3)}$	$b_{r-2}^{(3)}$	$b_{r-1}^{(3)}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
x_3	$x_0^{(r-3)}$	$a_1^{(r-3)}$	$a_2^{(r-3)}$	$\ldots$	$b_{r-4}^{(r-3)}$	$b_{r-3}^{(r-3)}$	$b_{r-2}^{(r-3)}$	$b_{r-1}^{(r-3)}$
x_2	$a_0^{(r-2)}$	$a_1^{(r-2)}$	$b_2^{(r-2)}$	$\ldots$	$b_{r-4}^{(r-2)}$	$b_{r-3}^{(r-2)}$	$b_{r-2}^{(r-2)}$	$b_{r-1}^{(r-2)}$
a_1	$a_0^{(r-1)}$	$b_1^{(r-1)}$	$b_2^{(r-1)}$	$\ldots$	$b_{r-4}^{(r-1)}$	$b_{r-3}^{(r-1)}$	$b_{r-2}^{(r-1)}$	$b_{r-1}^{(r-1)}$
a_0	b_0	b_1	b_2	$\ldots$	b_{r-3}	b_{r-4}	b_{r-2}	b_{r-1}

Table 1c.

□

We can derive r quasigroup equations from Table 1a with indeterminate $x_0, x_1, \ldots, x_{r-1}$, but the form of those equations is quite complicated. Example 1

shows how the system of equations looks like for a simple case when $r = 3$.

Example 1 Let $r = 3$. Then at first we have the following equalities from Table 1a:
$x_0^{(1)} = x_2 * x_0$; $x_1^{(1)} = (x_2 * x_0) * x_1$; $x_2^{(1)} = ((x_2 * x_0) * x_1) * x_2$; $x_0^{(2)} = x_1 * (x_2 * x_0)$; $x_1^{(2)} = (x_1 * (x_2 * x_0)) * ((x_2 * x_0) * x_1)$; $x_2^{(2)} = \big((x_1*(x_2*x_0))*((x_2*x_0)*x_1)\big)*\big(((x_2*x_0)*x_1)*x_2\big)$.

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

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

Since the above system is enough simple, one can show that for any given $x_0 \in Q$ there are uniquely determined values of x_1 and x_2. Thus, the system has $|Q| = n = n^{\lfloor 3/3 \rfloor}$ different solutions (x_0, x_1, x_2). After that we have to check which one of the solutions is a preimage of (b_0, b_1, b_2).

We notice here that the expressions $x_i^{(j)}$ are quite complicated when $i \approx j$, and in this case they contain exponential number of terms. Hence, solving a system of equations for not so large values of r is not effective.

3 Edon–$\mathcal{R}$ hash algorithm

Having one-way quasigroup function $\mathcal{R}_1$, we now define a hash algorithm named "Edon–$\mathcal{R}$" that map a message M of arbitrary length of $l \leq 2^{64}$ words (where by word we mean a w-bit word) into a hash value of N words. The number N is the input parameter as it is the message M as well.

The definition of Edon–$\mathcal{R}$ hash function includes one string of length $2N$ and one string of length N, where N is the desired hash digest size in w-bit words. Those strings are the following:

1. String $H = (h_0, h_1, \dots \dots, h_{2N-1})$ of length $2N$ that holds intermediate values of hashing H_i. The initial value H_0 is

   $$H_0 = (h_0, h_1, \dots, h_{2N-1}) = (0 \bmod 2^w, 1 \bmod 2^w, \dots, 2N-1 \bmod 2^w).$$

2. The string $M_i = (m_0, m_1, \dots, m_{N-1})$ which holds the i-th part of the padded message M'. More concretely, having a message $M = b_1||b_2||\dots||b_j||\dots||b_l$ that has length of l w-bit words, (we use the symbol $||$ for operation of concatenation) by padding the message we will produce a new message $M' = b_1||b_2||\dots||b_j||\dots||b_l||l_1||l_2||\dots||l_t||b_1||\dots||b_j$, where $l_1||l_2||\dots||l_t$ is the w-bit conjugate of the number l (l is considered as 64 bit number, $t = \lfloor \frac{64}{w} \rfloor + 1$), and $j \geq 0$ is the smallest non-negative integer such that $l + t + j \equiv 0 \bmod N$. Then, we will denote the parts of N consecutive w-bit words as

   $$M' \equiv M_1||M_2||\dots||M_k$$

   where $k = \frac{l+t+j}{N}$, and the length of every M_i is N w-bit words.

Note that our padding is very similar with popular Merkle-Damgård strengthening, but still it differs from it by the usage of message bits from its beginning.

Edon–$\mathcal{R}$ algorithm

Input: $(Q, *)$, N and M, where:
$(Q, *)$ is a shapeless quasigroup of order $2^w, w \geq 4$,
the number N is such that the length of the hash output is $w \times N$ bits and
M is the message to be hashed.

Output: A hash of length $w \times N$ bits.

1. **Pad** the message M, so the length of the padded message M' is multiple of N w-bit words i.e. $|M'| = k \times N$.
2. **Initialize** $H_0 = (0 \bmod 2^w, 1 \bmod 2^w, \dots, 2N - 1 \bmod 2^w)$.
3. **Compute** the hash with the following iterative procedure:

   For $i = 1$ to k do
   $H_i = \mathcal{R}_1(H_{i-1}||M_i) \bmod 2^{2wN}$;

Output:

$$\textit{Edon-}\mathcal{R}(M) = H_k \bmod 2^{wN}$$

The one-way function $\mathcal{R}_1$ in this concrete realization is considered as transformation $Q^{3N} \to Q^{3N}$ (i.e., we take $r = 3N$) and then, for obtaining the intermediate value H_i, we just apply the operation $\bmod 2^{2wN}$ that takes the last $2N$ w-bit words from the result of $\mathcal{R}_1$. Finally, since the requested output from the hash function is N w-bit words, we take just the last N w-bit words from the H_k and that is denoted as the operation $\bmod 2^{wN}$.

4 Analysis of the algorithm

In this section and in the following subsections we will discuss Edon–$\mathcal{R}$ security properties such as being collision resistant, being preimage and second-preimage resistant hash function.

4.1 Collision resistance of Edon–$\mathcal{R}$

A cryptographic hash function that produces n-bit outputs is collision resistant if the birthday attack is the most effective attack for finding collisions. By the properties of the birthday attack (see for example [30] Chapter 18) if the hash output consists of n bits, then finding a collision should happen after examining approximately $2^{n/2}$ input messages.

Showing the collision resistance property of a family of hash functions is much harder task than showing the collision resistance of a specifically defined hash function. That is because of the additional freedom of choice that attacker has for choosing a function from a family of hash functions. In the case of Edon–$\mathcal{R}$ that freedom of choice consists of choosing different shapeless quasigroups. It follows from the definition of Edon–$\mathcal{R}$ that in every iterative step of its compression function $\mathcal{R}_1$, strings of length $3N$ are mapped to strings of length $3N$. Additionally, it follows from Theorem 1 that the minimum number of referencing the lookup table of the shapeless quasigroup, no matter what are the relations in the used quasigroup, is $2^{\frac{3N}{3}} = 2^N$ operations.

From **Conjecture 1** we can state that the security of Edon–$\mathcal{R}$ is based on the computational infeasibility to find a solution of systems of quasigroup equations. Of course, if the quasigroup have some structural property, such as being a group, loop or having unit elements, then instead of having N equations to solve, we could define additional equations, reducing the possibilities for choosing elements and so reducing the number of guesses for finding collisions.

Basically, those are the arguments for our claims that every hash function from the family Edon–$\mathcal{R}$ is collision resistant.

4.2 Preimage resistance of Edon–$\mathcal{R}$

In this subsection we will discuss Edon–$\mathcal{R}$ property to be preimage and second-preimage resistant. Being preimage resistant means that it is computationally infeasibile to find the original message X if its hash $Y = \textit{Edon-}\mathcal{R}(X)$ is given. On the other hand, being second-preimage resistant means that it is computationally infeasible to find second message X' such that $Y = \textit{Edon-}\mathcal{R}(X) = \textit{Edon-}\mathcal{R}(X')$

n	$T = 2^{\frac{n}{2}}$	Edon-$\mathcal{R}$
24	$2^{12} = 4,096$	6,258
32	$2^{16} = 65,536$	50,681
40	$2^{20} = 1,048,576$	1,023,436
48	$2^{24} = 16,777,216$	19,280,524

Table 2. Experimental testing of the Birthday attack on Edon-$\mathcal{R}$ with differen number of n bits of the hash output. The column $T = 2^{\frac{n}{2}}$ is the theoretically expected number of messages before finding a collision and the column Edon-$\mathcal{R}$ shows the actual number of random messages that we examined before finding a collision.

The design of Edon–$\mathcal{R}$ is based on Merkle-Damgård iterating principles. In the light of latest attacks with multi-collisions, we adopted the suggestions of Lucks [15] and Coron at al. [3]. Namely, by setting the internal memory of the iterated compression function to be double of the output length, we eliminate the possible weaknesses against generic attacks of Joux, Kelsy and Schneier.

n	Edon-$\mathcal{R}$
40	$\bar{H} = 20.00, \sigma = 3.16$
64	$\bar{H} = 32.00, \sigma = 4.00$
80	$\bar{H} = 40.00, \sigma = 4.47$
128	$\bar{H} = 64.00, \sigma = 5.66$
160	$\bar{H} = 79.89, \sigma = 7.03$
256	$\bar{H} = 128.00, \sigma = 8.01$
320	$\bar{H} = 159.39, \sigma = 13.27$
384	$\bar{H} = 191.43, \sigma = 14.34$
512	$\bar{H} = 255.97, \sigma = 11.31$
1024	$\bar{H} = 512.00, \sigma = 16.01$

Table 3. Experimental examination of the avalanche criterion. The first column labelled as n is the number of bits of the hash output. The column Edon-$\mathcal{R}$ has the calculated average Hamming distance $\bar{H}$ and standard deviation σ.

The 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 $3N$ are mapped to strings of length $3N$ and then only the last significant $2N$ letters are kept for the next iterative step. Thus, by similar discussion as in the previous subsection (on the infeasibility of solving nonlinear quasigroup equations in shapeless quasigroups) we can claim that the workload for finding preimages

and second-preimages for any hash function of the family Edon–$\mathcal{R}$ is 2^N hash computations.

4.3 Some initial experimental tests of Edon–$\mathcal{R}$

The Edon–$\mathcal{R}$'s property to give hash outputs with variable lengths can be used to test Edon–$\mathcal{R}$ algorithms for collisions and resistance on "Birthday attack". With the C reference implementation of Edon-$\mathcal{R}$ we have chosen the number of output bits to be relatively small (24, 32, 40 and 48 bits), since for those cases we could examine all possibilities with modern PC computer in reasonable time. The number of examined input messages before finding a collision is shown in Table 2 and it is in compliance with the "Birthday attack" level of $2^{\frac{n}{2}}$, where n is the number of bits in the hash output.

We have also examined the avalanche property of Edon–$\mathcal{R}$ family of hash functions. Namely, we have made experiments measuring Hamming distance of the hash outputs when original messages differ in 1 bit. For that purpose for every n bits ($n \in \{40, 64, 80, 128, 160, 256, 320, 384, 512, 1024\}$) we generated 1,000,000 pairs of messages (M_1, M_2) with random length from 100 to 1100 bytes and the difference between M_1 and M_2 was 1 bit (the position of that one bit was also randomly chosen). The summary of the experiments is presented in Table 3. As it is shown on Table 3, Edon–$\mathcal{R}$ hash functions posses expected avalanche property, i.e., when two messages differ only in one bit, the expected Hamming distance between their n-bit hashes should be approximately $\frac{n}{2}$.

4.4 Speed analysis of the algorithm

If we look at the definition of Edon–$\mathcal{R}$, we will notice that the operations are mostly operations to determine memory location and the value in that location, (i.e., references to one dimensional or two dimensional arrays). So, from the perspective of the design of other hash functions (that are using mostly bit operations in registers, but also have references to one dimensional arrays) we can say that Edon–$\mathcal{R}$ is using slower approach, because operations with memory are several times slower then operations with registers. Nevertheless, we think that the properties to have variable lengths of hash output, to have stronger mathematical indications about function's strong collision resistance, as well as the fact that we have infinite number of cryptographically strong hash functions, is worthwhile. Moreover, modern microprocessors have enough cash memory of Level 1, working with the speed of the microprocessor, for storing the whole quasigroup (of order 64×64, or 256×256) and other one-dimensional strings of Edon–$\mathcal{R}$, so the operations can be executed inside the microprocessor and its Level 1 cash memory.

Another approach that can further increase the speed of execution of Edon–$\mathcal{R}$ can be in design of hardware implementation in a special microprocessor that will execute its compression function $\mathcal{R}_1$ in parallel, reducing the execution time to $O(N)$ microprocessor cycles.

5 Conclusions

We have defined an infinite class of one-way hash functions Edon–$\mathcal{R}$ with variable length of output. The cryptographic strength of Edon–$\mathcal{R}$ hash functions relies on conjectured computational infeasibility to solve system of quasigroup equations in shapeless quasigroups.

The property of having possibility to choose the length of hash output can be used in implementation of the applications that require hashes that have lengths different then 128, 160, 256, 384 or 512 bits. With Edon–$\mathcal{R}$ you can produce hash outputs that are 11 or 10111 bytes long. If some day, the computing power of the computers would allow to find collisions of 20 bytes long hashes by brute force, then the design of Edon-$\mathcal{R}$ will not change. Only the output of the hash functions will be increased.

Edon–$\mathcal{R}$ is not a single cryptographic hash function but a huge (infinite) class of cryptographic hash functions, and that fact can be effectively used in the design of different cryptographic protocols.

The compression function $\mathcal{R}_1$ of Edon–$\mathcal{R}$ can be implemented in hardware in parallel manner and then its execution time can be reduced only to $O(N)$ microprocessor cycles.

References

[1] Belousov, V.D.: Osnovi teorii kvazigrup i lup. (1967) "Nauka", Moskva

[2] S. Contini, A. Lenstra, R. Steinfeld, "VSH, an Efficient and Provable Collision Resistant Hash Function", NIST Cryptographic Hash Workshop 2005, http://www.csrc.nist.gov/pki/HashWorkshop/program.htm

[3] 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, volume 3621 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Germany, 2005.

[4] I. B. Damgård, "Collision free hash functions and public key signature schemes", Advances in CryptologyEUROCRYPT 87 (LNCS 304), 1988 203216.

[5] I. B. Damgård, "A design principle for hash functions", Advances in CryptologyCRYPTO 89 LNCS 435, G. Brassard, Ed., Springer-Verlag, 1990, pp. 416-427.

[6] P. Gauravaram, W. Millan and J. G. Nieto, "Some thoughts on Collision Attacks in the Hash Functions MD5, SHA-0 and SHA-1", Cryptology ePrint Archive: Report 2005/391.

[7] J. K. Gibson, "Discrete logarithm hash function that is collision free and one way", IEE proceedings, E: Computers and digital techniques, vol. **138**, nr. 6, 1991, pp. 407–410.

[8] 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)

[9] D. Gligoroski, S. Markovski, L. Kocarev and M. Gusev, "Edon80," ECRYPT Stream Cipher Project, 2005, http://www.ecrypt.eu.org/stream/index.html

[10] M. Goldmann and A. Russell "The complexity of solving equations over finite groups", In Proceedings of the 14th Annual IEEE Conference on Computational Complexity (CCC-99), 1999, pp. 80–86.

[11] M. Hall, "Combinatorial Theory", Blaisdell Publishing Company, Massachusetts, 1967

[12] A. Joux, "Multicollisions in iterated hash functions. Application to cascaded constructions", In M. Franklin, editor, Advances in Cryptology CRYPTO 2004, volume 3152 of Lecture Notes in Computer Science, 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, volume 3494 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, Germany, 2005, pp. 474-490.

[14] X. Lai, J.L. Massey: "Hash functions based on block ciphers", Advances in Cryptology, Proceedings Eurocrypt'92, LNCS 658, R.A. Rueppel, Ed., Springer-Verlag, 1993, pp. 55–70.

[15] S. Lucks, "Design Principles for Iterated Hash Functions", Cryptology ePrint Archive, report 2004/ 253.

[16] S. Markovski, D. Gligoroski, and L. Kocarev, "Unbiased Random Sequences from Quasigroup String Transformations," in Fast Software Encryption 2005, H. Gilbert and H. Handschuh (Eds.), LNCS 3557, 2005, pp. 163-180.

[17] Markovski, S., Gligoroski, D., Bakeva, V.: Quasigroup String Processing: Part 1. Contributions, Sec. Math. Tech. Sci., MANU **XX**, 1-2 (1999) 13–28

[18] S. Markovski, D. Gligoroski, and 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

[19] McKay, B.D., Rogoyski, E.:Latin squares of order 10. Electronic J. Comb. **2** (1995) http://ejc.math.gatech.edu:8080/Journal/journalhome.html

[20] R. Merkle, "One way hash functions and DES," Advances in Cryptology-Crypto'89, LNCS 435, G. Brassard, Ed., Springer-Verlag, 1990, pp. 428–446.

[21] C. Moore, "Predicting non-linear cellular automata quickly by decomposing them into linear ones", Physica (D), vol. 111, pp. 27–41.

[22] C. Moore, D. Therien, F. Lemieux, J. Berman and A Drisko, "Circuits and Expressions with Nonassociative Gates", Journal of Computer and System Sciences, vol. 60, nr 2, 2000, pp. 368–394.

[23] C. Moore, P. Tesson and D. Therien, "Satisfiability of Systems of Equations over Finite Monoids", In Proc. Mathematical Foundations of Computer Science (MFCS) 2001.

[24] B. Preneel, "The State of Cryptographic Hash Functions", Lectures on Data Security, Lecture Notes in Computer Science 1561, I. Damgård (ed.),1999, pp. 158–182.

[25] http://planeta.terra.com.br/informatica/paulobarreto/hflounge.html

[26] B. Preneel, A. Bosselaers, H. Dobbertin, "The cryptographic hash function RIPEMD-160," CryptoBytes, Vol. 3, No. 2, 1997, pp. 9–14.

[27] http://www.esat.kuleuven.ac.be/~bosselae/ripemd160.html

[28] R. L. Rivest, "The MD4 Message Digest Algorithm", Advances in Cryptology - Crypto'90, LNCS 537, Springer-Verlag (1990), pp. 303–311.

[29] R.L. Rivest, A. Shamir, AND L.M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", Communications of the ACM, 21 (1978), pp. 120-126.

[30] B. Schneier, *Applied Cryptography, Second Edition: Protocols, Algorthms, and Source Code in C*, John Wiley & Sons, ISBN 0471128457, 1996.

[31] X. Wang, D. Feng, X. Lai, H. Yu, "Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD", Cryptology ePrint Archive, Report 2004/199.

[32] X. Wang, Y. Yin and H. Yu, "Collision Search Attacks on SHA1", 2005, `http://theory.csail.mit.edu/~ yiqun/shanote.pdf`

[33] Y. Zheng, J. Pieprzyk, and J. Seberry, "HAVAL a one-way hashing algorithm with variable length of output", Advances in Cryptology-AUSCRYPT 92 (LNCS 718), 1993, pp. 83-104.