Cryptographic Hash Function

EDON-$\mathcal{R}$

Norwegian University of Science and Technology
Trondheim, Norway

Danilo Gligoroski
Rune Steinsmo Ødegård
Marija Mihova
Svein Johan Knapskog
Ljupco Kocarev
Aleš Drápal
Vlastimil Klima

January 2009

Abstract

This is the version 02 of the supporting documentation that describes in details the cryptographic hash function EDON-$\mathcal{R}$ which was submitted as a candidate for SHA-3 hash competition organized by National Institute of Standards and Technology (NIST), according to the public call [1].

The difference between version 01 and version 02 of the documentation is in the produced test vectors for HMAC. That is due to the fact that there was mismatch between rotation values defined in the documentation and implemented C code. Accordingly, C source code (in the accompanied CD) has been changed with the correct rotation values. So, in this documentation we do not change anything in the originally submitted algorithm, but just give the correct HMAC test values. In this version a minor change in the performance has been measured with Microsoft Visual Studio 2005, but we add new measurements performed by Intel C++ v 11.0.066 (that are slightly better than those obtained by Microsoft Visual Studio 2005). Additionally, we put a remark that our claims about free-start collisions in the Section 3.14 are not correct.

EDON-$\mathcal{R}$ is a cryptographic hash function with output size of n bits where $n = 224, 256, 384$ or 512. Its conjectured cryptographic security is: $O(2^{\frac{n}{2}})$ hash computations for finding collisions, $O(2^n)$ hash computations for finding preimages, $O(2^{n-k})$ hash computations for finding second preimages for messages shorter than 2^k bits. Additionally, it is resistant against length-extension attacks, resistant against multicollision attacks and it is provably resistant against differential cryptanalysis.

EDON-$\mathcal{R}$ has been designed to be much more efficient than SHA-2 cryptographic hash functions, while in the same time offering same or better security. The speed of the optimized 32-bit version on defined reference platform with Intel C++ v 11.0.066 is 6.26 cycles/byte for $n = 224, 256$ and 9.99 cycles/byte for $n = 384, 512$. The speed of the optimized 64-bit version on defined reference platform with Intel C++ v 11.0.066 is 4.40 cycles/byte for $n = 224, 256$ and 2.29 cycles/byte for $n = 384, 512$.

Contents

Contents

Cover page

- Name of the submitted algorithm: **EDON-$\mathcal{R}$**
- Principal submitter's name, e-mail address, telephone, fax, organization, and postal address:
  - Name: Danilo Gligoroski
  - Email: danilog@item.ntnu.no
  - Tel: +47 73 59 46 16
  - Fax: +47 73 59 69 73
  - Organization: Department of Telematics, Faculty of Information Technology, Mathematics and Electrical Engineering, The Norwegian University of Science and Technology (NTNU), O.S. Bragstads plass 2B, N-7491 Trondheim, Norway
- Name of the algorithm inventor(s)/developer(s):

  Inventors:
  - Danilo Gligoroski

  Developers and contributors:
  - Danilo Gligoroski, Norwegian University of Science and Technology, Norway
  - Rune Steinsmo Ødegård, Norwegian University of Science and Technology, Norway
  - Marija Mihova, "Sts Cyril and Methodius" University, Republic of Macedonia
  - Svein Johan Knapskog, Norwegian University of Science and Technology, Norway
  - Ljupco Kocarev, University of California San Diego, USA and Macedonian Academy of Sciences and Arts
  - Aleš Drápal, Charles University, Czech Republic
  - Vlastimil Klima, Independent cryptologist - consultant, Czech Republic

- Name of the owner, if any, of the algorithm:
  - Danilo Gligoroski
- Signature of the submitter

  ____

- (optional) Backup point of contact (with telephone, fax, postal address, e-mail address):
  - Name: Rune Steinsmo Ødegård
  - Email: rune.odegard@q2s.ntnu.no
  - Tel: +47 73 59 27 80
  - Fax: +47 73 59 27 90
  - Organization: "Centre for Quantifiable Quality of Service in Communication Systems. Centre of Excellence", Faculty of Information Technology, Mathematics and Electrical Engineering, The Norwegian University of Science and Technology (NTNU), O.S. Bragstads plass 2B, N-7491 Trondheim, Norway

CHAPTER 1

Algorithm Specifics

1.1 Bit Strings and Integers

The following terminology related to bit strings, byte strings and integers will be used:

1. A hex digit is an element of the set {`0, 1,..., 9, A, ..., F`}. A hex digit is the representation of a 4–bit string. For example, the hex digit "7" represents the 4–bit string "`0111`", and the hex digit "`A`" represents the 4–bit string "`1010`".

2. The "little-endian" convention is used when expressing string of bytes stored in memory. That means that beginning from some address "H" if the content of the memory is represented as a 1-byte address increment, then 32–bit and 64–bit integers are expressed as in the example given in Table 1.1. The prefix "`0x`" is used to annotate that the integer is expressed in hex digit notation.

3. The "big-endian" convention is used when expressing the "internal bit endianness" for both 32–bit and 64–bit words as integers. That means that within each word, the most significant bit is stored in the left-most bit position. More concretely, a word is a w–bit string that may be represented as a sequence of hex digits. To convert a word to hex digits, each 4–bit string is converted to its hex digit equivalent. For example, the 32–bit string "`1010 0001 0000 0011 1111 1110 0010 0011`" has a hexadecimal representation "`0xA103FE23`" and its value as unsigned long integer is 2701393443. The 64–bit string "`1010 0001 0000 0011 1111 1110 0010 0011 0011 0010 1110 1111 0011 0000 0001 1010`" has a hexadecimal representation "`0xA103FE2332EF301A`" and its value as unsigned long long integer is 11602396492168376346.

4. For EDON-$\mathcal{R}$ hash algorithm, the size of m bits of the message block, depends on the variant

Address in memory	Byte value
H	`23`
H+1	`FE`
H+2	`03`
H+3	`A1`

32–bit integer value: "`0xA103FE23`"

Address in memory	Byte value
H	`1A`
H+1	`30`
H+2	`EF`
H+3	`32`
H+4	`23`
H+5	`FE`
H+6	`03`
H+7	`A1`

64–bit integer value: "`0xA103FE2332EF301A`"

Table 1.1: Default design of the EDON-$\mathcal{R}$ is "Little-endian"

of the algorithm (EDON-$\mathcal{R}$224, EDON-$\mathcal{R}$256, EDON-$\mathcal{R}$384 or EDON-$\mathcal{R}$512).

(a) For EDON-$\mathcal{R}$224 and EDON-$\mathcal{R}$256, each message block has 512 bits, which are represented as a sequence of sixteen 32–bit words.

(b) For EDON-$\mathcal{R}$384 and EDON-$\mathcal{R}$512, each message block has 1024 bits, which are represented as a sequence of sixteen 64–bit words.

1.2 Parameters, variables and constants

The following parameters and variables are used in the specification of EDON-$\mathcal{R}$:

$n = 224, 256, 384, 512$ — The size of the hash digest.

EDON-$\mathcal{R}n$ — Hash algorithm that maps messages into hash values of size n bits.

$w = 32$ or $w = 64$ — w is the size of binary words that are used in EDON-$\mathcal{R}$. In EDON-$\mathcal{R}$224 and EDON-$\mathcal{R}$256 $w = 32$ and in EDON-$\mathcal{R}$384 and EDON-$\mathcal{R}$512 $w = 64$.

M — A message of arbitrary length less than 2^{64} bits.

l — Length of a message.

k — Number of zeroes appended to a message during the padding step.

M'	Padded message with length equal to a multiple of m. $M' = (M^{(1)}, \dots, M^{(N)})$
N	Number of blocks in the padded message.
$m = 512$ or $m = 1024$	Number of bits in the message block $M^{(i)}$.
$M^{(i)}$	i-th message block. Every message block $M^{(i)}$ is represented as a 16 dimensional vector of w-bit words i.e. as $M^{(i)} = (M_0^{(i)}, \dots, M_{15}^{(i)})$ or correspondingly as a pair of two vectors of length 8, $M^{(i)} \equiv (\mathbf{M}_0^{(i)}, \mathbf{M}_1^{(i)})$.
$M_j^{(i)}$	The j-th word of the i-th message block $M^{(i)} = (M_0^{(i)}, \dots, M_{15}^{(i)})$.
$P^{(i)}$	The i-th double pipe value. $P^{(0)}$ is the initial double pipe value. $P^{(N)}$ is the final double pipe value and is used to determine the message digest of n bits. Every double pipe $P^{(i)}$ is represented as a 16 dimensional vector of w-bit words i.e. as $P^{(i)} = (P_0^{(i)}, \dots, P_{15}^{(i)})$ or correspondingly as a pair of two vectors of length 8, $P^{(i)} \equiv (\mathbf{P}_0^{(i)}, \mathbf{P}_1^{(i)})$.
$(Q, *)$	A quasigroup with binary operation $*$.
$q = 256$ or $q = 512$	The exponent q determines the order of the quasigroup, i.e. $\lvert Q \rvert = 2^q$.
Q_{256}, Q_{512}	Quasigroups isotopic to $\left(\left(\mathbb{Z}_{2^w}\right)^8, +_8 \right)$ where $+_8$, is the operation of componentwise addition of two 8–dimensional vectors in $\left(\mathbb{Z}_{2^w}\right)^8$.
$\mathcal{R}$	Quasigroup reverse string transformation.
$\mathbb{Z}_2^w$	The set of binary words of length $w \in \{32, 64\}$.
π_1, π_2, π_3	Permutations of the sets $\{0,1\}^{256}, \{0,1\}^{512}$.
$\oplus_w$	Addition in $\mathbb{Z}_2^w$ (bitwise XOR of two w–bit words).
$\mathbf{X}$	Elements of Q_{256}, Q_{512}.
$H(M)$	Hash of message M.

Algorithm abbreviation	Message size l (in bits)	Block size m (in bits)	Word size w (in bits)	Endianess	Digest size n (in bits)	Support of "one-pass" streaming mode
EDON-$\mathcal{R}$224	$< 2^{64}$	512	32	Little-endian	224	Yes
EDON-$\mathcal{R}$256	$< 2^{64}$	512	32	Little-endian	256	Yes
EDON-$\mathcal{R}$384	$< 2^{64}$	1024	64	Little-endian	384	Yes
EDON-$\mathcal{R}$512	$< 2^{64}$	1024	64	Little-endian	512	Yes

Table 1.2: Basic properties of all four variants of the EDON-$\mathcal{R}$

1.3 General design properties of EDON-$\mathcal{R}$

EDON-$\mathcal{R}$ follows the general design pattern that is common for most of the known hash functions. It means that it has two stages (and several sub-stages within every stage):

1. Preprocessing
   (a) padding a message,
   (b) parsing the padded message into m–bit blocks, and
   (c) setting initialization values to be used in the hash computation.
2. Hash computation
   (a) using a conjectured one-way operation with huge quasigroups iteratively generates series of double pipe values,
   (b) The n Least Significant Bits (LSB) of the final double pipe value generated by the hash computation are used to determine the message digest.

Depending on the context we will sometimes refer to the hash function as EDON-$\mathcal{R}$ and sometimes as EDON-$\mathcal{R}$224, EDON-$\mathcal{R}$256, EDON-$\mathcal{R}$384 or EDON-$\mathcal{R}$512.

In Table 1.2, we give the basic properties of all four variants of the EDON-$\mathcal{R}$ hash algorithms.

The following operations are applied in EDON-$\mathcal{R}$:

1. Bitwise logic word operations $\oplus$ – XOR.
2. Addition $+$ modulo 2^{32} or modulo 2^{64}.

3. Rotate left (circular left shift) operation, $ROTL^r(x)$, where x is a 32–bit or 64–bit word and r is an integer with $0 \leq r < 32$ (resp. $0 \leq r < 64$).

1.4 Preprocessing

Preprocessing consists of three steps:

1. padding the message M,
2. parsing the padded message into message blocks, and
3. setting the initial double pipe value, $P^{(0)}$.

1.4.1 Padding the message

The message M, shall be padded before hash computation begins. The purpose of this padding is to ensure that the padded message is a multiple of 512 or 1024 bits, depending on the size of the message digest n.

EDON-$\mathcal{R}$224 and EDON-$\mathcal{R}$256

Suppose that the length of the message M is l bits. Append the bit "1" to the end of the message, followed by k zero bits, where k is the smallest, non-negative solution to the equation $l + 1 + k \equiv 448 \texttt{ mod } 512$. Then append the 64–bit block that is equal to the number l expressed using a binary representation. For example, the (8–bit ASCII) message "abc" has length $8 \times 3 = 24$, so the message is padded with the bit "1", then $448 - (24 + 1) = 423$ zero bits, and then the 64–bit binary representation of the number 24, to become the 512–bit padded message.

$$\underbrace{\texttt{01100001}}_{''\text{a}''}\ \underbrace{\texttt{01100010}}_{''\text{b}''}\ \underbrace{\texttt{01100011}}_{''\text{c}''}\ \texttt{1}\ \overbrace{\texttt{00}\ldots\texttt{00}}^{423}\ \overbrace{\texttt{00}\ldots\underbrace{\texttt{011000}}_{l=24}}^{64}$$

EDON-$\mathcal{R}$384 and EDON-$\mathcal{R}$512

Suppose that the length of the message M is l bits. Append the bit "1" to the end of the message, followed by k zero bits, where k is the smallest, non-negative solution to the equation $l + 1 + k \equiv 960 \texttt{ mod } 1024$. Then append the 64–bit block that is equal to the number l expressed using a binary representation. For example, the (8–bit ASCII) message "abc" has length $8 \times 3 = 24$, so the message is padded with the bit "1", then $960 - (24 + 1) = 935$ zero bits, and then the 64–bit binary representation of the number 24, to become the 1024–bit padded message.

$$\underbrace{\texttt{01100001}}_{''\mathrm{a}''}\ \underbrace{\texttt{01100010}}_{''\mathrm{b}''}\ \underbrace{\texttt{01100011}}_{''\mathrm{c}''}\ \texttt{1}\ \overbrace{\texttt{00}\ldots\texttt{00}}^{935}\ \overbrace{\texttt{00}\ldots\underbrace{\texttt{011000}}_{l=24}}^{64}$$

1.4.2 Parsing the message

After a message has been padded, it must be parsed into N m–bit blocks before the hash computation can begin.

EDON-$\mathcal{R}$224 and EDON-$\mathcal{R}$256

For EDON-$\mathcal{R}$224 and EDON-$\mathcal{R}$256, the padded message is parsed into N 512–bit blocks, $M^{(1)}$, $M^{(2)}, \ldots, M^{(N)}$. Since the 512 bits of the input block may be expressed as sixteen 32–bit words, the first 32 bits of message block i are denoted $M_0^{(i)}$, the next 32 bits are $M_1^{(i)}$, and so on up to $M_{15}^{(i)}$.

EDON-$\mathcal{R}$384 and EDON-$\mathcal{R}$512

For EDON-$\mathcal{R}$384 and EDON-$\mathcal{R}$512, the padded message is parsed into N 1024–bit blocks, $M^{(1)}$, $M^{(2)}, \ldots, M^{(N)}$. Since the 1024 bits of the input block may be expressed as sixteen 64–bit words, the first 64 bits of message block i are denoted $M_0^{(i)}$, the next 64 bits are $M_1^{(i)}$, and so on up to $M_{15}^{(i)}$.

1.4.3 Setting the initial double pipe value $P^{(0)}$

Before the hash computation begins for each of the hash algorithms, the initial double pipe value $P^{(0)}$ must be set. The size and the value of words in $P^{(0)}$ depend on the message digest size n. As it is shown in the following subsections the constants consist of a concatenation of the consecutive 8-bit natural numbers. Since EdonR224 is the same as EdonR256 except for the final truncation, they have to have different initial values. Thus, the initial double pipe of EdonR224 starts from the byte value `0x00` and takes all 64 successive byte values up to the value `0x3F`. Then, the initial double pipe of EdonR256 starts from the byte value `0x40` and takes all 64 successive byte values up to the value `0x7F`. The situation is the same with EdonR384 and EdonR512, but since now the variables are 64-bit long, the initial double pipe of EdonR384 starts from the byte value `0x00` and

$P_0^{(0)} =$ 0x00010203	$P_1^{(0)} =$ 0x04050607
$P_2^{(0)} =$ 0x08090A0B	$P_3^{(0)} =$ 0x0C0D0E0F
$P_4^{(0)} =$ 0x10111213	$P_5^{(0)} =$ 0x14151617
$P_6^{(0)} =$ 0x18191A1B	$P_7^{(0)} =$ 0x1C1D1E1F
$P_8^{(0)} =$ 0x20212223	$P_9^{(0)} =$ 0x24252627
$P_{10}^{(0)} =$ 0x28292A2B	$P_{11}^{(0)} =$ 0x2C2D2E2F
$P_{12}^{(0)} =$ 0x30313233	$P_{13}^{(0)} =$ 0x24353637
$P_{14}^{(0)} =$ 0x38393A3B	$P_{15}^{(0)} =$ 0x3C3D3E3F

Table 1.3: Initial double pipe $P^{(0)}$ for EDON-$\mathcal{R}$224

$P_0^{(0)} =$ 0x40414243	$P_1^{(0)} =$ 0x44454647
$P_2^{(0)} =$ 0x48494A4B	$P_3^{(0)} =$ 0x4C4D4E4F
$P_4^{(0)} =$ 0x50515253	$P_5^{(0)} =$ 0x54555657
$P_6^{(0)} =$ 0x58595A5B	$P_7^{(0)} =$ 0x5C5D5E5F
$P_8^{(0)} =$ 0x60616263	$P_9^{(0)} =$ 0x64656667
$P_{10}^{(0)} =$ 0x68696A6B	$P_{11}^{(0)} =$ 0x6C6D6E6F
$P_{12}^{(0)} =$ 0x70717273	$P_{13}^{(0)} =$ 0x74757677
$P_{14}^{(0)} =$ 0x78797A7B	$P_{15}^{(0)} =$ 0x7C7D7E7F

Table 1.4: Initial double pipe $P^{(0)}$ for EDON-$\mathcal{R}$256

takes all 128 successive byte values up to the value `0x7F` and the initial double pipe of EdonR512 starts from the byte value `0x80` and takes all 128 successive byte values up to the value `0xFF`.

EDON-$\mathcal{R}$224

For EDON-$\mathcal{R}$224, the initial double pipe value $P^{(0)}$ shall consist of sixteen 32–bit words given in Table 1.3.

EDON-$\mathcal{R}$256

For EDON-$\mathcal{R}$256, the initial double pipe value $P^{(0)}$ shall consist of sixteen 32–bit words given in Table 1.4.

$P_0^{(0)}$ = 0x0001020304050607	$P_1^{(0)}$ = 0x08090A0B0C0D0E0F
$P_2^{(0)}$ = 0x1011121314151617	$P_3^{(0)}$ = 0x18191A1B1C1D1E1F
$P_4^{(0)}$ = 0x2021222324252627	$P_5^{(0)}$ = 0x28292A2B2C2D2E2F
$P_6^{(0)}$ = 0x3031323324353637	$P_7^{(0)}$ = 0x38393A3B3C3D3E3F
$P_8^{(0)}$ = 0x4041424344454647	$P_9^{(0)}$ = 0x48494A4B4C4D4E4F
$P_{10}^{(0)}$ = 0x5051525354555657	$P_{11}^{(0)}$ = 0x58595A5B5C5D5E5F
$P_{12}^{(0)}$ = 0x6061626364656667	$P_{13}^{(0)}$ = 0x68696A6B6C6D6E6F
$P_{14}^{(0)}$ = 0x7071727374757677	$P_{15}^{(0)}$ = 0x78797A7B7C7D7E7F

Table 1.5: Initial double pipe $P^{(0)}$ for Edon-$\mathcal{R}$384

$P_0^{(0)}$ = 0x8081828384858687	$P_1^{(0)}$ = 0x88898A8B8C8D8E8F
$P_2^{(0)}$ = 0x9091929394959697	$P_3^{(0)}$ = 0x98999A9B9C9D9E9F
$P_4^{(0)}$ = 0xA0A1A2A3A4A5A6A7	$P_5^{(0)}$ = 0xA8A9AAABACADAEAF
$P_6^{(0)}$ = 0xB0B1B2B3B4B5B6B7	$P_7^{(0)}$ = 0xB8B9BABBBCBDBEBF
$P_8^{(0)}$ = 0xC0C1C2C3C4C5C6C7	$P_9^{(0)}$ = 0xC8C9CACBCCCDCECF
$P_{10}^{(0)}$ = 0xD0D1D2D3D4D5D6D7	$P_{11}^{(0)}$ = 0xD8D9DADBDCDDDEDF
$P_{12}^{(0)}$ = 0xE0E1E2E3E4E5E6E7	$P_{13}^{(0)}$ = 0xE8E9EAEBECEDEEEF
$P_{14}^{(0)}$ = 0xF0F1F2F3F4F5F6F7	$P_{15}^{(0)}$ = 0xF8F9FAFBFCFDFEFF

Table 1.6: Initial double pipe $P^{(0)}$ for Edon-$\mathcal{R}$512

Edon-$\mathcal{R}$384

For Edon-$\mathcal{R}$384, the initial double pipe value $P^{(0)}$ shall consist of sixteen 64–bit words given in Table 1.4.3.

Edon-$\mathcal{R}$512

For Edon-$\mathcal{R}$512, the initial double pipe value $P^{(0)}$ shall consist of sixteen 64–bit words given in Table 1.4.3.

CHAPTER 2

Description of the Hash Algorithm EDON-$\mathcal{R}$

2.1 Mathematical preliminaries and notation

In this section we need to repeat some parts of the definition of the class of one-way candidate functions recently defined in [2, 3]. For that purpose we need also several brief definitions for quasigroups and quasigroup string transformations.

Definition 1. A quasigroup $(Q, *)$ is an algebraic structure consisting of a nonempty set Q and a binary operation $* : Q^2 \rightarrow Q$ with the property that each of the equations

$$\begin{aligned} a * x &= b \\ y * a &= b \end{aligned} \tag{2.1.1}$$

has unique solutions x and y in Q.

Closely related combinatorial structures to finite quasigroups are Latin squares, since the main body of the multiplication table of a quasigroup is just a Latin square.

Definition 2. A Latin square is an $n \times n$ table filled with n different symbols in such a way that each symbol occurs exactly once in each row and exactly once in each column.

Definition 3. A pair of Latin squares is said to be orthogonal if the n^2 pairs formed by juxtaposing the two squares are all distinct.

More detailed information about theory of quasigroups, quasigroup string processing, Latin squares and hash functions can be found in [4–8].

Quasigroup operation of order 2^{256}

Input: $\mathbf{X} = (X_0, X_1, \ldots, X_7)$ and $\mathbf{Y} = (Y_0, Y_1, \ldots, Y_7)$

where X_i and Y_i are 32–bit variables.

Output: $\mathbf{Z} = (Z_0, Z_1, \ldots, Z_7)$ where Z_i are 32–bit variables.

Temporary 32–bit variables: $T_0, \ldots, T_{15}$.

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

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

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

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

Table 2.1: An algorithmic description of a quasigroup of order 2^{256}.

2.1.1 Algorithmic definition of quasigroups of orders 2^{256} and 2^{512}

First we give an algorithmic description of an operation that takes two eight-component vectors $\mathbf{X} = (X_0, X_1, \ldots, X_7)$ and $\mathbf{Y} = (Y_0, Y_1, \ldots, Y_7)$ where X_i and Y_i are either 32–bit or 64–bit variables, and computes a new eight-component vector $\mathbf{Z} = (Z_0, Z_1, \ldots, Z_7)$. Operation "+" denotes addition modulo 2^{32} or modulo 2^{64}, the operation $\oplus$ is the logical operation of bitwise exclusive or and the operation $ROTL^{r}(X_i)$ is the operation of bit rotation of the 32–bit or 64–bit variable X_i, to the left for r positions.

Quasigroup operation of order 2^{512}

Input: $\mathbf{X} = (X_0, X_1, \ldots, X_7)$ and $\mathbf{Y} = (Y_0, Y_1, \ldots, Y_7)$

where X_i and Y_i are 64–bit variables.

Output: $\mathbf{Z} = (Z_0, Z_1, \ldots, Z_7)$ where Z_i are 64–bit variables.

Temporary 64–bit variables: $T_0, \ldots, T_{15}$.

1.
$$\begin{array}{lll}
T_0 & \leftarrow & ROTL^0(\texttt{0xAAAAAAAAAAAAAAAA} + X_0 + X_1 + X_2 + X_4 + X_7); \\
T_1 & \leftarrow & ROTL^5(X_0 + X_1 + X_3 + X_4 + X_7); \\
T_2 & \leftarrow & ROTL^{15}(X_0 + X_1 + X_4 + X_6 + X_7); \\
T_3 & \leftarrow & ROTL^{22}(X_2 + X_3 + X_5 + X_6 + X_7); \\
T_4 & \leftarrow & ROTL^{31}(X_1 + X_2 + X_3 + X_5 + X_6); \\
T_5 & \leftarrow & ROTL^{40}(X_0 + X_2 + X_3 + X_4 + X_5); \\
T_6 & \leftarrow & ROTL^{50}(X_0 + X_1 + X_5 + X_6 + X_7); \\
T_7 & \leftarrow & ROTL^{59}(X_2 + X_3 + X_4 + X_5 + X_6);
\end{array}$$

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

3.
$$\begin{array}{lll}
T_0 & \leftarrow & ROTL^0(\texttt{0x5555555555555555} + Y_0 + Y_1 + Y_2 + Y_5 + Y_7); \\
T_1 & \leftarrow & ROTL^{10}(Y_0 + Y_1 + Y_3 + Y_4 + Y_6); \\
T_2 & \leftarrow & ROTL^{19}(Y_0 + Y_1 + Y_2 + Y_3 + Y_5); \\
T_3 & \leftarrow & ROTL^{29}(Y_2 + Y_3 + Y_4 + Y_6 + Y_7); \\
T_4 & \leftarrow & ROTL^{36}(Y_0 + Y_1 + Y_3 + Y_4 + Y_5); \\
T_5 & \leftarrow & ROTL^{44}(Y_2 + Y_4 + Y_5 + Y_6 + Y_7); \\
T_6 & \leftarrow & ROTL^{48}(Y_1 + Y_2 + Y_5 + Y_6 + Y_7); \\
T_7 & \leftarrow & ROTL^{55}(Y_0 + Y_3 + Y_4 + Y_6 + Y_7);
\end{array}$$

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

Table 2.2: An algorithmic description of a quasigroup of order 2^{512}.

2.1.2 Algebraic definition of quasigroups of orders 2^{256} and 2^{512}

In this subsection we will present the same quasigroups that have been described in the previous subsection in an algebraic way.

For that purpose we use the following notation: we identify Q as a set of cardinality 2^q, $q = 256, 512$, and elements $x \in Q$ are represented in their bitwise form as q-bit words

$$\mathbf{X} \equiv (\widetilde{x}_0, \widetilde{x}_1, \ldots, \widetilde{x}_{q-2}, \widetilde{x}_{q-1}) \equiv \widetilde{x}_0 \cdot 2^{q-1} + \widetilde{x}_1 \cdot 2^{q-2} + \ldots + \widetilde{x}_{q-2} \cdot 2 + \widetilde{x}_{q-1}$$

where $\widetilde{x}_i \in \{0, 1\}$, but also as the set $\Big(\mathbb{Z}_{2^w}\Big)^8$, $w = 32, 64$.

Actually, we shall be constructing quasigroups $(Q, *)$ as isotopes of $\Big(\big(\mathbb{Z}_{2^w}\big)^8, +_8\Big)$, $w = 32, 64$ where $+_8$, is the operation of componentwise addition of two 8–dimensional vectors in $\big(\mathbb{Z}_{2^w}\big)^8$. We shall thus define three permutations $\pi_i : \mathbb{Z}_2^q \rightarrow \mathbb{Z}_2^q, 1 \leq i \leq 3$ so that

$$\mathbf{X} * \mathbf{Y} \equiv \pi_1(\pi_2(\mathbf{X}) +_8 \pi_3(\mathbf{Y}))$$

for all $\mathbf{X}, \mathbf{Y} \in \Big(\mathbb{Z}_{2^w}\Big)^8$.

Let us denote by $Q_{256} = \{0, 1\}^{256}$ and $Q_{512} = \{0, 1\}^{512}$ the corresponding sets of 256–bit and 512–bit words. Our intention is to define EDON-$\mathcal{R}$ by the following bitwise operations on w–bit values (where $w = 32$ or $w = 64$):

1. Addition between w–bit words modulo 2^w,
2. Rotation of w–bits to the left for r positions,
3. Bitwise XOR operations $\oplus$ on w–bit words.

Further, we introduce the following convention:

- Elements $\mathbf{X} \in Q_{256}$ are represented as $\mathbf{X} = (X_0, X_1, \ldots, X_7)$, where X_i are 32–bit words,
- Elements $\mathbf{X} \in Q_{512}$ are represented as $\mathbf{X} = (X_0, X_1, \ldots, X_7)$ where X_i are 64–bit words.

The left rotation of a w–bit word Y by r positions will be denoted by $ROTL^r(Y)$. Note that this operation can be expressed as a linear matrix–vector multiplication over the ring $(\mathbb{Z}_2, +, \times)$ i.e. $ROTL^r(Y) = \mathbf{E}^r \cdot Y$ where $\mathbf{E}^r \in \mathbb{Z}_2^w \times \mathbb{Z}_2^w$ is a matrix obtained from the identity matrix by rotating its columns by r positions in the direction top to bottom. Further on, if we have a vector $X \in Q_q$ where $q = 256, 512$ represented as $\mathbf{X} = (X_0, X_1, \ldots, X_7)$ and we want to rotate all X_i by r_i $(0 \leq$

$i \leq 7$) positions to the left, then we denote that operation by $ROTL^{\mathbf{r}}(\mathbf{X})$, where $\mathbf{r} = (r_0, \ldots, r_7) \in \{0, 1, \ldots, w-1\}^7$ is called the rotation vector. The operation $ROTL^{\mathbf{r}}(\mathbf{X})$ can also be represented as a linear matrix–vector multiplication over the ring $(\mathbb{Z}_2, +, \times)$ i.e. $ROTL^{\mathbf{r}}(\mathbf{X}) = \mathbf{D}^{\mathbf{r}} \cdot \mathbf{X}$ where $\mathbf{D}^{\mathbf{r}} \in \mathbb{Z}_2^q \times \mathbb{Z}_2^q$,

$$\mathbf{D}^{\mathbf{r}} = \begin{pmatrix} \mathbf{E}^{r_0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{E}^{r_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{E}^{r_2} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{E}^{r_3} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{E}^{r_4} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{E}^{r_5} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{E}^{r_6} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{E}^{r_7} \end{pmatrix},$$

submatrices $\mathbf{E}^{r_i} \in \mathbb{Z}_2^w \times \mathbb{Z}_2^w$, $0 \leq i \leq 7$ are obtained from the identity matrix by rotating its columns by r_i positions in the direction top to bottom, and the submatrices $\mathbf{0} \in \mathbb{Z}_2^w \times \mathbb{Z}_2^w$ are the zero matrix.

Further on, we use the following notation:

- $\widehat{\mathbb{A}}_1, \widehat{\mathbb{A}}_3 : \left(\mathbb{Z}_{2^w}\right)^8 \rightarrow \left(\mathbb{Z}_{2^w}\right)^8$ are two bijective transformations in $\left(\mathbb{Z}_{2^w}\right)^8$ over the ring $(\mathbb{Z}_{2^w}, +, \times)$ where $w = 32$ or $w = 64$. The mappings $\widehat{\mathbb{A}}_i, i = 1, 3$ can be described as:

  $$\widehat{\mathbb{A}}_i(\mathbf{X}) = \mathbf{C}_i + \mathbb{A}_i \cdot \mathbf{X},$$

  where $\mathbf{C}_i \in \left(\mathbb{Z}_{2^w}\right)^8, i = 1, 2$ are two constant vectors and $\mathbb{A}_1$ and $\mathbb{A}_3$ are two 8×8 invertible matrices over the ring $(\mathbb{Z}_{2^w}, +, \times)$. Since they look like affine transformations in vector fields, sometimes we will call these two transformations also "*affine bijective transformations*" although strictly speaking we are not working in any vector field. All elements in those two matrices are either 0 or 1, since we want to avoid the operations of multiplication (as more costly microprocessor operations) in the ring $(\mathbb{Z}_{2^w}, +, \times)$, and stay only with operations of addition.

- $\mathbb{A}_2, \mathbb{A}_4 : \left(\mathbb{Z}_{2^w}\right)^8 \rightarrow \left(\mathbb{Z}_{2^w}\right)^8$ are two linear bijective transformations of Q_q that are described by two invertible matrices (we use the same notation: $\mathbb{A}_2, \mathbb{A}_4$) of order $q \times q$ over the ring $(\mathbb{Z}_2, +, \times)$ ($q = 256$ or $q = 512$). Since we want to apply XOR operations on w–bit registers,

q	$\mathbf{r}_{1,q}$	$\mathbf{r}_{2,q}$
256	(0, 4, 8, 13, 17, 22, 24, 29)	(0, 5, 9, 11, 15, 20, 25, 27)
512	(0, 5, 15, 22, 31, 40, 50, 59)	(0, 10, 19, 29, 36, 44, 48, 55)

Table 2.3: Rotation vectors for definition of π_2 and π_3.

the matrices $\mathbb{A}_2$ and $\mathbb{A}_4$ will be of the form

$$\begin{pmatrix} \mathbb{B}_{1,1} & \mathbb{B}_{1,2} & \dots & \mathbb{B}_{1,8} \\ \mathbb{B}_{2,1} & \mathbb{B}_{2,2} & \dots & \mathbb{B}_{2,8} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{B}_{8,1} & \mathbb{B}_{8,2} & \dots & \mathbb{B}_{8,8} \end{pmatrix},$$

where $\mathbb{B}_{i,j} \in \mathbb{Z}_2^w \times \mathbb{Z}_2^w, \ \ 1 \leq i,j \leq 8$ are either the identity matrix or the zero matrix i.e. $\mathbb{B}_{i,j} \in \{\mathbf{0}, \mathbf{1}\}$.

Now we give the formal definitions for the permutations: π_1, π_2 and π_3.

Definition 4. Transformations $\pi_1 : Q_q \rightarrow Q_q$ ($q = 256, 512$) are defined as:

$$\pi_1(X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7) = (X_5, X_6, X_7, X_0, X_1, X_2, X_3, X_4)$$

Lemma 1. Transformations π_1 are permutations. □

Definition 5. Transformations $\pi_2 : Q_q \rightarrow Q_q$ and $\pi_3 : Q_q \rightarrow Q_q$ are defined as:

$$\pi_2 \equiv \widehat{\mathbb{A}}_1 \circ ROTL^{\mathbf{r}_{1,q}} \circ \mathbb{A}_2$$
$$\pi_3 \equiv \widehat{\mathbb{A}}_3 \circ ROTL^{\mathbf{r}_{2,q}} \circ \mathbb{A}_4$$

where the rotation vectors $\mathbf{r}_{i,q}, \ i = 1, 2, \ q = 256, 512$ are given in Table 2.3, and the information about $\widehat{\mathbb{A}}_1$, $\mathbb{A}_2$, $\widehat{\mathbb{A}}_3$ and $\mathbb{A}_4$ is given in Table 2.4. There, the symbols $\mathbf{1}, \mathbf{0} \in \mathbb{Z}_2^w \times \mathbb{Z}_2^w$ are the identity matrix and the zero matrix, and the constants $const_{i,q}, \ i = 1, 2, \ q = 256, 512$ have the following values (given in hexadecimal notation): $const_{1,256}$ = 0xAAAAAAAA, $const_{2,256}$ = 0x55555555, $const_{1,512}$ = 0xAAAAAAAAAAAAAAAA and $const_{2,512}$ = 0x5555555555555555. The rationale for choosing these constants is in Section 3.12.

Lemma 2. Transformations π_2 and π_3 are permutations on $Q_q, \ q = 256, 512$.

$\widehat{\mathbb{A}}_1$	$\mathbb{A}_2$	$\widehat{\mathbb{A}}_3$	$\mathbb{A}_4$
$\begin{pmatrix} const_{1,q} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1&1&1&0&1&0&0&1 \\ 1&1&0&1&1&0&0&1 \\ 1&1&0&0&1&0&1&1 \\ 0&0&1&1&0&1&1&1 \\ 0&1&1&1&0&1&1&0 \\ 1&0&1&1&1&1&0&0 \\ 1&1&0&0&0&1&1&1 \\ 0&0&1&1&1&1&1&0 \end{pmatrix}$	$\begin{pmatrix} \mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{1}&\mathbf{0}&\mathbf{1}&\mathbf{1}&\mathbf{0} \\ \mathbf{0}&\mathbf{0}&\mathbf{1}&\mathbf{0}&\mathbf{0}&\mathbf{1}&\mathbf{1}&\mathbf{0} \\ \mathbf{0}&\mathbf{0}&\mathbf{1}&\mathbf{1}&\mathbf{0}&\mathbf{1}&\mathbf{0}&\mathbf{0} \\ \mathbf{1}&\mathbf{1}&\mathbf{0}&\mathbf{0}&\mathbf{1}&\mathbf{0}&\mathbf{0}&\mathbf{0} \\ \mathbf{1}&\mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{1}&\mathbf{0}&\mathbf{0}&\mathbf{1} \\ \mathbf{0}&\mathbf{1}&\mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{1}&\mathbf{1} \\ \mathbf{0}&\mathbf{0}&\mathbf{1}&\mathbf{1}&\mathbf{1}&\mathbf{0}&\mathbf{0}&\mathbf{0} \\ \mathbf{1}&\mathbf{1}&\mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{1} \end{pmatrix}$	$\begin{pmatrix} const_{2,q} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1&1&1&0&0&1&0&1 \\ 1&1&0&1&1&0&1&0 \\ 1&1&1&1&0&1&0&0 \\ 0&0&1&1&1&0&1&1 \\ 1&1&0&1&1&1&0&0 \\ 0&0&1&0&1&1&1&1 \\ 0&1&1&0&0&1&1&1 \\ 1&0&0&1&1&0&1&1 \end{pmatrix}$	$\begin{pmatrix} \mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{1}&\mathbf{1}&\mathbf{0}&\mathbf{1}&\mathbf{0} \\ \mathbf{0}&\mathbf{0}&\mathbf{1}&\mathbf{0}&\mathbf{0}&\mathbf{1}&\mathbf{0}&\mathbf{1} \\ \mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{1}&\mathbf{0}&\mathbf{1}&\mathbf{1} \\ \mathbf{1}&\mathbf{1}&\mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{1}&\mathbf{0}&\mathbf{0} \\ \mathbf{0}&\mathbf{0}&\mathbf{1}&\mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{1}&\mathbf{1} \\ \mathbf{1}&\mathbf{1}&\mathbf{0}&\mathbf{1}&\mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{0} \\ \mathbf{1}&\mathbf{0}&\mathbf{0}&\mathbf{1}&\mathbf{1}&\mathbf{0}&\mathbf{0}&\mathbf{0} \\ \mathbf{0}&\mathbf{1}&\mathbf{1}&\mathbf{0}&\mathbf{0}&\mathbf{1}&\mathbf{0}&\mathbf{0} \end{pmatrix}$

Table 2.4: Matrices $\widehat{\mathbb{A}}_1$, $\mathbb{A}_2$, $\widehat{\mathbb{A}}_3$ and $\mathbb{A}_4$.

Proof. The proof follows immediately from the fact that all transformations $\mathbb{A}_i$, $i = 1, 2, 3, 4$ and $ROTL^{\mathbf{r}_{i,q}}, i = 1, 2,\ q = 256, 512$ are expressed by invertible matrices over the rings $(\mathbb{Z}_{2^w}, +, \times)$, $w = 32, 64$ or over the ring $(\mathbb{Z}_2, +, \times)$. □

Theorem 1. Operations $*_q : Q_q^2 \rightarrow Q_q$ defined as:

$$\mathbf{X} *_q \mathbf{Y} = \pi_1(\pi_2(\mathbf{X})\ +_8\ \pi_3(\mathbf{Y}))$$

are non-commutative quasigroup operations that are not loops.

Proof. We give a proof for $q = 256$ and the other case for $q = 512$ is similar.

To show that $*_{256}$ is not a loop we have to show that there is no unit element $\mathbf{E} \in Q_{256}$ such that for every $\mathbf{A} \in Q_{256}$, $\mathbf{A} *_{256} \mathbf{E} = \mathbf{A} = \mathbf{E} *_{256} \mathbf{A}$. Let us suppose that there is a neutral element $\mathbf{E} \in Q_{256}$. Let us first put

$$\pi_2(\mathbf{E}) -_8 \pi_3(\mathbf{E}) = \mathbf{Const}_E$$

where $\mathbf{Const}_E \in Q_{256}$ is a constant element and the operation $-_8$ is the componentwise subtraction modulo 2^{32}.

From the concrete definition of the quasigroup operation $*_{256}$ for the neutral element $\mathbf{E}$ we get:

$$\pi_1(\pi_2(\mathbf{E}) +_8 \pi_3(\mathbf{A})) = \pi_1(\pi_2(\mathbf{A}) +_8 \pi_3(\mathbf{E}))$$

Since π_1 is a permutation we can remove it from the last equation and we get:

$$\pi_2(\mathbf{E}) +_8 \pi_3(\mathbf{A}) = \pi_2(\mathbf{A}) +_8 \pi_3(\mathbf{E})$$

and if we rearrange the last equation we get:

$$\pi_2(\mathbf{A}) -_8 \pi_3(\mathbf{A}) = \pi_2(\mathbf{E}) -_8 \pi_3(\mathbf{E}) = \mathbf{Const}_E$$

The last equation states that for every $\mathbf{A} \in Q_{256}$ the expression $\pi_2(\mathbf{A}) -_8 \pi_3(\mathbf{A})$ is a constant. This is not true. For example $\pi_2(1) -_8 \pi_3(1) \neq \pi_2(2) -_8 \pi_3(2)$. Thus we conclude that $*_{256}$ is not a loop. □

Note that the quasigroups cannot be associative since every associative quasigroup is a group and every group possesses a unit element.

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

Definition 6.

1. $\mathcal{R}_{256} : Q_{256}^4 \rightarrow Q_{256}^2 \equiv \mathcal{R}$ where $\mathcal{R}$ is defined as in Definition 8 over Q_{256} with the quasigroup operation $*_{256}$.
2. $\mathcal{R}_{512} : Q_{512}^4 \rightarrow Q_{512}^2 \equiv \mathcal{R}$ where $\mathcal{R}$ is defined as in Definition 8 over Q_{512} with the quasigroup operation $*_{512}$.

2.1.3 The one-way function $\mathcal{R}$: A reverse quasigroup string transformation

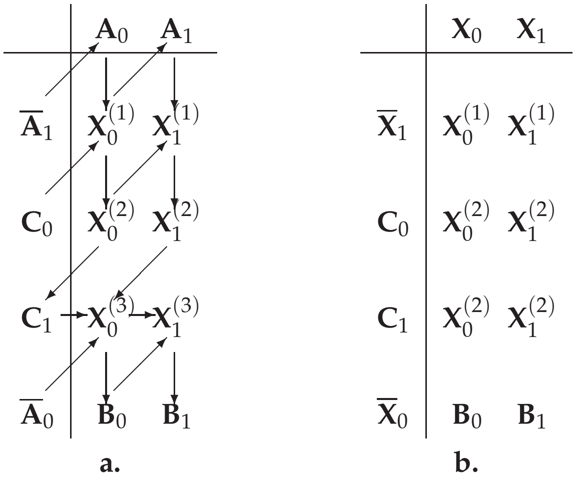
The reverse quasigroup string transformation as a candidate one-way function has been introduced in [9], and a generic hash function with reverse quasigroup string transformation has been described in [2, 3]. A concrete hash function with similar name: Edon-R(n) for $n = 256, 384, 512$ has been described in [10]. Many properties from that function are present in the design of EDON-$\mathcal{R}$, but we can say that without loosing security properties of the hash function, the design of EDON-$\mathcal{R}$ is now simplified and performance is much better compared to the older Edon-R(n). Additionally, we can say that the concept of reverse quasigroup string transformation is present also in another cryptographic primitive - the stream cipher Edon80 [11]. Edon80 IV Setup procedure is a conjectured one-way function and so far no cryptographic weaknesses have been found for the Edon80 IV Setup, although Edon80 have been under the public scrutiny from the cryptographic community for more than 3 years.

Definition 7. For a given $\mathbf{X} \in Q_q, q = 256, 512$, which can be represented as an eight component vector $\mathbf{X} = (X_0, X_1, \ldots, X_7) \in \left(\mathbb{Z}_{2^w}\right)^8, w = 32, 54$, the reversed vector $\overline{\mathbf{X}}$ is defined as:

$$\overline{\mathbf{X}} = (X_7, X_6, \ldots, X_0)$$

Definition 8. For a given quasigroup $*_q, q = 256, 512$, (shortly denoted as $*$ i.e. without the index q) the one-way function $\mathcal{R} : Q_q^4 \rightarrow Q_q^2$ used in EDON-$\mathcal{R}$ hash function is defined as:

$$\mathcal{R}(\mathbf{C}_0, \mathbf{C}_1, \mathbf{A}_0, \mathbf{A}_1) \quad = \quad (\mathbf{B}_0, \mathbf{B}_1)$$



	$\mathbf{A}_0$	$\mathbf{A}_1$
$\overline{\mathbf{A}}_1$	$\mathbf{X}_0^{(1)}$	$\mathbf{X}_1^{(1)}$
$\mathbf{C}_0$	$\mathbf{X}_0^{(2)}$	$\mathbf{X}_1^{(2)}$
$\mathbf{C}_1$	$\mathbf{X}_0^{(3)}$	$\mathbf{X}_1^{(3)}$
$\overline{\mathbf{A}}_0$	$\mathbf{B}_0$	$\mathbf{B}_1$

a.

	$\mathbf{X}_0$	$\mathbf{X}_1$
$\overline{\mathbf{X}}_1$	$\mathbf{X}_0^{(1)}$	$\mathbf{X}_1^{(1)}$
$\mathbf{C}_0$	$\mathbf{X}_0^{(2)}$	$\mathbf{X}_1^{(2)}$
$\mathbf{C}_1$	$\mathbf{X}_0^{(2)}$	$\mathbf{X}_1^{(2)}$
$\overline{\mathbf{X}}_0$	$\mathbf{B}_0$	$\mathbf{B}_1$

b.

Table 2.5: a. Schematic representation of the function $\mathcal{R}$, **b.** Conjectured one-wayness of $\mathcal{R}$ comes from the difficulty to solve a system of two equations where $\mathbf{B}_0$, $\mathbf{B}_1$, $\mathbf{C}_0$ and $\mathbf{C}_1$ are given, and $\mathbf{A}_0 = X_0$ and $\mathbf{A}_1 = X_1$ are indeterminate variables.

where

$$\mathbf{B}_0 = \overline{\mathbf{A}}_0 * ((\mathbf{C}_0 * (\overline{\mathbf{A}}_1 * \mathbf{A}_0)) * \mathbf{C}_1$$

$$\mathbf{B}_1 = (\overline{\mathbf{A}}_0 * ((\mathbf{C}_0 * (\overline{\mathbf{A}}_1 * \mathbf{A}_0)) * \mathbf{C}_1) * ((\mathbf{C}_0 * (\overline{\mathbf{A}}_1 * \mathbf{A}_0)) * ((\overline{\mathbf{A}}_1 * \mathbf{A}_0) * \mathbf{A}_1) * (\mathbf{C}_0 * (\overline{\mathbf{A}}_1 * \mathbf{A}_0)) * \mathbf{C}_1).$$

The reasons why the expressions for $\mathbf{B}_0$ and $\mathbf{B}_1$ are as they are given in Definition 8 can be easily understood by looking at the Table 2.5a. The diagonal arrows can be interpreted as quasigroup operations between the source and the destination, and the vertical or the horizontal arrows as equality signs "=".

The conjectured one-wayness of $\mathcal{R}$ can be explained by Table 2.5b. Let us assume that only the values $\mathbf{B}_0$, $\mathbf{B}_1$, $\mathbf{C}_0$ and $\mathbf{C}_1$ are given. In order to find pre-image values $\mathbf{A}_0 = \mathbf{X}_0$ and $\mathbf{A}_1 = \mathbf{X}_1$ we can use Definition 8 and obtain the following relations for the elements of Table 2.5b:

$$\begin{gathered}
\mathbf{X}_0^{(1)} = \overline{\mathbf{A}}_1 * \mathbf{A}_0 \\
\mathbf{X}_1^{(1)} = \mathbf{X}_0^{(1)} * \mathbf{A}_1 = (\overline{\mathbf{A}}_1 * \mathbf{A}_0) * \mathbf{A}_1 \\
\mathbf{X}_0^{(2)} = \mathbf{C}_0 * \mathbf{X}_0^{(1)} = \mathbf{C}_0 * (\overline{\mathbf{A}}_1 * \mathbf{A}_0) \\
\mathbf{X}_1^{(2)} = \mathbf{X}_0^{(2)} * \mathbf{X}_1^{(1)} = (\mathbf{C}_0 * (\overline{\mathbf{A}}_1 * \mathbf{A}_0)) * ((\overline{\mathbf{A}}_1 * \mathbf{A}_0) * \mathbf{A}_1) \\
\mathbf{X}_0^{(3)} = \mathbf{X}_0^{(2)} * \mathbf{C}_1 = (\mathbf{C}_0 * (\overline{\mathbf{A}}_1 * \mathbf{A}_0)) * \mathbf{C}_1 \\
\mathbf{X}_1^{(3)} = \mathbf{X}_1^{(2)} * \mathbf{X}_0^{(3)} = ((\mathbf{C}_0 * (\overline{\mathbf{A}}_1 * \mathbf{A}_0)) * ((\overline{\mathbf{A}}_1 * \mathbf{A}_0) * \mathbf{A}_1)) * ((\mathbf{C}_0 * (\overline{\mathbf{A}}_1 * \mathbf{A}_0)) * \mathbf{C}_1) \\
\mathbf{B}_0 = \overline{\mathbf{A}}_0 * \mathbf{X}_0^{(3)} = \overline{\mathbf{A}}_0 * ((\mathbf{C}_0 * (\overline{\mathbf{A}}_1 * \mathbf{A}_0)) * \mathbf{C}_1) \\
\mathbf{B}_1 = \mathbf{B}_0 * \mathbf{X}_1^{(3)} = \\
(\overline{\mathbf{A}}_0 * ((\mathbf{C}_0 * (\overline{\mathbf{A}}_1 * \mathbf{A}_0)) * \mathbf{C}_1)) * (((\mathbf{C}_0 * (\overline{\mathbf{A}}_1 * \mathbf{A}_0)) * ((\overline{\mathbf{A}}_1 * \mathbf{A}_0) * \mathbf{A}_1)) * ((\mathbf{C}_0 * (\overline{\mathbf{A}}_1 * \mathbf{A}_0)) * \mathbf{C}_1))
\end{gathered}$$

From them, we can obtain the following system of quasigroup equations with indeterminates $\mathbf{X}_0, \mathbf{X}_1$:

$$\begin{cases} \mathbf{B}_0 = \overline{\mathbf{X}}_0 * ((\mathbf{C}_0 * (\overline{\mathbf{X}}_1 * \mathbf{X}_0)) * \mathbf{C}_1) \\ \mathbf{B}_1 = (\overline{\mathbf{X}}_0 * ((\mathbf{C}_0 * (\overline{\mathbf{X}}_1 * \mathbf{X}_0)) * \mathbf{C}_1)) * (((\mathbf{C}_0 * (\overline{\mathbf{X}}_1 * \mathbf{X}_0)) * ((\overline{\mathbf{X}}_1 * \mathbf{X}_0) * \mathbf{X}_1)) * ((\mathbf{C}_0 * (\overline{\mathbf{X}}_1 * \mathbf{X}_0)) * \mathbf{C}_1)). \end{cases}$$

One can show that for any given $\mathbf{A}_0 = \mathbf{X}_0 \in Q$ either there are values of $\mathbf{A}_1 = \mathbf{X}_1$ as a solution or there is no solution. However, if the quasigroup operation is non-commutative, non-associative, the quasigroup operations are not linear in the underlying algebraic structure, and if the size of the quasigroup is very big (for example 2^{256} or 2^{512}) then solving this simple system of just two quasigroup equations is hard. Actually there is no known efficient method for solving such systems of quasigroup equations.

Of course, one inefficient method for solving that system would be to try every possible value for $\mathbf{A}_0 = \mathbf{X}_0 \in Q$ until obtaining the other indeterminate $\mathbf{A}_1 = \mathbf{X}_1$. That brute force method would require in average $\frac{1}{2}|Q|$ attempts to guess $\mathbf{A}_0 = \mathbf{X}_0 \in Q$ before solving the system.

2.2 Generic description for all variants of the Edon-$\mathcal{R}$

First we are giving a generic description for all variants of the Edon-$\mathcal{R}$ hash algorithm. Then, in the following subsections we are giving some concrete specifics for four different message digest sizes: $n = 224$, $n = 256$, $n = 384$ and $n = 512$. The generic description of Edon-$\mathcal{R}$ hash algorithm is given in Table 2.6.

Algorithm: EDON-$\mathcal{R}$

Input: Message M of length l bits, and the message digest size n.

Output: A message digest *Hash*, that is long n bits.

```
1. Preprocessing
    (a) Pad the message $M$.
    (b) Parse the padded message into $N$, $m$-bit message blocks, $M^{(1)}, M^{(2)}, \dots, M^{(N)}$.
    (c) Set the initial value of the double pipe $P^{(0)}$.
2. Hash computation
    For $i = 1$ to $N$
        $P^{(i)} = \mathcal{R}(P^{(i-1)}, M^{(i)});$
3. $Hash$ =Take_$n$_Least_Significant_Bits$(P^{(N)})$.
```

Table 2.6: A generic description of the EDON-$\mathcal{R}$ hash algorithm

In the generic description the words of the initial double pipe $P_0^{(i-1)}, P_1^{(i-1)}, \dots, P_{15}^{(i-1)}$ are represented as two vectors of length 8 i.e. $(P_0^{(i-1)}, P_1^{(i-1)}, \dots, P_{15}^{(i-1)}) \equiv (\mathbf{P}_0^{(0)}, \mathbf{P}_1^{(0)}) \equiv P^{(0)}$. Then, by each iteration, they are replaced by intermediate double pipe value, $P^{(i)} = (\mathbf{P}_0^{(i)}, \mathbf{P}_1^{(i)})$, ending with the final double pipe value $P^{(N)} = (\mathbf{P}_0^{(N)}, \mathbf{P}_1^{(N)})$. The final result of EDON-$\mathcal{R}$ is a n–bit message digest that are the least significant n bits from the final double pipe.

Similar notation is used for the values of the padded message $M' = (M^{(1)}, M^{(2)}, \dots, M^{(N)})$. Namely, every message block $M^{(i)}$ is represented as a pair of two vectors of length 8, $M^{(i)} \equiv (\mathbf{M}_0^{(i)}, \mathbf{M}_1^{(i)})$. A graphic representation of the EDON-$\mathcal{R}$ hash algorithm is given in the Figure 2.1.

2.2.1 EDON-$\mathcal{R}$224 and EDON-$\mathcal{R}$256

EDON-$\mathcal{R}$224 and EDON-$\mathcal{R}$256 may be used to hash a message M, having a length of l bits, where $0 \leq l < 2^{64}$. The algorithms use:

1. A double pipe of sixteen 32–bit working variables represented as a pair of two vectors of length eight, and

2. in every iteration it needs additional sixteen 32–bit working variables that come from the

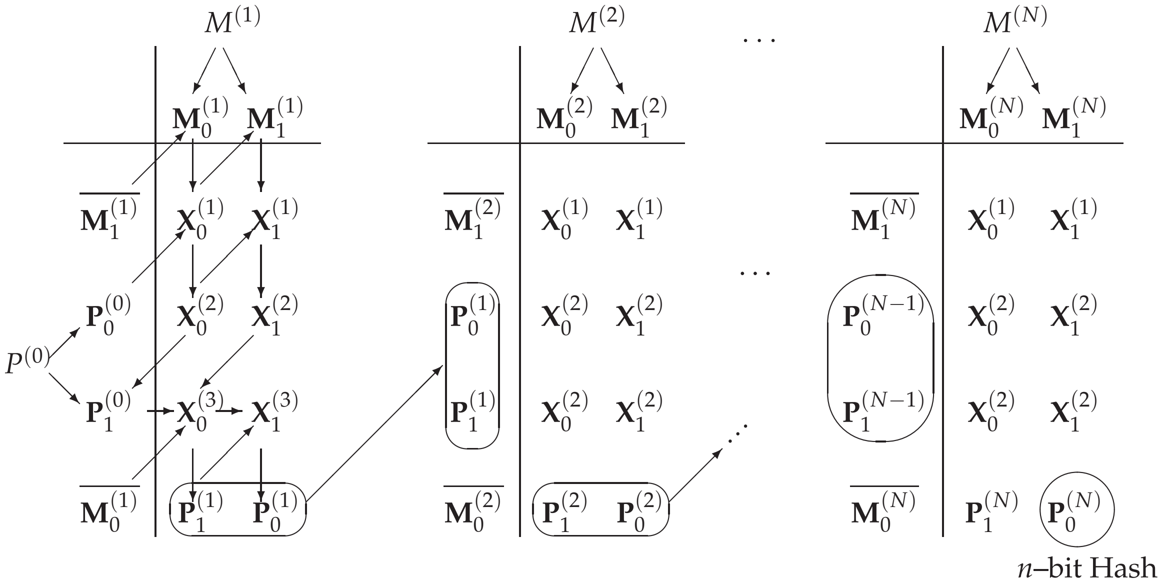

$M^{(1)}$
$M^{(2)}$
...
$M^{(N)}$
$\mathbf{M}_0^{(1)}$ $\mathbf{M}_1^{(1)}$
$\mathbf{M}_0^{(2)}$ $\mathbf{M}_1^{(2)}$
$\mathbf{M}_0^{(N)}$ $\mathbf{M}_1^{(N)}$
$\overline{\mathbf{M}_1^{(1)}}$ $\mathbf{X}_0^{(1)}$ $\mathbf{X}_1^{(1)}$
$\overline{\mathbf{M}_1^{(2)}}$ $\mathbf{X}_0^{(1)}$ $\mathbf{X}_1^{(1)}$
$\overline{\mathbf{M}_1^{(N)}}$ $\mathbf{X}_0^{(1)}$ $\mathbf{X}_1^{(1)}$
$\mathbf{P}_0^{(0)}$ $\mathbf{X}_0^{(2)}$ $\mathbf{X}_1^{(2)}$
$\mathbf{P}_0^{(1)}$ $\mathbf{X}_0^{(2)}$ $\mathbf{X}_1^{(2)}$
...
$\mathbf{P}_0^{(N-1)}$ $\mathbf{X}_0^{(2)}$ $\mathbf{X}_1^{(2)}$
$P^{(0)}$
$\mathbf{P}_1^{(0)}$ $\mathbf{X}_0^{(3)}$ $\mathbf{X}_1^{(3)}$
$\mathbf{P}_1^{(1)}$ $\mathbf{X}_0^{(2)}$ $\mathbf{X}_1^{(2)}$
$\mathbf{P}_1^{(N-1)}$ $\mathbf{X}_0^{(2)}$ $\mathbf{X}_1^{(2)}$
$\overline{\mathbf{M}_0^{(1)}}$ $\mathbf{P}_1^{(1)}$ $\mathbf{P}_0^{(1)}$
$\overline{\mathbf{M}_0^{(2)}}$ $\mathbf{P}_1^{(2)}$ $\mathbf{P}_0^{(2)}$
...
$\overline{\mathbf{M}_0^{(N)}}$ $\mathbf{P}_1^{(N)}$ $\mathbf{P}_0^{(N)}$
n–bit Hash

Figure 2.1: A graphic representation of the EDON-$\mathcal{R}$ hash algorithm.

message (represented as a pair of two vectors of length eight).

EDON-$\mathcal{R}$224 and EDON-$\mathcal{R}$256 preprocessing

1. Pad the message M.

2. Parse the padded message into N 512–bit blocks, $M^{(1)}$, $M^{(2)}$,..., $M^{(N)}$.

3. Set the initial double pipe value $P^{(0)}$ as defined in Table 1.3 for BWM224, or as defined in Table 1.4 for BWM256.

2.2.2 EDON-$\mathcal{R}$384 and EDON-$\mathcal{R}$512

EDON-$\mathcal{R}$384 and EDON-$\mathcal{R}$512 may be used to hash a message M, having a length of l bits, where $0 \leq l < 2^{64}$. The algorithms use

1. A double pipe of sixteen 64–bit working variables represented as a pair of two vectors of length eight, and

2. in every iteration it needs additional sixteen 64–bit working variables that come from the message (represented as a pair of two vectors of length eight).

EDON-$\mathcal{R}$384 and EDON-$\mathcal{R}$512 preprocessing

1. Pad the message M.

2. Parse the padded message into N 1024–bit blocks, $M^{(1)}$, $M^{(2)}$,. . . , $M^{(N)}$.

3. Set the initial double pipe value $P^{(0)}$ as defined in Table 1.4.3 for BWM384, or as defined in Table 1.4.3 for BWM512.

CHAPTER 3

Design Rationale

3.1 Choosing 32–bit and 64–bit operations

We have decided to choose just three types of operations: addition modulo 2^{32} or modulo 2^{64}, XOR-ing and left rotations. This is 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 other cryptographic primitives have been designed following the same rationale such as: Salsa20 [12], The Tiny Encryption Algorithm [13], IDEA [14], SHA-1 and SHA-2 - to name a few.

3.2 Reasons for default little-endian design

Previous versions of Edon-R(n) as well as the earliest version of EDON-$\mathcal{R}$ were designed to be big-endian by default. However, as the designing phase was coming to its end, and we started the optimization phase, we changed the default design to be little-endian since an overwhelming majority of CPU platforms in the world are little-endian.

3.3 Choosing permutations π_1, π_2 and π_3

Our goal was to design a structure that is a non-commutative, non-associative, highly nonlinear quasigroup of order 2^{256} (or 2^{512}) in order to apply the principles of the hash family Edon–$\mathcal{R}$ first presented on the second NIST cryptographic hash workshop [2]. We have found a way to construct such structures by applying some basic permutations π_1, π_2 and π_3 on the sets $\{0,1\}^{256}$ and $\{0,1\}^{512}$.

$$L_1 = \left[\begin{array}{cccccccc} 0 & 7 & 1 & 3 & 2 & 4 & 6 & 5 \\ 4 & 1 & 7 & 6 & 3 & 0 & 5 & 2 \\ 7 & 0 & 4 & 2 & 5 & 3 & 1 & 6 \\ 1 & 4 & 0 & 5 & 6 & 2 & 7 & 3 \\ 2 & 3 & 6 & 7 & 1 & 5 & 0 & 4 \\ \hline 5 & 2 & 3 & 1 & 7 & 6 & 4 & 0 \\ 3 & 6 & 5 & 0 & 4 & 7 & 2 & 1 \\ 6 & 5 & 2 & 4 & 0 & 1 & 3 & 7 \end{array}\right] = \left[\begin{array}{c} L_{1,1} \\ L_{1,2} \end{array}\right] \qquad L_2 = \left[\begin{array}{cccccccc} 0 & 4 & 2 & 3 & 1 & 6 & 5 & 7 \\ 7 & 6 & 3 & 2 & 5 & 4 & 1 & 0 \\ 5 & 3 & 1 & 6 & 0 & 2 & 7 & 4 \\ 1 & 0 & 5 & 4 & 3 & 7 & 2 & 6 \\ 2 & 1 & 0 & 7 & 4 & 5 & 6 & 3 \\ \hline 3 & 5 & 7 & 0 & 6 & 1 & 4 & 2 \\ 4 & 7 & 6 & 1 & 2 & 0 & 3 & 5 \\ 6 & 2 & 4 & 5 & 7 & 3 & 0 & 1 \end{array}\right] = \left[\begin{array}{c} L_{2,1} \\ L_{2,2} \end{array}\right]$$

Table 3.1: Two mutually orthogonal Latin squares used to define the permutations π_2 and π_3

The permutation π_1 is simple rotation on 256 or 512–bit words. It can be effectively realized just by appropriate referencing of the 32–bit (resp. 64–bit) variables. The role of the permutation π_1 is to do the componentwise mixing (diffusion) on the whole q–bit word. That diffusion then have influence on the next application of the quasigroup operation $*_q$ (since we apply two such operations in every row). The decision to define π_1 as:

$$\pi_1(X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7) = (X_5, X_6, X_7, X_0, X_1, X_2, X_3, X_4),$$

i.e., as a rotation to the right for 3 positions was done because 3 is relatively prime to 8.

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. That nonlinear mixing is achieved because we perform operations in two different rings: $(\mathbb{Z}_{2^w}, +, \times), w = 32, 64$ and $(\mathbb{Z}_2, +, \times)$. For the choice of the permutations π_2 and π_3 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 as well as in cryptography [5, 15–18].

3.4 Criteria for choosing the Latin squares - part one

For permutations π_2 and π_3 we used two orthogonal Latin squares of order 8 given in Table 3.1.

By splitting L_1 and L_2 in two (upper and lower) Latin rectangles $L_{1,1}$, $L_{1,2}$, $L_{2,1}$ and $L_{2,2}$ and taking columns of those rectangles as sets, we actually constructed four symmetric non-balanced block designs (for an excellent brief introduction on block designs see for example [19]). The non-balanced symmetric block designs corresponding to $L_{1,1}$ and $L_{2,1}$ are with parameters $(v, k, \lambda) = (8, 5, \lambda)$ where $\lambda \in \{2, 3, 4\}$, and those corresponding to $L_{1,2}$ and $L_{2,2}$ are with parameters $(v, k, \lambda) =$

$(8, 3, \lambda)$ where $\lambda \in \{0, 1, 2\}$. We used the incidence matrix obtained by $L_{1,1}$ to bijectively transform the variables by addition modulo $2^w, w = 32, 64$ (work in the ring $(\mathbb{Z}_{2^w}, +, \times)$) and the incidence matrix obtained by $L_{1,2}$ to bijectively transform the variables by XORing of w–bit variables (work in the ring $(\mathbb{Z}_2, +, \times)$).

As we mentioned in Section 2.1.2, the matrix $\mathbb{A}_1$ is an 8×8 invertible matrix in the ring $(\mathbb{Z}_{2^w}, +, \times)$, $w = 32, 64$ and the matrix $\mathbb{A}_2$ is a $q \times q$, $(q = 256, 512)$ invertible matrix in the ring $(\mathbb{Z}_2, +, \times)$. Similarly, from the Latin rectangles $L_{2,1}$ and $L_{2,2}$ we got the invertible incidence matrices $\mathbb{A}_3$ and $\mathbb{A}_4$.

It is an interesting observation that we split the Latin rectangles in 5:3 ratio, not in 4:4 ratio. It comes from the fact that the symmetry of the corresponding formulas for calculation of the determinant of the incidence matrices when the splitting is 4:4, always gives the result 0 (singular value) in the ring $(\mathbb{Z}_2, +, \times)$.

3.5 EDON-$\mathcal{R}$ is provably resistant against differential cryptanalysis

In this section we will prove the resistance of EDON-$\mathcal{R}$ against differential cryptanalysis. We will achieve that by examining the differential characteristics of the permutations π_2 and π_3. More specifically we will trace how one bit difference is diffused by π_2 and π_3. Additionally, this will explain our rationale for choosing permutations π_2 and π_3.

Let us first recall the algebraic definition of π_2 and π_3:

$$\pi_2 \equiv \widehat{\mathbb{A}}_1 \circ ROTL^{\mathbf{r}_{1,q}} \circ \mathbb{A}_2,$$
$$\pi_3 \equiv \widehat{\mathbb{A}}_3 \circ ROTL^{\mathbf{r}_{2,q}} \circ \mathbb{A}_4,$$

where the rotation vectors $\mathbf{r}_{i,q}$, $i = 1, 2$, $q = 256, 512$ are given in Table 2.3, and the information about $\widehat{\mathbb{A}}_1$, $\mathbb{A}_2$, $\widehat{\mathbb{A}}_3$ and $\mathbb{A}_4$ is given in Table 2.4.

Although the matrices $\mathbb{A}_2$ and $\mathbb{A}_4$ are $q \times q$ matrices, because of their special form which is composed just from the block matrices $\mathbf{0}$ and $\mathbf{1}$ (i.e. the zero matrix and the identity matrix) we are abusing the notation and in this section we are annotating with $\mathbb{A}_2$ and $\mathbb{A}_4$ also as 8×8 matrices.

Additionally, let us recall that the matrices $\mathbb{A}_1$ and $\mathbb{A}_2$ are obtained from the Latin square L_1 defined in Table 3.1 ($\mathbb{A}_1$ is obtained as an incident matrix from the upper 5×8 Latin rectangle $L_{1,1}$ and $\mathbb{A}_2$ is obtained as an incident matrix from the lower 3×8 Latin rectangle $L_{1,2}$), and that the matrices $\mathbb{A}_3$ and $\mathbb{A}_4$ are obtained from the Latin square L_2 defined in Table 3.1 ($\mathbb{A}_3$ is obtained as an incident matrix from the upper 5×8 Latin rectangle $L_{2,1}$ and $\mathbb{A}_4$ is obtained as an incident matrix from the lower 3×8 Latin rectangle $L_{2,2}$).

Definition 9. For a given Boolean matrix $\mathbb{M}_{8\times 8} = (m_{i,j}), m_{i,j} \in \{0,1\}$ and for every column $j \in \{0,7\}$ we define the set of non-zero elements of the j-th column as $R_{\mathbb{M},j} = \{i | m_{i,j} = 1\}$.

Note that indexing of the columns and rows in the matrix $\mathbb{M}$ is from 0 to 7 and that for matrices $\mathbb{A}_1$ and $\mathbb{A}_3$ in every column there are exactly 5 ones i.e., $|R_{\mathbb{A}_i,j}| = 5,\ i = 1,3,\ \forall j \in \{0,\ldots,7\}$.

Definition 10. For the Latin rectangles $L_{1,2} = (l_{i,j}^{(1)}),\ L_{2,2} = (l_{i,j}^{(2)})\ i = 0,1,2,\ j = 0,1,\ldots 7$, we denote the sets of elements of their j-th column as $L_{1,2}^{(j)} = \{l_{0,j}^{(1)}, l_{1,j}^{(1)}, l_{2,j}^{(1)}\}$ and as $L_{2,2}^{(j)} = \{l_{0,j}^{(2)}, l_{1,j}^{(2)}, l_{2,j}^{(2)}\}$.

Definition 11. For the permutation $\pi_2 : Q_q \to Q_q,\ q = 256, 512$ defined as: $\pi_2 \equiv \widehat{\mathbb{A}}_1 \circ ROTL^{\mathbf{r}_{1,q}} \circ \mathbb{A}_2$ the diffusion matrix $\mathbf{Diff}_{\pi_2} = (d_{i,j})$ is a square matrix of order 8×8 where

$$d_{i,j} = |R_{\mathbb{A}_1,i} \cap L_{1,2}^{(j)}|.$$

For the permutation $\pi_3 : Q_q \to Q_q,\ q = 256, 512$ defined as: $\pi_3 \equiv \widehat{\mathbb{A}}_3 \circ ROTL^{\mathbf{r}_{2,q}} \circ \mathbb{A}_4$ the diffusion matrix $\mathbf{Diff}_{\pi_3} = (d_{i,j})$ is a square matrix of order 8×8 where

$$d_{i,j} = |R_{\mathbb{A}_3,i} \cap L_{2,2}^{(j)}|.$$

Diffusion matrices for π_2 and π_3 are given in Table 3.2.

$\mathbf{Diff}_{\pi_2}$	$\mathbf{Diff}_{\pi_3}$
$\begin{pmatrix} 2&3&2&2&1&2&1&2 \\ 1&2&1&3&2&2&2&2 \\ 2&1&2&2&3&1&2&2 \\ 2&1&2&2&2&2&2&2 \\ 1&2&2&2&2&2&1&3 \\ 3&2&2&1&2&2&2&1 \\ 2&2&2&1&2&2&3&1 \\ 2&2&2&2&1&2&2&2 \end{pmatrix}$	$\begin{pmatrix} 1&2&2&2&2&2&2&2 \\ 2&1&2&2&2&2&2&2 \\ 2&2&1&2&2&2&2&2 \\ 2&2&2&1&2&2&2&2 \\ 2&2&2&2&1&2&2&2 \\ 2&2&2&2&2&1&2&2 \\ 2&2&2&2&2&2&1&2 \\ 2&2&2&2&2&2&2&1 \end{pmatrix}$

Table 3.2: Diffusion matrices $\mathbf{Diff}_{\pi_2}$ and $\mathbf{Diff}_{\pi_3}$.

Based on the definition of the diffusion matrices for π_2 and π_3 it is relatively straightforward to prove the following proposition:

Proposition 1. $\mathbf{Diff}_{\pi_2} = \Big(\mathbb{A}_1 \cdot \mathbb{A}_2\Big)^T$ and $\mathbf{Diff}_{\pi_3} = \Big(\mathbb{A}_3 \cdot \mathbb{A}_4\Big)^T$, where $\mathbb{A}_i, i = 1,2,3,4$ are the matrices given in Table 2.4 and "^{T}" is a transposition of a matrix. □

In Table 3.3 we give the absolute value of the eigenvalues and the corresponding eigenvectors for both diffusion matrices $\mathbf{Diff}_{\pi_2}$ and $\mathbf{Diff}_{\pi_3}$. Notice the interesting property that both matrices have: For the biggest eigenvalue λ_1 its corresponding eigenvector is $\mathbf{s}_1 = (1,1,1,1,1,1,1,1)$. This

	$\lvert\lambda_1\rvert$	$\lvert\lambda_2\rvert$	$\lvert\lambda_3\rvert$	$\lvert\lambda_4\rvert$	$\lvert\lambda_5\rvert$	$\lvert\lambda_6\rvert$	$\lvert\lambda_7\rvert$	$\lvert\lambda_8\rvert$
Eigenvalues	15.0	1.55603	1.55603	1.0	1.0	1.0	0.642661	0.642661
	$\mathbf{s}_1$	$\mathbf{s}_2$	$\mathbf{s}_3$	$\mathbf{s}_4$	$\mathbf{s}_5$	$\mathbf{s}_6$	$\mathbf{s}_7$	$\mathbf{s}_8$
and eigenvectors for $\mathbf{Diff}_{\pi_2}$	$\begin{pmatrix} 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \end{pmatrix}$	$\begin{pmatrix} -1.50411 - 1.22685i \\ 1.74693 - 2.46378i \\ -1.0 \\ 1.50411 + 1.22685i \\ -0.104877 - 1.55249i \\ -1.74693 + 2.46378i \\ 0.104877 + 1.55249i \\ 1.0 \end{pmatrix}$	$\begin{pmatrix} -1.50411 + 1.22685i \\ 1.74693 + 2.46378i \\ -1.0 \\ 1.50411 - 1.22685i \\ -0.104877 + 1.55249i \\ -1.74693 - 2.46378i \\ 0.104877 - 1.55249i \\ 1.0 \end{pmatrix}$	$\begin{pmatrix} 5.0 \\ -1.0 \\ -7.0 \\ -1.0 \\ 3.0 \\ -3.0 \\ 1.0 \\ 3.0 \end{pmatrix}$	$\begin{pmatrix} 2.0 \\ 1.0 \\ -4.0 \\ -1.0 \\ -1.0 \\ 2.0 \\ 0.0 \\ 1.0 \end{pmatrix}$	$\begin{pmatrix} -1.0 \\ 0.0 \\ 1.0 \\ 0.0 \\ 0.0 \\ -1.0 \\ 1.0 \\ 0.0 \end{pmatrix}$	$\begin{pmatrix} 0.504108 - 0.106312i \\ 0.253069 + 0.213498i \\ -1.0 \\ -0.504108 + 0.106312i \\ -0.395123 - 0.506844i \\ -0.253069 - 0.213498i \\ 0.395123 + 0.506844i \\ 1.0 \end{pmatrix}$	$\begin{pmatrix} 0.504108 + 0.106312i \\ 0.253069 - 0.213498i \\ -1.0 \\ -0.504108 - 0.106312i \\ -0.395123 + 0.506844i \\ -0.253069 + 0.213498i \\ 0.395123 - 0.506844i \\ 1.0 \end{pmatrix}$
	$\lvert\lambda_1\rvert$	$\lvert\lambda_2\rvert$	$\lvert\lambda_3\rvert$	$\lvert\lambda_4\rvert$	$\lvert\lambda_5\rvert$	$\lvert\lambda_6\rvert$	$\lvert\lambda_7\rvert$	$\lvert\lambda_8\rvert$
Eigenvalues	15.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	$\mathbf{s}_1$	$\mathbf{s}_2$	$\mathbf{s}_3$	$\mathbf{s}_4$	$\mathbf{s}_5$	$\mathbf{s}_6$	$\mathbf{s}_7$	$\mathbf{s}_8$
and eigenvectors for $\mathbf{Diff}_{\pi_3}$	$\begin{pmatrix} 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \end{pmatrix}$	$\begin{pmatrix} -1.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 1.0 \end{pmatrix}$	$\begin{pmatrix} -1.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 1.0 \\ 0.0 \end{pmatrix}$	$\begin{pmatrix} -1.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 1.0 \\ 0.0 \\ 0.0 \end{pmatrix}$	$\begin{pmatrix} -1.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 1.0 \\ 0.0 \\ 0.0 \\ 0.0 \end{pmatrix}$	$\begin{pmatrix} -1.0 \\ 0.0 \\ 0.0 \\ 1.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \end{pmatrix}$	$\begin{pmatrix} -1.0 \\ 0.0 \\ 1.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \end{pmatrix}$	$\begin{pmatrix} -1.0 \\ 1.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \end{pmatrix}$

Table 3.3: Eigenvalues and the eigenvectors for the diffusion matrices $\mathbf{Diff}_{\pi_2}$ and $\mathbf{Diff}_{\pi_3}$.

property, as we will show in this and in the following sections, is the crucial one for proving that EDON-$\mathcal{R}$ hash function is resistant against differential cryptanalysis.

In what follows, when X and Y are two w-bit words, the notation $Hamming(X, Y) = \delta$ denotes that X and Y differs in exactly δ bits.

Theorem 2. Let $\mathbf{X}, \mathbf{X}' \in Q_q$ be represented as $\mathbf{X} = (X_0, \ldots, X_7)$ and $\mathbf{X}' = (X'_0, \ldots, X'_7)$, and let $\mathbf{Y} = \pi_a(\mathbf{X})$, $\mathbf{Y}' = \pi_a(\mathbf{X}')$, $a = 2, 3$. If $\mathbf{X}$ and $\mathbf{X}'$ differ in one bit, i.e. the Hamming distance $Hamming(\mathbf{X}, \mathbf{X}') = 1$, and if that one-bit distance is in the i-th word i.e. $Hamming(X_i, X'_i) = 1$, $i = 0, \ldots, 7$, then

$$Hamming(Y_j, Y'_j) \geq d_{i,j},\ j = 0, \ldots, 7$$

where $d_{i,j}$ are values in the matrix $\mathbf{Diff}_{\pi_a}$.

Proof. We will prove the theorem for the case $a = 2$ i.e., for the permutation π_2, and the other case for the permutation π_3 is similar.

Let us recall that $\pi_2 = \widehat{\mathbb{A}}_1 \circ ROTL^{\mathbf{r}_{1,q}} \circ \mathbb{A}_2$ i.e. for $\mathbf{X} \in Q_q$,

$$\pi_2(\mathbf{X}) = \mathbb{A}_2(ROTL^{\mathbf{r}_{1,q}}(\widehat{\mathbb{A}}_1(\mathbf{X}))).$$

Further let us denote by $\mathbf{\Delta}$ the difference vector between $\mathbf{X}$ and $\mathbf{X}'$ i.e.

$$\mathbf{\Delta} = \mathbf{X} \oplus \mathbf{X}'.$$

Moreover, since from the conditions in the theorem we have that one-bit distance is in the i-th word i.e. $Hamming(X_i, X_i') = 1,\ i = 0, \ldots, 7$, we can say that

$$\mathbf{\Delta} = (\underbrace{0, \ldots, 0}_{i-1}, \Delta_i, \underbrace{0, \ldots, 0}_{7-i}),$$

where $0 \in Z_{2^w}$ and $\Delta_i = X_i \oplus X_i'$. Now, instead of using two direct transformations $\pi_2(\mathbf{X})$ and $\pi_2(\mathbf{X}')$ in order to trace the differences we will work with $\pi_2(\mathbf{\Delta})$.

Having in mind that the operation addition modulo 2^w of two variables $X, X' \in Z_{2^w}, w = 32, 64$ that differ in one bit i.e. when $Hamming(X, X') = 1$, with any constant $C \in Z_{2^w}$, does not decrease the Hamming distance, i.e.

$$Hamming(X + C, X' + C) \geq 1,\ \forall C \in Z_{2^w},$$

we have that

$$\mathbf{\Delta_1} = \widehat{\mathbb{A}}_1(\mathbf{\Delta}) = (\delta_0^{(1)}, \delta_1^{(1)}, \ldots, \delta_7^{(1)})$$

where

$$\delta_j^{(1)} = \begin{cases} 0, & if\ j \notin R_{\mathbb{A}_1,i} \\ \Delta_j, & if\ j \in R_{\mathbb{A}_1,i}. \end{cases}$$

So, $\mathbf{\Delta_1}$ has exactly 5 nonzero elements since $|R_{\mathbb{A}_1,i}| = 5,\ \forall i \in \{0, 7\}$. However, the rightmost position of bit differences in every $\Delta_j, j \in R_{\mathbb{A}_1,i}$ is the same since the difference actually comes from the original difference $\Delta_i = X_i \oplus X_i'$. The situation changes after applying rotation transformation $ROTL^{\mathbf{r}_{1,q}}$ on $\mathbf{\Delta_1}$. Let

$$\mathbf{\Delta_2} = ROTL^{\mathbf{r}_{1,q}}(\mathbf{\Delta_1}) = (\delta_0^{(2)}, \delta_1^{(2)}, \ldots, \delta_7^{(2)}),$$

where

$$\delta_j^{(2)} = \begin{cases} 0, & if\ j \notin R_{\mathbb{A}_1,i} \\ ROTL^{r_j}(\Delta_j), & if\ j \in R_{\mathbb{A}_1,i}. \end{cases}$$

Now $\mathbf{\Delta_2}$ has also exactly 5 nonzero elements, but the rightmost position of differences in every $\delta_0^{(2)}, j \in R_{\mathbb{A}_1,i}$ is different. And having in mind the definition of the rotation values in Table 2.3 we can conclude that there are no neighbors in the rightmost positions of differences in every $\Delta_j, j \in R_{\mathbb{A}_1,i}$.

Finally the transformation $\mathbb{A}_2$ is applied on $\mathbf{\Delta_2}$ and we have

$$\mathbf{\Delta_3} = \mathbb{A}_2(\mathbf{\Delta_2}) = (\delta_0^{(3)}, \delta_1^{(3)}, \ldots, \delta_7^{(3)}),$$

where

$$\delta_j^{(3)} = \delta_{\mu_1}^{(2)} \oplus \delta_{\mu_2}^{(2)} \oplus \delta_{\mu_3}^{(2)},\ \mu_1, \mu_2, \mu_3 \in L_{1,2}^{(j)}.$$

Bearing in mind that $Hamming(Y_j, Y_j') = |\delta_j^{(3)}|,\ j = 0, \ldots, 7$ the conclusion that

$$Hamming(Y_j, Y_j') \geq d_{i,j},\ j = 0, \ldots, 7$$

where $d_{i,j}$ are values in the matrix $\mathbf{Diff}_{\pi_2}$ follows directly. □

Lemma 3. Let X and X' be two w-bit variables with a Hamming distance of two bits. If the two difference bits of X and X' are not neighboring bits, then for all w-bit constants C, $Hamming(X + C, X' + C) \geq 2$ where the operation $+$ denotes addition modulo 2^w.

Proof. It is enough to exhaustively search all situations when the length of the word is $w = 4$. In that case, when the two difference bits of X and X' are not neighboring bits, the relation

$$Hamming(X + C, X' + C) \geq 2$$

always holds for all 4-bit values of C. For the bigger values of w, we can always treat the 4-bit cases as an included substring. □

We will need the conclusions from Lemma 3 for proving the following properties of the differential characteristics of the quasigroup operation $*_q$.

Corollary 1. Let $\mathbf{X}, \mathbf{X}', \mathbf{Y} \in Q_q$ be represented as $\mathbf{X} = (X_0, \ldots, X_7)$, $\mathbf{X}' = (X_0', \ldots, X_7')$, $\mathbf{Y} = (Y_0, \ldots, Y_7)$ and let $\mathbf{Z} = \mathbf{X} *_q \mathbf{Y}$, $\mathbf{Z}' = \mathbf{X}' *_q \mathbf{Y}$. If $\mathbf{X}$ and $\mathbf{X}'$ differ in one bit, i.e. the Hamming distance $Hamming(\mathbf{X}, \mathbf{X}') = 1$, and if that one-bit difference is in the i-th word i.e. $Hamming(X_i, X_i') = 1,\ i = 0, \ldots, 7$, then

$$Hamming(Z_j, Z_j') \geq d_{i,j},\ j = 0, \ldots, 7$$

where $d_{i,j}$ are values in the matrix $\mathbf{Diff}_{\pi_2}$.

Proof. (sketch) The proof follows from the definition of the quasigroup operation

$$\mathbf{X} *_q \mathbf{Y} = \pi_1(\pi_2(\mathbf{X})\ +_8\ \pi_3(\mathbf{Y})),$$

Theorem 2, the fact that the minimal difference among any two values in the rotation vectors $\mathbf{r}_{1,q}$ and $\mathbf{r}_{2,q}$ is bigger than 2 and Lemma 3. □

Corollary 2. Let $\mathbf{X}, \mathbf{Y}, \mathbf{Y}' \in Q_q$ be represented as $\mathbf{X} = (X_0, \ldots, X_7)$, $\mathbf{Y} = (Y_0, \ldots, Y_7)$, $\mathbf{Y}' = (Y_0', \ldots, Y_7')$, and let $\mathbf{Z} = \mathbf{X} *_q \mathbf{Y}$, $\mathbf{Z}' = \mathbf{X} *_q \mathbf{Y}'$. If $\mathbf{Y}$ and $\mathbf{Y}'$ differ in one bit, i.e. the Hamming distance $Hamming(\mathbf{Y}, \mathbf{Y}') = 1$, and if that one-bit difference is in the i-th word i.e. $Hamming(Y_i, Y_i') = 1,\ i = 0, \ldots, 7$, then

$$Hamming(Z_j, Z_j') \geq d_{i,j},\ j = 0, \ldots, 7$$

where $d_{i,j}$ are values in the matrix $\mathbf{Diff}_{\pi_3}$. □

Definition 12. Let $\mathbf{X}, \mathbf{X}', \mathbf{Y}, \mathbf{Y}' \in Q_q$ and let $\mathbf{\Delta_X} = \mathbf{X} \oplus \mathbf{X}'$ and $\mathbf{\Delta_Y} = \mathbf{Y} \oplus \mathbf{Y}'$ be two difference vectors. Let $\mathbf{Z} = \mathbf{X} *_q \mathbf{Y}$ and $\mathbf{Z}' = \mathbf{X}' *_q \mathbf{Y}'$. The vector $\mathcal{D}_{(\mathbf{\Delta_X}, \mathbf{\Delta_Y})} = (\delta_0, \ldots, \delta_7) \in \big(\mathbb{Z}\big)^8$ is called *bit flip counter for the quasigroup operation* $*_q$, if every $\delta_i,\ i = 0, \ldots, 7$ is a counter of the minimal number of bit flips that the quasigroup operation $*_q$ performs to transfer the value $\mathbf{Z}$ to the value $\mathbf{Z}'$.

For the $\mathcal{D}_{(\mathbf{\Delta_X}, \mathbf{\Delta_Y})}$ we have the following Theorem:

Theorem 3.

$$\mathcal{D}_{(\mathbf{\Delta_X}, \mathbf{\Delta_Y})} = \mathbf{Diff}_{\pi_2} \cdot \mathbf{\Delta_X} + \mathbf{Diff}_{\pi_3} \cdot \mathbf{\Delta_Y}.$$

Proof. (Sketch) We just need to represent $\mathbf{\Delta_X}$ and $\mathbf{\Delta_Y}$ as a sum of one-bit difference vectors and apply Theorem 2, Corollary 1 and Corollary 2. □

For given constant values $\mathbf{C}_0$ and $\mathbf{C}_1$ let us define the intermediate values $\mathbf{D}_i$ obtained by the function $\mathcal{R}(\mathbf{C}_0, \mathbf{C}_1, \mathbf{X}, \mathbf{Y})$ (as they are represented in Table 3.4a). If we have two differentials $\mathbf{\Delta_X}$ and $\mathbf{\Delta_Y}$ in order to trace all differentials for $\mathbf{D}_i$ instead of the complex notation $\mathcal{D}_{(\mathbf{\Delta_X}, \mathbf{\Delta_Y})}$ we will use the notation $\mathcal{D}_i$ for the corresponding $\mathbf{D}_i$ and the initial differentials $\mathbf{\Delta_X}$ and $\mathbf{\Delta_Y}$.

The relations between $\mathbf{\Delta_X}$, $\mathbf{\Delta_Y}$ and $\mathcal{D}_i,\ i = 1, \ldots, 8$ are shown in the Table 3.4 and in Table 3.5.

In Table 3.6 we give an example for the difference vectors: $\mathbf{\Delta_X} = (1, 0, 0, 0, 0, 0, 0, 0)$ and $\mathbf{\Delta_Y} = (0, 0, 0, 0, 0, 0, 0, 0)$, while in the Table 3.7 we give an example for the difference vectors: $\mathbf{\Delta_X} = (0, 0, 0, 0, 0, 0, 0, 0)$ and $\mathbf{\Delta_Y} = (1, 0, 0, 0, 0, 0, 0, 0)$.

Notice the small variance between the values of the vectors $\mathcal{D}_i,\ i = 1, \ldots, 8$ in Table 3.6 and Table 3.7. That is not by accident. Actually it is a consequence of the basic property of the diffusion matrices $\mathbf{Diff}_{\pi_2}$ and $\mathbf{Diff}_{\pi_3}$ and that property is one of the most important properties that guarantee that Edon-$\mathcal{R}$ is resistant against differential cryptanalysis attacks. That property is proved in the following subsection.

	$\mathbf{X}$	$\mathbf{Y}$
$\overline{\mathbf{Y}}$	$\mathbf{D}_1$	$\mathbf{D}_2$
$\mathbf{C}_0$	$\mathbf{D}_3$	$\mathbf{D}_4$
$\mathbf{C}_1$	$\mathbf{D}_5$	$\mathbf{D}_6$
$\overline{\mathbf{X}}$	$\mathbf{D}_7$	$\mathbf{D}_8$

a.

	$\boldsymbol{\Delta}_{\mathbf{X}}$	$\boldsymbol{\Delta}_{\mathbf{Y}}$
$\overline{\boldsymbol{\Delta}_{\mathbf{Y}}}$	$\mathcal{D}_1 = \mathbf{Diff}_{\pi_2} \cdot \overline{\boldsymbol{\Delta}_{\mathbf{Y}}} + \mathbf{Diff}_{\pi_3} \cdot \boldsymbol{\Delta}_{\mathbf{X}}$	$\mathcal{D}_2 = \mathbf{Diff}_{\pi_2} \cdot \mathcal{D}_1 + \mathbf{Diff}_{\pi_3} \cdot \boldsymbol{\Delta}_{\mathbf{Y}}$
$\mathbf{0}$	$\mathcal{D}_3 = \mathbf{Diff}_{\pi_2} \cdot \mathbf{0} + \mathbf{Diff}_{\pi_3} \cdot \mathcal{D}_1$	$\mathcal{D}_4 = \mathbf{Diff}_{\pi_2} \cdot \mathcal{D}_3 + \mathbf{Diff}_{\pi_3} \cdot \mathcal{D}_2$
$\mathbf{0}$	$\mathcal{D}_5 = \mathbf{Diff}_{\pi_2} \cdot \mathcal{D}_3 + \mathbf{Diff}_{\pi_3} \cdot \mathbf{0}$	$\mathcal{D}_6 = \mathbf{Diff}_{\pi_2} \cdot \mathcal{D}_4 + \mathbf{Diff}_{\pi_3} \cdot \mathcal{D}_5$
$\overline{\boldsymbol{\Delta}_{\mathbf{X}}}$	$\mathcal{D}_7 = \mathbf{Diff}_{\pi_2} \cdot \overline{\boldsymbol{\Delta}_{\mathbf{X}}} + \mathbf{Diff}_{\pi_3} \cdot \mathcal{D}_5$	$\mathcal{D}_8 = \mathbf{Diff}_{\pi_2} \cdot \mathcal{D}_7 + \mathbf{Diff}_{\pi_3} \cdot \mathcal{D}_6$

b.

Table 3.4: **a.** General scheme for computing the one-way function $\mathcal{R}(\mathbf{C}_0, \mathbf{C}_1, \mathbf{X}, \mathbf{Y})$ and the intermediate values $\mathbf{D}_i$, $i = 1, \ldots, 8$. **b.** Counting the minimal number of bit flips $\mathcal{D}_i$, $i = 1, \ldots, 8$ when applying the quasigroup operation $*_q$ on $\mathbf{X}$ and $\mathbf{Y}$ that differ by difference vectors $\boldsymbol{\Delta}_{\mathbf{X}}$ and $\boldsymbol{\Delta}_{\mathbf{Y}}$. Corresponding difference vectors for the fixed values $\mathbf{C}_0$ and $\mathbf{C}_1$ are the zero vector $\mathbf{0}$.

$$\mathcal{D}_1 = \mathbf{Diff}_{\pi_2} \cdot \overline{\boldsymbol{\Delta}_{\mathbf{Y}}} + \mathbf{Diff}_{\pi_3} \cdot \boldsymbol{\Delta}_{\mathbf{X}}$$
$$\mathcal{D}_2 = \mathbf{Diff}_{\pi_2} \cdot (\mathbf{Diff}_{\pi_2} \cdot \overline{\boldsymbol{\Delta}_{\mathbf{Y}}} + \mathbf{Diff}_{\pi_3} \cdot \boldsymbol{\Delta}_{\mathbf{X}}) + \mathbf{Diff}_{\pi_3} \cdot \boldsymbol{\Delta}_{\mathbf{Y}}$$

$$\mathcal{D}_3 = \mathbf{Diff}_{\pi_3} \cdot (\mathbf{Diff}_{\pi_2} \cdot \overline{\boldsymbol{\Delta}_{\mathbf{Y}}} + \mathbf{Diff}_{\pi_3} \cdot \boldsymbol{\Delta}_{\mathbf{X}})$$
$$\mathcal{D}_4 = \mathbf{Diff}_{\pi_2} \cdot \Big(\mathbf{Diff}_{\pi_3} \cdot (\mathbf{Diff}_{\pi_2} \cdot \overline{\boldsymbol{\Delta}_{\mathbf{Y}}} + \mathbf{Diff}_{\pi_3} \cdot \boldsymbol{\Delta}_{\mathbf{X}})\Big) + \mathbf{Diff}_{\pi_3} \cdot \Big(\mathbf{Diff}_{\pi_2} \cdot (\mathbf{Diff}_{\pi_2} \cdot \overline{\boldsymbol{\Delta}_{\mathbf{Y}}} + \mathbf{Diff}_{\pi_3} \cdot \boldsymbol{\Delta}_{\mathbf{X}}) + \mathbf{Diff}_{\pi_3} \cdot \boldsymbol{\Delta}_{\mathbf{Y}}\Big)$$

$$\mathcal{D}_5 = \mathbf{Diff}_{\pi_2} \cdot \Big(\mathbf{Diff}_{\pi_3} \cdot (\mathbf{Diff}_{\pi_2} \cdot \overline{\boldsymbol{\Delta}_{\mathbf{Y}}} + \mathbf{Diff}_{\pi_3} \cdot \boldsymbol{\Delta}_{\mathbf{X}})\Big)$$
$$\mathcal{D}_6 = \mathbf{Diff}_{\pi_2} \cdot \bigg(\mathbf{Diff}_{\pi_2} \cdot \Big(\mathbf{Diff}_{\pi_3} \cdot (\mathbf{Diff}_{\pi_2} \cdot \overline{\boldsymbol{\Delta}_{\mathbf{Y}}} + \mathbf{Diff}_{\pi_3} \cdot \boldsymbol{\Delta}_{\mathbf{X}})\Big) + \mathbf{Diff}_{\pi_3} \cdot \Big(\mathbf{Diff}_{\pi_2} \cdot (\mathbf{Diff}_{\pi_2} \cdot \overline{\boldsymbol{\Delta}_{\mathbf{Y}}} + \mathbf{Diff}_{\pi_3} \cdot \boldsymbol{\Delta}_{\mathbf{X}}) + \mathbf{Diff}_{\pi_3} \cdot \boldsymbol{\Delta}_{\mathbf{Y}}\Big)\bigg) +$$
$$\mathbf{Diff}_{\pi_3} \cdot \bigg(\mathbf{Diff}_{\pi_2} \cdot \Big(\mathbf{Diff}_{\pi_3} \cdot (\mathbf{Diff}_{\pi_2} \cdot \overline{\boldsymbol{\Delta}_{\mathbf{Y}}} + \mathbf{Diff}_{\pi_3} \cdot \boldsymbol{\Delta}_{\mathbf{X}})\Big)\bigg)$$

$$\mathcal{D}_7 = \mathbf{Diff}_{\pi_2} \cdot \overline{\boldsymbol{\Delta}_{\mathbf{X}}} + \mathbf{Diff}_{\pi_3} \cdot \bigg(\mathbf{Diff}_{\pi_2} \cdot \Big(\mathbf{Diff}_{\pi_3} \cdot (\mathbf{Diff}_{\pi_2} \cdot \overline{\boldsymbol{\Delta}_{\mathbf{Y}}} + \mathbf{Diff}_{\pi_3} \cdot \boldsymbol{\Delta}_{\mathbf{X}})\Big)\bigg)$$
$$\mathcal{D}_8 = \mathbf{Diff}_{\pi_2} \cdot \Bigg(\mathbf{Diff}_{\pi_2} \cdot \overline{\boldsymbol{\Delta}_{\mathbf{X}}} + \mathbf{Diff}_{\pi_3} \cdot \bigg(\mathbf{Diff}_{\pi_2} \cdot \Big(\mathbf{Diff}_{\pi_3} \cdot (\mathbf{Diff}_{\pi_2} \cdot \overline{\boldsymbol{\Delta}_{\mathbf{Y}}} + \mathbf{Diff}_{\pi_3} \cdot \boldsymbol{\Delta}_{\mathbf{X}})\Big)\bigg)\Bigg) + \mathbf{Diff}_{\pi_3} \cdot \Bigg(\mathbf{Diff}_{\pi_2} \cdot \bigg(\mathbf{Diff}_{\pi_2} \cdot \Big(\mathbf{Diff}_{\pi_3} \cdot$$
$$(\mathbf{Diff}_{\pi_2} \cdot \overline{\boldsymbol{\Delta}_{\mathbf{Y}}} + \mathbf{Diff}_{\pi_3} \cdot \boldsymbol{\Delta}_{\mathbf{X}})\Big) + \mathbf{Diff}_{\pi_3} \cdot \Big(\mathbf{Diff}_{\pi_2} \cdot (\mathbf{Diff}_{\pi_2} \cdot \overline{\boldsymbol{\Delta}_{\mathbf{Y}}} + \mathbf{Diff}_{\pi_3} \cdot \boldsymbol{\Delta}_{\mathbf{X}}) + \mathbf{Diff}_{\pi_3} \cdot \boldsymbol{\Delta}_{\mathbf{Y}}\Big)\bigg) + \mathbf{Diff}_{\pi_3} \cdot \bigg(\mathbf{Diff}_{\pi_2} \cdot \Big(\mathbf{Diff}_{\pi_3} \cdot$$
$$(\mathbf{Diff}_{\pi_2} \cdot \overline{\boldsymbol{\Delta}_{\mathbf{Y}}} + \mathbf{Diff}_{\pi_3} \cdot \boldsymbol{\Delta}_{\mathbf{X}})\Big)\bigg)\Bigg)$$

Table 3.5: The relations between the vectors of minimal number of bit flips for the function $\mathcal{R}$ when the initial difference vectors are $\boldsymbol{\Delta}_{\mathbf{X}}$ and $\boldsymbol{\Delta}_{\mathbf{Y}}$.

	$\mathbf{\Delta_X} = (1,0,0,0,0,0,0,0)$	$\mathbf{\Delta_Y} = (0,0,0,0,0,0,0,0)$
$\overline{\mathbf{\Delta_Y}} = (0,0,0,0,0,0,0,0)$	(1, 2, 2, 2, 2, 2, 2, 2)	(28, 29, 28, 28, 29, 27, 28, 28)
$\mathbf{0}$	(29, 28, 28, 28, 28, 28, 28, 28)	(844, 842, 844, 844, 842, 846, 844, 844)
$\mathbf{0}$	(422, 421, 422, 422, 421, 423, 422, 422)	(18984, 18985, 18982, 18986, 18985, 18983, 18984, 18986)
$\overline{\mathbf{\Delta_X}} = (0,0,0,0,0,0,0,1)$	(6330, 6331, 6330, 6330, 6332, 6328, 6329, 6330)	(379716, 379715, 379721, 379713, 379716, 379717, 379715, 379712)

Table 3.6: Vectors of minimal number of bit flips for the function $\mathcal{R}$ when the initial difference vectors are $\mathbf{\Delta_X} = (1,0,0,0,0,0,0,0)$ and $\mathbf{\Delta_Y} = (0,0,0,0,0,0,0,0)$.

	$\mathbf{\Delta_X} = (0,0,0,0,0,0,0,0)$	$\mathbf{\Delta_Y} = (1,0,0,0,0,0,0,0)$
$\overline{\mathbf{\Delta_Y}} = (0,0,0,0,0,0,0,1)$	(2, 2, 2, 2, 3, 1, 1, 2)	(29, 30, 32, 30, 31, 30, 29, 29)
$\mathbf{0}$	(28, 28, 28, 28, 27, 29, 29, 28)	(873, 872, 868, 872, 870, 872, 874, 874)
$\mathbf{0}$	(422, 422, 420, 422, 421, 422, 423, 423)	(19406, 19409, 19406, 19406, 19406, 19405, 19405, 19407)
$\overline{\mathbf{\Delta_X}} = (0,0,0,0,0,0,0,0)$	(6328, 6328, 6330, 6328, 6329, 6328, 6327, 6327)	(386016, 386011, 386017, 386016, 386016, 386018, 386017, 386014)

Table 3.7: Vectors of minimal number of bit flips for the function $\mathcal{R}$ when the initial difference vectors are $\mathbf{\Delta_X} = (0,0,0,0,0,0,0,0)$ and $\mathbf{\Delta_Y} = (1,0,0,0,0,0,0,0)$.

3.5.1 The variance of the elements of $\mathcal{D}_i$

Let $\lambda_1, \lambda_2, \ldots, \lambda_n$ be the eigenvalues of some matrix $\mathbb{M}$, arranged such that $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$, where $|x+iy| = \sqrt{x^2+y^2}$. The eigenvector corresponding to the eigenvalue λ_i is denoted by $\mathbf{s}_i = [s_{1i}, s_{2i}, \ldots, s_{ni}]^T$. Let $\mathbb{S}$ be the matrix formed by letting the i-th column of $\mathbb{S}$ be equal to the i-th eigenvector of $\mathbb{M}$, i.e. $\mathbb{S} = [\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n]$. Let the i-th vector-row of the matrix $\mathbb{S}^{-1}$ be denoted by $\mathbf{s}'_i = [s'_{1i}, s'_{2i}, \ldots, s'_{ni}]^T$, such that $(\mathbb{S}^{-1})^T = [\mathbf{s}'_1, \mathbf{s}'_2, \ldots, \mathbf{s}'_n]$. Let $\mathbf{\Lambda}^n$ be defined as a diagonal matrix with $\mathbf{\Lambda}_{ii} = \lambda_i$. We know from linear algebra that the matrix $\mathbb{M}$ can be written as $\mathbb{M} = \mathbb{S}\mathbf{\Lambda}\mathbb{S}^{-1}$. This gives us the following,

$$\mathbb{M}^n = \mathbb{S}\mathbf{\Lambda}^n\mathbb{S}^{-1} = \mathbb{S}\sum_{i=1}^{n}(\mathbf{\Lambda}^{(i)})^n\mathbb{S}^{-1} = \sum_{i=1}^{n}\lambda_i^n \begin{bmatrix} s_{1i}\mathbf{s}'_i \\ s_{2i}\mathbf{s}'_i \\ \ldots \\ s_{ni}\mathbf{s}'_i \end{bmatrix}, \tag{3.5.1}$$

where $\mathbf{\Lambda}^{(i)}$ is a square matrix with all elements equal to 0, except $\mathbf{\Lambda}^{(i)}_{ii}$ which is equal to λ_i, i.e.

$$\mathbf{\Lambda}^{(i)}_{jk} = \begin{cases} \lambda_i, & i = j = k \\ 0, & other \end{cases}$$

Proposition 2. Let $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$ be the eigenvalues of $\mathbb{M}$. If the vector $\mathbb{1} = (1, 1, \ldots, 1)$

is the eigenvector for the greatest eigenvalue, λ_1. Then for each vector $\mathbf{a}$

$$\lim_{n\to\infty} \frac{(\mathbb{M}^n \mathbf{a})_i}{\lambda_1^n} = (\mathbf{s}_1')^T \mathbf{a},\ \forall i = 1, \dots, n, \tag{3.5.2}$$

Proof. From (3.5.1) we have,

$$\mathbb{M}^n \mathbf{a} = \sum_{i=1}^{n} \lambda_i^n \begin{bmatrix} s_{1i}\mathbf{s}_i' \\ s_{2i}\mathbf{s}_i' \\ \dots \\ s_{ni}\mathbf{v}_i' \end{bmatrix} \mathbf{a} = \lambda_1^n \begin{bmatrix} \mathbf{s}_1' \\ \mathbf{s}_1' \\ \dots \\ \mathbf{s}_1' \end{bmatrix} \mathbf{a} + \sum_{i=2}^{n} \lambda_i^n \begin{bmatrix} s_{1i}\mathbf{s}_i' \\ s_{2i}\mathbf{s}_i' \\ \dots \\ s_{ni}\mathbf{s}_i' \end{bmatrix} \mathbf{a} = \lambda_1^n \begin{bmatrix} \mathbf{s}_1'\mathbf{a} \\ \mathbf{s}_1'\mathbf{a} \\ \dots \\ \mathbf{s}_1'\mathbf{a} \end{bmatrix} + \sum_{i=2}^{n} \lambda_i^n \begin{bmatrix} s_{1i}\mathbf{s}_i'\mathbf{a} \\ s_{2i}\mathbf{s}_i'\mathbf{a} \\ \dots \\ s_{ni}\mathbf{s}_i'\mathbf{a} \end{bmatrix}.$$

Directly from this holds (3.5.2). □

Proposition 3. Let $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$ be the eigenvalues of $\mathbb{M}$. If the vector $\mathbb{1}$ is the eigenvector for the greatest eigenvalue, λ_1 and $|\lambda_1| > |(\lambda_2)^2|$. Then for each vector $\mathbf{a}$

$$\lim_{n\to\infty} \frac{Var(\mathbb{M}^n \mathbf{a})}{\min\limits_i (\mathbb{M}^n \mathbf{a})_i} = 0. \tag{3.5.3}$$

Proof. Let $b_j^{(n)} = (\mathbb{M}^n \mathbf{a})_j$. Then from (3.5.1) we have that

$$b_j^{(n)} = \lambda_1^n \mathbf{s}_1' \mathbf{a} + \sum_{i=2}^{n} \lambda_i^n s_{ji} \mathbf{s}_i' \mathbf{a},$$

and

$$\begin{aligned} Avr(b^{(n)}) &= \frac{1}{n}\sum_{j=1}^{n} b_j^{(n)} = \lambda_1^n \mathbf{s}_1'\mathbf{a} + \frac{1}{n}\sum_{j=1}^{n}\sum_{i=2}^{n} \lambda_i^n s_{ji}\mathbf{s}_i'\mathbf{a} = \lambda_1^n \mathbf{s}_1'\mathbf{a} + \sum_{i=2}^{n} \lambda_i^n \left(\frac{1}{n}\sum_{j=1}^{n} s_{ji}\right) \mathbf{s}_i'\mathbf{a} \\ &= \lambda_1^n \mathbf{s}_1'\mathbf{a} + \sum_{i=2}^{n} \lambda_i^n Avr(s_i)\mathbf{s}_i'\mathbf{a}. \end{aligned}$$

Now,

$$b_j^{(n)} - Avr(b^{(n)}) = \sum_{i=2}^{n} \lambda_i^n s_{ji}\mathbf{s}_i'\mathbf{a} - \sum_{i=2}^{n} \lambda_i^n Avr(s_i)\mathbf{s}_i'\mathbf{a} = \sum_{i=2}^{n} \lambda_i^n (s_{ji} - Avr(s_i))\mathbf{s}_i'\mathbf{a}.$$

From this and the Proposition2 we have that

$$\begin{aligned} \lim_{n\to\infty} \frac{Var(\mathbb{M}^n \mathbf{a})}{\min\limits_i (\mathbb{M}^n \mathbf{a})_i} &= \lim_{n\to\infty} \frac{\dfrac{Var(\mathbb{M}^n \mathbf{a})}{\lambda_1^n}}{\dfrac{\min_i (\mathbb{M}^n \mathbf{a})_i}{\lambda_1^n}} = \lim_{n\to\infty} \frac{\dfrac{\sum_{j=1}^n \left(\sum_{i=2}^n \lambda_i^n (s_{ji} - Avr(s_i))\mathbf{s}_i'\mathbf{a}\right)^2}{\lambda_1^n}}{\dfrac{\min_i (\mathbb{M}^n \mathbf{a})_i}{\lambda_1^n}} \\ &= \frac{1}{\mathbf{s}_1'\mathbf{a}} \lim_{n\to\infty} \frac{\sum_{j=1}^n \left(\sum_{i=2}^n \lambda_i^n (s_{ji} - Avr(s_i))\mathbf{s}_i'\mathbf{a}\right)^2}{\lambda_1^n} = \frac{1}{\mathbf{s}_1'\mathbf{a}} \cdot 0 = 0. \end{aligned}$$

□

Proposition 4. Let $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$ be the eigenvectors either for $\mathbf{Diff}_{\pi_2}$ or for $\mathbf{Diff}_{\pi_3}$. Then the vector $\mathbb{1}$ is the eigenvector for the greatest eigenvalue, λ_1 and moreover $|\lambda_1| > |\lambda_2^2|$.

Proof. The absolute values of the eigenvalues with their corresponding eigenvectors are given in Table 3.3. □

Finally, from all previous claims in this subsection the following theorem follows:

Theorem 4. The variance of the elements of the $\mathcal{D}_i,\ i = 1, \ldots, 8$ decreases (relative to the minimal element in the vectors $\mathcal{D}_i,\ i = 1, \ldots, 8$), with every row of quasigroup string transformations in the compression function $\mathcal{R}$. □

3.5.2 Differential characteristics of the compression function $\mathcal{R}$

As the values $\delta_j^{(i)},\ j = 0, \ldots, 7$ in every vector of minimal number of bit flips $\mathcal{D}_i = (\delta_0^{(i)}, \delta_1^{(i)}, \ldots, \delta_7^{(i)})$, $i = 1, \ldots, 8$ tend to have very small variance, we have reasons to assume that the number of bit flips for every single bit is also equally distributed i.e. with very small variance. Having this assumption, we can prove the following theorem:

Theorem 5. Let $\mathcal{D}_i = (\delta_0^{(i)}, \delta_1^{(i)}, \ldots, \delta_7^{(i)}),\ i = 1, \ldots, 8$ be a vector of minimal number of bit flips for the function $\mathcal{R}$ where the size of the word is w bits ($w = 32, 64$), and let $\mathbf{\Delta_{D_i}} = (\Delta_{D_0}^{(i)}, \Delta_{D_1}^{(i)}, \ldots, \Delta_{D_7}^{(i)}) = (\Delta_0^{(i)}, \ldots, \Delta_{w-1}^{(i)}, \Delta_w^{(i)}, \ldots, \Delta_{2w-1}^{(i)}, \Delta_{2w}^{(i)}, \ldots, \ldots, \Delta_{7w-1}^{(i)}, \Delta_{7w}^{(i)}, \ldots, \Delta_{8w-1}^{(i)}),\ i = 1, \ldots, 8$ (where $\Delta_j^{(i)} \in \{0, 1\}$, $j = 0, \ldots, 8w - 1$) are the corresponding differentials in the intermediate variables $\mathbf{\Delta_{D_i}}$ for some initially chosen differentials $\mathbf{\Delta_X}$ and $\mathbf{\Delta_Y}$ (where at least one of them is a non-zero differential). If the number of bit flips for every single bit is equally distributed then the probabilities that every difference bit $\Delta_j^{(i)}$ is 0 or 1 are given as:

$$\begin{aligned} Pr\big(\Delta_j^{(i)} &= 0|\mathbf{\Delta_X}, \mathbf{\Delta_Y}\big) = 0.5 + \epsilon_{\delta_\mu^{(i)}}, \\ Pr\big(\Delta_j^{(i)} &= 1|\mathbf{\Delta_X}, \mathbf{\Delta_Y}\big) = 0.5 - \epsilon_{\delta_\mu^{(i)}}, \end{aligned}$$

where $\mu = \left\lfloor \frac{j}{w} \right\rfloor$ and $\epsilon_{\delta_\mu^{(i)}} \leq 0.5 \left(\frac{w-2}{w}\right)^{\delta_\mu^{(i)}}$.

Proof. From the conditions of the Theorem we have that the minimal number of bit flips for the $\Delta_{D_\mu}^{(i)}$ is $\delta_\mu^{(i)}$ where $\mu = 0, \ldots, 7$. Note that $\Delta_{D_\mu}^{(i)}$ is a w-bit word. The probability that the value of any difference bit $\Delta_j^{(i)}$ is equal to 0 is the probability that the number of bit flips for that particular bit is even. Taking into the consideration the assumption that the number of bit flips for every bit in

$\Delta^{(i)}_{D_\mu}$ is equally distributed, we can conclude that in one experiment the probability that the bit is flipped is $\frac{1}{w}$ and the probability that it is not flipped is $\left(1-\frac{1}{w}\right)$. Then if we have $\delta^{(i)}_\mu$ experiments, the probability that the number of bit flips for that particular bit is even can be computed as:

$$Pr\big(\Delta^{(i)}_j = 0\big) = \sum_{\substack{r=0\\ r \text{ is even}}}^{\delta^{(i)}_\mu} \binom{\delta^{(i)}_\mu}{r} \left(\frac{1}{w}\right)^r \left(1-\frac{1}{w}\right)^{\delta^{(i)}_\mu - r}.$$

Similarly we can compute the probability:

$$Pr\big(\Delta^{(i)}_j = 1\big) = \sum_{\substack{r=1\\ r \text{ is odd}}}^{\delta^{(i)}_\mu} \binom{\delta^{(i)}_\mu}{r} \left(\frac{1}{w}\right)^r \left(1-\frac{1}{w}\right)^{\delta^{(i)}_\mu - r}.$$

Now if take into account that

$$Pr\big(\Delta^{(i)}_j = 0\big) + Pr\big(\Delta^{(i)}_j = 1\big) = 1$$

and

$$\lim_{\delta^{(i)}_\mu \to \infty} \left|Pr\big(\Delta^{(i)}_j = 0\big) - Pr\big(\Delta^{(i)}_j = 1\big)\right| = 0$$

then we can rewrite the last limit as:

$$\begin{array}{l} Pr\big(\Delta^{(i)}_j = 0 | \mathbf{\Delta_X}, \mathbf{\Delta_Y}\big) = 0.5 + \epsilon_{0,\delta^{(i)}_\mu}, \\ Pr\big(\Delta^{(i)}_j = 1 | \mathbf{\Delta_X}, \mathbf{\Delta_Y}\big) = 0.5 - \epsilon_{1,\delta^{(i)}_\mu} \end{array}.$$

Finding explicit expressions for $\epsilon_{0,\delta^{(i)}_\mu}$ and $\epsilon_{1,\delta^{(i)}_\mu}$ is hard, but having concrete numerical values $\delta^{(i)}_\mu$ we can compute them as:

$$\epsilon_{0,\delta^{(i)}_\mu} = \sum_{\substack{r=0\\ r \text{ is even}}}^{\delta^{(i)}_\mu} \binom{\delta^{(i)}_\mu}{r} \left(\frac{1}{w}\right)^r \left(1-\frac{1}{w}\right)^{\delta^{(i)}_\mu - r} - 0.5,$$

$$\epsilon_{1,\delta^{(i)}_\mu} = 0.5 - \sum_{\substack{r=1\\ r \text{ is odd}}}^{\delta^{(i)}_\mu} \binom{\delta^{(i)}_\mu}{r} \left(\frac{1}{w}\right)^r \left(1-\frac{1}{w}\right)^{\delta^{(i)}_\mu - r}.$$

or we can consider them as approximately the same value that is upper bounded by:

$$\epsilon_{\delta^{(i)}_\mu} \approx \epsilon_{0,\delta^{(i)}_\mu} \approx \epsilon_{1,\delta^{(i)}_\mu} \leq 0.5 \left(\frac{w-2}{w}\right)^{\delta^{(i)}_\mu}.$$

□

$\mathbf{\Delta_X} = (1,0,0,0,0,0,0,0)$		$\mathbf{\Delta_Y} = (0,0,0,0,0,0,0,0)$	
$w = 32$	$w = 64$	$w = 32$	$w = 64$
$\epsilon \leq 2^{-1.09}$	$\epsilon \leq 2^{-1.05}$	$\epsilon \leq 2^{-3.51}$	$\epsilon \leq 2^{-2.24}$
$\epsilon \leq 2^{-3.61}$	$\epsilon \leq 2^{-2.28}$	$\epsilon \leq 2^{-79.40}$	$\epsilon \leq 2^{-39.57}$
$\epsilon \leq 2^{-40.20}$	$\epsilon \leq 2^{-20.28}$	$\epsilon \leq 2^{-1768.4}$	$\epsilon \leq 2^{-870.45}$
$\epsilon \leq 2^{-590.20}$	$\epsilon \leq 2^{-290.85}$	$\epsilon \leq 2^{-35356}$	$\epsilon \leq 2^{-17393}$

Table 3.8: Upper bounds for the deviations ϵ. The probability that a bit will have a differential $\Delta = 1$ is $0.5 - \epsilon$, and the probability that a bit will have a differential $\Delta = 0$ is $0.5 + \epsilon$. The initial difference vectors are $\mathbf{\Delta_X} = (1,0,0,0,0,0,0,0)$ and $\mathbf{\Delta_Y} = (0,0,0,0,0,0,0,0)$.

For one of the smallest differentials $(\mathbf{\Delta_X}, \mathbf{\Delta_Y}) = \big((1,0,0,0,0,0,0,0), (0,0,0,0,0,0,0,0)\big)$ the values for the corresponding $\epsilon_{\delta_\mu^{(i)}}$ both for $w = 32$ and $w = 64$ are given in Table 3.8. The values for the other one-bit differences are similar, and the values of $\epsilon_{\delta_\mu^{(i)}}$ for the differentials with initial difference in more then one bit are even smaller.

We consider that Theorem 5 is the proof of the EDON-$\mathcal{R}$'s resistance against differential cryptanalysis.

3.6 Criteria for choosing the Latin squares - part two

Having described in detail the differential characteristics of the defined quasigroup operations $*_q$, we can describe the reasons and the criteria by which we have chosen the Latin squares L_1 and L_2 by which we are defining the quasigroup operation $*_q$. The criteria are descibed in Table 3.9.

For complying with the first criterion we took all 2165 main classes of orthogonal Latin squares of order 8 that are listed on Brendan McKay's web page [20]. For every one of them, first by permuting their rows and then their columns we produced $(8!)^2 \approx 2^{30.6}$ orthogonal isotopes. Permutations were ordered by the lexicographic ordering. Next, we filtered that number of orthogonal Latin squares by the Criterion 2: Latin squares that give diffusion matrices $\mathbf{Diff}_{\pi_2}$ and $\mathbf{Diff}_{\pi_3}$ that do not have zeroes. We further filtered the number of Latin squares by selecting those Latin squares that have a maximum variance computed on all 64 elements of the matrix $\mathbf{Diff}_{\pi_2}$ and minimum variance computed on all 64 elements of the matrix $\mathbf{Diff}_{\pi_2}$ (Criterion 3).

By exhaustive search we found that Latin squares that comply with all 4 criteria give matrices $\mathbf{Diff}_{\pi_2}$ with maximum variance of $\frac{19}{63}$ and matrices $\mathbf{Diff}_{\pi_3}$ with minimum variance of $\frac{1}{9}$. The first such pair of Latin squares was chosen for EDON-$\mathcal{R}$.

3.7 On some properties of the matrices $\mathbb{A}_i$

As a part of our cryptanalysis of the EDON-$\mathcal{R}$ hash function we present here several interesting mathematical properties for the matrices $\mathbb{A}_i$ defined in Section 2.1.2 and in the Table 2.4, when those matrices are considered as matrices with elements in $\mathbb{Z}_2$. We want to stress here that we are not aware that any of these properties can be used for launching some concrete attack on the hash function EDON-$\mathcal{R}$.

From the definition of matrices $\mathbb{A}_i$, it is obvious that

$$\mathbb{A}_1 + \mathbb{A}_2 = \mathbb{1} \tag{3.7.1}$$

and

$$\mathbb{A}_3 + \mathbb{A}_4 = \mathbb{1} \tag{3.7.2}$$

where $\mathbb{1}$ is a square 8×8 matrix with all elements equal to 1. We also find that it is an interesting fact that similar properties hold for their inverse matrices:

$$\mathbb{A}_1^{-1} + \mathbb{A}_2^{-1} = \mathbb{1} \tag{3.7.3}$$

Criteria	Reasons
1. L_1 and L_2 are orthogonal Latin squares.	8 w-bit variables belonging to $\mathbf{X}$ are to be mixed with 8 w-bit variables belonging to $\mathbf{Y}$ in such a way that all pairs are combined by some operation (addition, or XORing).
2. $\mathbf{Diff}_{\pi_2}$ and $\mathbf{Diff}_{\pi_3}$ do not have zeroes.	The situation where $\mathbf{X} *_q \mathbf{Y} = \mathbf{Z}$ and some difference either in $\mathbf{X}$ or in $\mathbf{Y}$ will not affect some of the eight words of $\mathbf{Z}$ are to be avoided.
3. Elements of the matrix $\mathbf{Diff}_{\pi_2}$ have the biggest possible variance.	This is an analogy to the "confusion" principle in cryptology. Choosing $\mathbf{Diff}_{\pi_2}$ with the biggest possible variance improves the resistance against cryptanalysis because there is no regular pattern how the computations are performed.
4. Elements of the matrix $\mathbf{Diff}_{\pi_3}$ have the smallest possible variance.	This is an analogy to the "diffusion" principle in cryptology. Choosing $\mathbf{Diff}_{\pi_3}$ with the smallest possible variance increases the diffusion of the bit differences in the greatest possible way, with the smallest possible variances in the pattern of the computations that are performed.

Table 3.9: Criteria for choosing the Latin squares

$$\underset{L}{\left[\begin{array}{cccccccc} 5 & 8 & 3 & 4 & 2 & 7 & 1 & 6 \\ 3 & 5 & 2 & 1 & 7 & 4 & 6 & 8 \\ 2 & 4 & 8 & 6 & 1 & 5 & 7 & 3 \\ 6 & 3 & 5 & 2 & 4 & 1 & 8 & 7 \\ 7 & 2 & 1 & 3 & 8 & 6 & 5 & 4 \\ \hline 4 & 1 & 7 & 8 & 6 & 3 & 2 & 5 \\ 1 & 7 & 6 & 5 & 3 & 8 & 4 & 2 \\ 8 & 6 & 4 & 7 & 5 & 2 & 3 & 1 \end{array}\right]} \underset{\mathbb{A}_1}{\begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}}$$

Figure 3.1: Example of how the map works. We see that the 3rd element of row 4 of $\mathbb{A}_1$ is 1 because the number 3 is above the line in column 4 of L. Note that we denote the elements from 1 to 8 instead from 0 to 7.

and

$$\mathbb{A}_3^{-1} + \mathbb{A}_4^{-1} = \mathbb{1}. \tag{3.7.4}$$

In what follows, we prove that this complementary property is true for every pair of Boolean matrices obtained from Latin squares if certain preconditions are fulfilled.

Let us define a map from the set of Latin squares to the set of boolean matrices by the following definition:

Definition 13. Let $\mathcal{L}^n$ be the set of Latin squares of size n, and let $M(\mathbb{Z}_2)$ be the set of Boolean matrices. We define the map

$$\begin{array}{rccc} F_k^n: & \mathcal{L}^n & \longrightarrow & M(\mathbb{Z}_2) \times M(\mathbb{Z}_2) \\ & L & \longmapsto & (\mathbb{A}_1, \mathbb{A}_2) \end{array}$$

where

$$(\mathbb{A}_1)_{ij} = \begin{cases} 1 & \text{if } j \in \{L_{1i}, \dots, L_{ki}\} \\ 0 & \text{else} \end{cases} \text{ and } (\mathbb{A}_2)_{ij} = \begin{cases} 1 & \text{if } j \in \{L_{ki}, \dots, L_{ni}\} \\ 0 & \text{else} \end{cases}$$

To easier see how the map works imagine drawing a line between row k and $k+1$ of the Latin square as shown in Figure 3.1. Then the j-th element of row i of $\mathbb{A}_1$ equals 1 if the number j is above the drawn line in the i-th column of the Latin square. From the definition of Latin squares we see that the weight of every row and column of $\mathbb{A}_1$ is k, and that $\mathbb{A}_2 = \mathbb{A}_1 + \mathbb{1}$, where $\mathbb{1}$ is the all 1 matrix of appropriate size. Notice that for each pair (L, k) such that $F_k^n(L) = (\mathbb{A}_1, \mathbb{A}_2)$, there exists a Latin square, L', such that $F^n_{n-k}(L') = (\mathbb{A}_2, \mathbb{A}_1)$[1]. This fact will be used to prove some of the results in the next section.

In this section we show that if both n and $n-k$ is odd, the map from the previous section defines an interesting class of boolean matrices where both $\mathbb{A}_2 = \mathbb{A}_1 + \mathbb{1}$ and, if the matrices are non-singular, $\mathbb{A}_2^{-1} = \mathbb{A}_1^{-1} + \mathbb{1}$ as well. To do this we first need to prove the following two Lemmas.

Lemma 4. Let L be a Latin square, let $n-k$ be odd, and let $F^n_k(L) = (\mathbb{A}_1, \mathbb{A}_2)$. Then $\det(\mathbb{A}_1) = 1$ mod 2 if and only if $\det(\mathbb{A}_2) = 1$ mod 2.

Proof. Let $\det(\mathbb{A}_1) = 1$ mod 2. This means that the vector columns $v_1, \ldots, v_n$ of $\mathbb{A}_1$ are linearly independent. Let $\tilde{v}_1, \ldots, \tilde{v}_2$ be the corresponding vector columns of $\mathbb{A}_2$, i.e. $\tilde{v}_i = v_i + \mathbb{1}$. Assume $\det(\mathbb{A}_2) = 0$ mod 2. This means that there exists some $\tilde{v}_l$ and $b_1, \ldots b_n \in \{0, 1\}$ such that

$$\mathbb{1} + v_l = \tilde{v}_l = \sum_{i \neq l} b_i \tilde{v}_i = \sum_{i \neq l} b_i(\mathbb{1} + v_i) = \begin{cases} \mathbb{1} + \sum_{i \neq l} b_i v_i & \text{if } \sum b_i = 1 \\ \sum_{i \neq l} b_i v_i & \text{if } \sum b_i = 0 \end{cases}$$

The first case implies that $\mathbb{1} + v_l = \mathbb{1} + \sum_{i \neq l} b_i v_i \Rightarrow v_l = \sum_{i \neq l} b_i v_i$, which is not possible since the vectors $v_1, \ldots, v_n$ of $\mathbb{A}_1$ are linearly independent. The second case, $\mathbb{1} + v_l = \sum_{i \neq l} b_i v_i$, is not possible either, since the vector on right side of the equation has even weight, and a sum of an even number of vectors with the same weight must have even weight. The left side of the equation must have odd weight since $n-k$ is odd. Since neither of the above cases is possible, the assumption that $\det(\mathbb{A}_2) = 0$ mod 2 is wrong.

This means $\det(\mathbb{A}_1) = 1$ mod 2 $\Rightarrow \det(\mathbb{A}_2) = 1$ mod 2. To prove $\det(\mathbb{A}_2) = 1$ mod 2 $\Rightarrow \det(\mathbb{A}_1) = 1$ mod 2 notice that for every pair (L, k) such that $F^n_k(L) = (\mathbb{A}_1, \mathbb{A}_2)$ there exists a pair (L', k') such that $F^n_{k'}(L') = (\mathbb{A}_2, \mathbb{A}_1)$. □

Lemma 5. Let L be a Latin square, let k and $n-k$ be odd and let $F^n_k(L) = (\mathbb{A}_1, \mathbb{A}_2)$. If $\mathbb{A}_1$ is non-singular then the weight of any row or column of $\mathbb{A}_1^{-1}$ or $\mathbb{A}_2^{-1}$ is odd.

Proof. Let the $l_{i,1}, \ldots l_{i,k}$ be the k indexes where row i of $\mathbb{A}_1$ is different from zero. Assume $\mathbb{A}_1$ (and therefore also $\mathbb{A}_2$ by Lemma 4) is non-singular and let b_{ij} be the elements of $\mathbb{A}_1^{-1}$. Since

[1] The Latin square L' is just the square L where the top k-rows and the bottom $n-k$-rows has exhanged place

$\mathbb{A}_1\mathbb{A}_1^{-1} = I$ we have the following equations for any column j of $\mathbb{A}_1^{-1}$

$$\begin{aligned} b_{l_{1,1}j} + \cdots + b_{l_{1k}j} &= 0 \\ &\vdots \\ b_{l_{j-1,1}j} + \cdots + b_{l_{j-1,k}j} &= 0 \\ b_{l_{j,1}j} + \cdots + b_{l_{j,k}j} &= 1 \\ b_{l_{j+1,1}j} + \cdots + b_{l_{j+1,k}j} &= 0 \\ &\vdots \\ b_{l_{n,1}j} + \cdots + b_{l_{n,k}j} &= 0 \end{aligned}$$

By the property of the Latin square, we know that for each i the index b_{ij} must appear in the above equations exactly k times. Using this fact and summing the above equation together we get $k(b_{1j} + \cdots + b_{nj}) = 1$. Since k is odd this means that the weight of every column of $\mathbb{A}_1^{-1}$ must be 1. The proof for the rows of $\mathbb{A}_1^{-1}$ is similar, starting with the equation $\mathbb{A}_1^{-1}\mathbb{A}_1 = I$.

The proof for any row or column of $\mathbb{A}_2$ follows by the fact that for every pair (L, k) such that $F_k^n(L) = (\mathbb{A}_1, \mathbb{A}_2)$ there exists a pair (L', k') such that $F_{k'}^n(L') = (\mathbb{A}_2, \mathbb{A}_1)$. □

We are now ready to prove the main result in this section.

Theorem 6. Let L be a Latin square, let $F_k^n = (\mathbb{A}_1, \mathbb{A}_2)$, where k and $n-k$ is odd. If $\det(\mathbb{A}_1) = 1 \mod 2$, then

$$\mathbb{A}_2^{-1} = \mathbb{A}_1^{-1} + \mathbb{1}$$

Proof. First we know that $\mathbb{A}_2$ is non-singular by Lemma 4. We then use this fact in the following equations.

$$\begin{aligned} \mathbb{A}_1 &= \mathbb{1} + \mathbb{A}_2 \qquad (3.7.5) \\ \mathbb{A}_1\mathbb{A}_1^{-1} &= \mathbb{1}\mathbb{A}_1^{-1} + \mathbb{A}_2\mathbb{A}_1^{-1} \qquad (3.7.6) \\ I &= \mathbb{1}\mathbb{A}_1^{-1} + \mathbb{A}_2\mathbb{A}_1^{-1} \qquad (3.7.7) \\ \mathbb{A}_2^{-1} &= \mathbb{A}_2^{-1}\mathbb{1}\mathbb{A}_1^{-1} + \mathbb{A}_1^{-1} \qquad (3.7.8) \\ \mathbb{A}_2^{-1} &= \mathbb{1} + \mathbb{A}_1^{-1} \qquad (3.7.9) \end{aligned}$$

Where equality 3.7.5 follows from the definition of Latin squares, and the step from equality 3.7.8 to equality 3.7.9 follows from Lemma 5. □

3.8 EDON-$\mathcal{R}$ is a double-pipe iterated compression function

The design of EDON-$\mathcal{R}$ is a double-pipe iterated compression function. Although it is similar to the classic Merkle-Damgård iterated design [21–23], in the light of the latest attacks with multi-collisions, it is also essentially different from it. In the design of EDON-$\mathcal{R}$ we have decided to incorporate the suggestions of Lucks [24, 25] and Coron et al. [26] by setting the size of the internal memory of the iterated compression function to be twice as large as the output length required. This design avoids the weaknesses against the generic attacks of Joux [27] and Kelsy and Schneier [28], thereby guaranteeing resistance against a generic multicollision attack and length extension attacks.

Doubling of the internal memory in our design is a result of the fact that in every iterative step of the compression function, the strings of length $4n$ bits ($2n$ bits from the double pipe and $2n$ bits from the message) are mapped to strings of length $2n$ bits which are becoming the actual value of the double pipe for the next iterative step.

3.9 Natural resistance of EDON-$\mathcal{R}$ against generic length extension attacks

Generic length extension attacks on iterated hash function based upon Merkle-Damgård iterative design principles [22, 23] works as follows:

Let $M = M_1||M_2|| \dots ||M_N$ be a message consisting of exactly N blocks that will be iteratively digested by some compression function $C(A, B)$ according to the Merkle-Damgård iterative design principles, and where A and B are messages (input parameters for the compression function) that have the same length as the final message digest. Let P_M be the padding block of M obtained according to the Merkle-Damgård strengthening. Then, the digest H of the message M, is computed as

$$H(M) = C(\dots C(C(IV, M_1), M_2) \dots, P_M),$$

where IV is the initial fixed value for the hash function.

Now suppose that the attacker does not know the message M, but knows (or can easily guess) the length of the message M. The attacker knows the padding block P_M. Now, the attacker can construct a new message $M' = P_M||M'_1$ such that he knows the hash digest of the message $M||M'$, i.e.,

$$H(M||M') = C(C(H(M), M'_1), P_{M'}),$$

where $P_{M'}$ is the padding (Merkle-Damgård strengthening) of the message $M||M'$.

Edon-$\mathcal{R}$ has a natural resistance against this generic attack due to the fact that it is iterated with the chaining variables that has a length that is two times greater than the final digest value (see also the work of Lucks [29]).

3.10 Testing avalanche properties of Edon-$\mathcal{R}$

We show the avalanche propagation of the initial one bit differences of the compression function of $\mathcal{R}$ during their evolution in all 8 quasigroup operations $*_q, (q = 256, 512)$.

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 100 randomly generated messages of n–bits.

The results for $n = 256$ are shown in Table 3.10. Notice that a 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 in the third row (in bold). Similar results are obtained for $n = 512$ and are shown in Table 3.11. There also a 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 from the third row (in bold).

3.11 All collision paths of $\mathcal{R}$ and local collisions

The design of the compression function $\mathcal{R}$ in Edon-$\mathcal{R}$ is pretty different from the design of compression functions of known hash functions that are designed from scratch. While other compression functions have 64, 80 or even more iterating steps, $\mathcal{R}$ has 8 steps. So far, all successful attacks against the MDx and SHA families of hash functions exploited local collisions in the processing of the data block. Local collisions are collisions that can be found within a few steps of the compression function.

In what follows we find local collisions for Edon-$\mathcal{R}$ and discuss difficulties how these local collisions can lead to collisions of the whole function.

$Min = 15$ $Avr = 28.86$ $Max = 50$	$Min = 92$ $Avr = 117.66$ $Max = 139$
$Min = 96$ $Avr = 120.06$ $Max = 143$	$Min = 99$ **Avr = 127.72** $Max = 152$
$Min = 106$ **Avr=128.69** $Max = 150$	$Min = 105$ **Avr=127.91** $Max = 152$
$Min = 108$ **Avr=127.79** $Max = 150$	$Min = 98$ **Avr=128.05** $Max = 150$

a.

$Min = 16$ $Avr = 34.71$ $Max = 72$	$Min = 66$ $Avr = 119.98$ $Max = 157$
$Min = 66$ $Avr = 118.63$ $Max = 156$	$Min = 91$ **Avr=128.00** $Max = 160$
$Min = 95$ **Avr=128.00** $Max = 160$	$Min = 95$ **Avr=128.04** $Max = 162$
$Min = 96$ **Avr=128.01** $Max = 158$	$Min = 94$ **Avr=127.94** $Max = 161$

b.

Table 3.10: 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)

The small number of steps in the compression function $\mathcal{R}$ as well as the algebraic properties of quasigroup operations allow us to describe all possible collision paths within the compression function which, we emphasize again, is a unique property among all known hash functions that are designed from scratch.

In order to track the collision paths for the compression function $\mathcal{R}$ we introduce a definition for quasigroup operation between sets of cardinality one and two.

Definition 14. Let $A_1 = \{\mathbf{A_1}\}, A_2 = \{\mathbf{A_1}, \mathbf{A_2}\}, B_1 = \{\mathbf{B_1}\}, B_2 = \{\mathbf{B_1}, \mathbf{B_2}\}, C_1 = \{\mathbf{C_1}\}, C_2 = \{\mathbf{C_1}, \mathbf{C_2}\}$ be sets of cardinality one or two and where $\mathbf{A}_i, \mathbf{B}_i$ and $\mathbf{C}_i \in Q_q (q = 256, 512)$. The operation of quasigroup multiplication $*_q$ between these sets is defined by Table 3.12.

Following directly from the properties of the unique solutions of equations of type (2.1.1) it is easy to prove the following two propositions:

Proposition 5. If $\mathbf{B_1} \neq \mathbf{B_2}$ then $\{\mathbf{A_1}\} *_q \{\mathbf{B_1}, \mathbf{B_2}\} = \{\mathbf{C_1}, \mathbf{C_2}\}$ such that $\mathbf{C_1} \neq \mathbf{C_2}$. □

Proposition 6. If $\mathbf{A_1} \neq \mathbf{A_2}$ then $\{\mathbf{A_1}, \mathbf{A_2}\} *_q \{\mathbf{B_1}\} = \{\mathbf{C_1}, \mathbf{C_2}\}$ such that $\mathbf{C_1} \neq \mathbf{C_2}$. □

$Min = 15$ $Avr = 24.10$ $Max = 51$	$Min = 91$ $Avr = 140.31$ $Max = 181$
$Min = 125$ $Avr = 85.39$ $Max = 231$	$Min = 220$ $\mathbf{Avr = 255.51}$ $Max = 295$
$Min = 213$ $\mathbf{Avr = 255.69}$ $Max = 295$	$Min = 218$ $\mathbf{Avr = 256.03}$ $Max = 292$
$Min = 216$ $\mathbf{Avr = 255.31}$ $Max = 294$	$Min = 221$ $\mathbf{Avr = 255.83}$ $Max = 288$

a.

$Min = 16$ $Avr = 35.62$ $Max = 77$	$Min = 88$ $Avr = 167.20$ $Max = 254$
$Min = 70$ $Avr = 156.55$ $Max = 260$	$Min = 205$ $\mathbf{Avr = 255.94}$ $Max = 303$
$Min = 192$ $Avr = 252.41$ $Max = 304$	$Min = 207$ $\mathbf{Avr = 256.01}$ $Max = 302$
$Min = 205$ $\mathbf{Avr = 256.02}$ $Max = 310$	$Min = 206$ $\mathbf{Avr = 256.02}$ $Max = 305$

b.

Table 3.11: 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)

$*_q$	$B_1 = \{\mathbf{B_1}\}$	$B_2 = \{\mathbf{B_1}, \mathbf{B_2}\}$		
$A_1 = \{\mathbf{A_1}\}$	$C_1 = \{\mathbf{C_1}\}$ where $\mathbf{A_1} *_q \mathbf{B_1} = \mathbf{C_1}$	$C_2 = \{\mathbf{C_1}, \mathbf{C_2}\}$ where $\mathbf{A_1} *_q \mathbf{B_1} = \mathbf{C_1}$ and $\mathbf{A_1} *_q \mathbf{B_2} = \mathbf{C_2}$		
$A_2 = \{\mathbf{A_1}, \mathbf{A_2}\}$	$C_2 = \{\mathbf{C_1}, \mathbf{C_2}\}$ where $\mathbf{A_1} *_q \mathbf{B_1} = \mathbf{C_1}$ and $\mathbf{A_2} *_q \mathbf{B_1} = \mathbf{C_2}$	$C_2 = \{\mathbf{C_1}, \mathbf{C_2}\}$ where $\mathbf{A_1} *_q \mathbf{B_1} = \mathbf{C_1}$ and $\mathbf{A_2} *_q \mathbf{B_2} = \mathbf{C_2}$	or	$C_1 = \{\mathbf{C_1}\}$ where $\mathbf{A_1} *_q \mathbf{B_1} = \mathbf{C_1}$ and $\mathbf{A_2} *_q \mathbf{B_2} = \mathbf{C_1}$

Table 3.12: Definition of quasigroup operation between one or two-element sets.

However if both $\mathbf{A_1} \neq \mathbf{A_2}$ and $\mathbf{B_1} \neq \mathbf{B_2}$ then $\{\mathbf{A_1}, \mathbf{A_2}\} *_q \{\mathbf{B_1}, \mathbf{B_2}\}$ can be either $\{\mathbf{C_1}, \mathbf{C_2}\}$ or $\{\mathbf{C_1}\}$ and this is formulated in the following proposition:

Proposition 7. If $\mathbf{A_1} \neq \mathbf{A_2}$ and $\mathbf{B_1} \neq \mathbf{B_2}$ then $\{\mathbf{A_1}, \mathbf{A_2}\} *_q \{\mathbf{B_1}, \mathbf{B_2}\}$ can be either $\{\mathbf{C_1}, \mathbf{C_2}\}$ (where $\mathbf{C_1} \neq \mathbf{C_2}$) or $\{\mathbf{C_1}\}$. □

We formalize the notion of collisions for the compression function $\mathcal{R}$ by the following definition:

Definition 15. Let $(\mathbf{C_0},\mathbf{C_1},\mathbf{X_1},\mathbf{X_2}),(\mathbf{C_0},\mathbf{C_1},\mathbf{X_3},\mathbf{X_4}) \in Q^4$ where $\mathbf{C_0}$ and $\mathbf{C_1}$ are initial constants defined in Subsection 1.4.3, and $(\mathbf{X_1},\mathbf{X_2}) \neq (\mathbf{X_3},\mathbf{X_4})$. If $\mathcal{R}(\mathbf{C_0},\mathbf{C_1},\mathbf{X_1},\mathbf{X_2}) = (\mathbf{D},\mathbf{Y})$ and $\mathcal{R}(\mathbf{C_0},\mathbf{C_1},\mathbf{X_3},\mathbf{X_4}) = (\mathbf{E},\mathbf{Y})$ then the quintette $(\mathbf{X_1},\mathbf{X_2},\mathbf{X_3},\mathbf{X_4})$ is a collision for $\mathcal{R}$.

Using the Definition 14 and Definition 15 we can trace all possible paths that can produce collisions in the compression function $\mathcal{R}$. That is formulated in the following theorem:

Theorem 7. If $(\mathbf{X_1},\mathbf{X_2}) \neq (\mathbf{X_3},\mathbf{X_4})$ are two pairs of values in Q_q. Then all possible differential paths starting with the set $\{\mathbf{X_1},\mathbf{X_2},\mathbf{X_3},\mathbf{X_4}\}$ that can produce collisions in the compression function $\mathcal{R}$ are described in Table 3.13 and Table 3.14. □

	$\{\mathbf{X}_1\}$	$\{\mathbf{X}_2,\mathbf{X}_3\}$
$\{\overline{\mathbf{X}}_2,\overline{\mathbf{X}}_3\}$	$\{\mathbf{D}_1,\mathbf{D}_2\}$	$\{\mathbf{D}_3\}$
$\{\mathbf{C}_0\}$	$\{\mathbf{D}_4,\mathbf{D}_5\}$	$\{\mathbf{D}_6,\mathbf{D}_7\}$
$\{\mathbf{C}_1\}$	$\{\mathbf{D}_8,\mathbf{D}_9\}$	$\{\mathbf{D}_{10},\mathbf{D}_{11}\}$
$\{\overline{\mathbf{X}}_1\}$	$\{\mathbf{D}_{12},\mathbf{D}_{13}\}$	$\{\mathbf{D}_{14}\}$

a.

	$\{\mathbf{X}_1\}$	$\{\mathbf{X}_2,\mathbf{X}_3\}$
$\{\overline{\mathbf{X}}_2,\overline{\mathbf{X}}_3\}$	$\{\mathbf{D}_1,\mathbf{D}_2\}$	$\{\mathbf{D}_3,\mathbf{D}_4\}$
$\{\mathbf{C}_0\}$	$\{\mathbf{D}_5,\mathbf{D}_6\}$	$\{\mathbf{D}_7\}$
$\{\mathbf{C}_1\}$	$\{\mathbf{D}_8,\mathbf{D}_9\}$	$\{\mathbf{D}_{10},\mathbf{D}_{11}\}$
$\{\overline{\mathbf{X}}_1\}$	$\{\mathbf{D}_{12},\mathbf{D}_{13}\}$	$\{\mathbf{D}_{14}\}$

b.

	$\{\mathbf{X}_1\}$	$\{\mathbf{X}_2,\mathbf{X}_3\}$
$\{\overline{\mathbf{X}}_2,\overline{\mathbf{X}}_3\}$	$\{\mathbf{D}_1,\mathbf{D}_2\}$	$\{\mathbf{D}_3,\mathbf{D}_4\}$
$\{\mathbf{C}_0\}$	$\{\mathbf{D}_5,\mathbf{D}_6\}$	$\{\mathbf{D}_7,\mathbf{D}_8\}$
$\{\mathbf{C}_1\}$	$\{\mathbf{D}_9,\mathbf{D}_{10}\}$	$\{\mathbf{D}_{11},\mathbf{D}_{12}\}$
$\{\overline{\mathbf{X}}_1\}$	$\{\mathbf{D}_{13},\mathbf{D}_{14}\}$	$\{\mathbf{D}_{15}\}$

c.

	$\{\mathbf{X}_1,\mathbf{X}_2\}$	$\{\mathbf{X}_3\}$
$\{\overline{\mathbf{X}}_3\}$	$\{\mathbf{D}_1,\mathbf{D}_2\}$	$\{\mathbf{D}_3,\mathbf{D}_4\}$
$\{\mathbf{C}_0\}$	$\{\mathbf{D}_5,\mathbf{D}_6\}$	$\{\mathbf{D}_7\}$
$\{\mathbf{C}_1\}$	$\{\mathbf{D}_8,\mathbf{D}_9\}$	$\{\mathbf{D}_{10},\mathbf{D}_{11}\}$
$\{\overline{\mathbf{X}}_1,\overline{\mathbf{X}}_2\}$	$\{\mathbf{D}_{12},\mathbf{D}_{13}\}$	$\{\mathbf{D}_{14}\}$

d.

	$\{\mathbf{X}_1,\mathbf{X}_2\}$	$\{\mathbf{X}_3\}$
$\{\overline{\mathbf{X}}_3\}$	$\{\mathbf{D}_1,\mathbf{D}_2\}$	$\{\mathbf{D}_3,\mathbf{D}_4\}$
$\{\mathbf{C}_0\}$	$\{\mathbf{D}_5,\mathbf{D}_6\}$	$\{\mathbf{D}_7,\mathbf{D}_8\}$
$\{\mathbf{C}_1\}$	$\{\mathbf{D}_9,\mathbf{D}_{10}\}$	$\{\mathbf{D}_{11}\}$
$\{\overline{\mathbf{X}}_1,\overline{\mathbf{X}}_2\}$	$\{\mathbf{D}_{12}\}$	$\{\mathbf{D}_{13}\}$

e.

	$\{\mathbf{X}_1,\mathbf{X}_2\}$	$\{\mathbf{X}_3\}$
$\{\overline{\mathbf{X}}_3\}$	$\{\mathbf{D}_1,\mathbf{D}_2\}$	$\{\mathbf{D}_3,\mathbf{D}_4\}$
$\{\mathbf{C}_0\}$	$\{\mathbf{D}_5,\mathbf{D}_6\}$	$\{\mathbf{D}_7,\mathbf{D}_8\}$
$\{\mathbf{C}_1\}$	$\{\mathbf{D}_9,\mathbf{D}_{10}\}$	$\{\mathbf{D}_{11},\mathbf{D}_{12}\}$
$\{\overline{\mathbf{X}}_1,\overline{\mathbf{X}}_2\}$	$\{\mathbf{D}_{13},\mathbf{D}_{14}\}$	$\{\mathbf{D}_{15}\}$

f.

Table 3.13: First part of the description of all possible differential paths in the compression function $\mathcal{R}$ that can give collisions. Cases a. – f.

The corresponding quasigroup equations for all these cases are given in Table 3.15, Table 3.16 and Table 3.17.

	$\{\mathbf{X}_1,\mathbf{X}_2\}$	$\{\mathbf{X}_3,\mathbf{X}_4\}$
$\{\overline{\mathbf{X}}_3,\overline{\mathbf{X}}_4\}$	$\{\mathbf{D}_1\}$	$\{\mathbf{D}_2,\mathbf{D}_3\}$
$\{\mathbf{C}_0\}$	$\{\mathbf{D}_4\}$	$\{\mathbf{D}_5,\mathbf{D}_6\}$
$\{\mathbf{C}_1\}$	$\{\mathbf{D}_7\}$	$\{\mathbf{D}_8,\mathbf{D}_9\}$
$\{\overline{\mathbf{X}}_1,\overline{\mathbf{X}}_2\}$	$\{\mathbf{D}_{10},\mathbf{D}_{11}\}$	$\{\mathbf{D}_{12}\}$

g.

	$\{\mathbf{X}_1,\mathbf{X}_2\}$	$\{\mathbf{X}_3,\mathbf{X}_4\}$
$\{\overline{\mathbf{X}}_3,\overline{\mathbf{X}}_4\}$	$\{\mathbf{D}_1,\mathbf{D}_2\}$	$\{\mathbf{D}_3\}$
$\{\mathbf{C}_0\}$	$\{\mathbf{D}_4,\mathbf{D}_5\}$	$\{\mathbf{D}_6,\mathbf{D}_7\}$
$\{\mathbf{C}_1\}$	$\{\mathbf{D}_8,\mathbf{D}_9\}$	$\{\mathbf{D}_{10}\}$
$\{\overline{\mathbf{X}}_1,\overline{\mathbf{X}}_2\}$	$\{\mathbf{D}_{11}\}$	$\{\mathbf{D}_{12}\}$

h.

	$\{\mathbf{X}_1,\mathbf{X}_2\}$	$\{\mathbf{X}_3,\mathbf{X}_4\}$
$\{\overline{\mathbf{X}}_3,\overline{\mathbf{X}}_4\}$	$\{\mathbf{D}_1,\mathbf{D}_2\}$	$\{\mathbf{D}_3\}$
$\{\mathbf{C}_0\}$	$\{\mathbf{D}_4,\mathbf{D}_5\}$	$\{\mathbf{D}_6,\mathbf{D}_7\}$
$\{\mathbf{C}_1\}$	$\{\mathbf{D}_8,\mathbf{D}_9\}$	$\{\mathbf{D}_{10},\mathbf{D}_{11}\}$
$\{\overline{\mathbf{X}}_1,\overline{\mathbf{X}}_2\}$	$\{\mathbf{D}_{12},\mathbf{D}_{13}\}$	$\{\mathbf{D}_{14}\}$

i.

	$\{\mathbf{X}_1,\mathbf{X}_2\}$	$\{\mathbf{X}_3,\mathbf{X}_4\}$
$\{\overline{\mathbf{X}}_3,\overline{\mathbf{X}}_4\}$	$\{\mathbf{D}_1,\mathbf{D}_2\}$	$\{\mathbf{D}_3,\mathbf{D}_4\}$
$\{\mathbf{C}_0\}$	$\{\mathbf{D}_5,\mathbf{D}_6\}$	$\{\mathbf{D}_7\}$
$\{\mathbf{C}_1\}$	$\{\mathbf{D}_8,\mathbf{D}_9\}$	$\{\mathbf{D}_{10},\mathbf{D}_{11}\}$
$\{\overline{\mathbf{X}}_1,\overline{\mathbf{X}}_2\}$	$\{\mathbf{D}_{12},\mathbf{D}_{13}\}$	$\{\mathbf{D}_{14}\}$

j.

	$\{\mathbf{X}_1,\mathbf{X}_2\}$	$\{\mathbf{X}_3,\mathbf{X}_4\}$
$\{\overline{\mathbf{X}}_3,\overline{\mathbf{X}}_4\}$	$\{\mathbf{D}_1,\mathbf{D}_2\}$	$\{\mathbf{D}_3,\mathbf{D}_4\}$
$\{\mathbf{C}_0\}$	$\{\mathbf{D}_5,\mathbf{D}_6\}$	$\{\mathbf{D}_7,\mathbf{D}_8\}$
$\{\mathbf{C}_1\}$	$\{\mathbf{D}_9,\mathbf{D}_{10}\}$	$\{\mathbf{D}_{11}\}$
$\{\overline{\mathbf{X}}_1,\overline{\mathbf{X}}_2\}$	$\{\mathbf{D}_{12}\}$	$\{\mathbf{D}_{13}\}$

k.

	$\{\mathbf{X}_1,\mathbf{X}_2\}$	$\{\mathbf{X}_3,\mathbf{X}_4\}$
$\{\overline{\mathbf{X}}_3,\overline{\mathbf{X}}_4\}$	$\{\mathbf{D}_1,\mathbf{D}_2\}$	$\{\mathbf{D}_3,\mathbf{D}_4\}$
$\{\mathbf{C}_0\}$	$\{\mathbf{D}_5,\mathbf{D}_6\}$	$\{\mathbf{D}_7,\mathbf{D}_8\}$
$\{\mathbf{C}_1\}$	$\{\mathbf{D}_9,\mathbf{D}_{10}\}$	$\{\mathbf{D}_{11},\mathbf{D}_{12}\}$
$\{\overline{\mathbf{X}}_1,\overline{\mathbf{X}}_2\}$	$\{\mathbf{D}_{13},\mathbf{D}_{14}\}$	$\{\mathbf{D}_{15}\}$

l.

Table 3.14: Second part of the description of all possible differential paths in the compression function $\mathcal{R}$ that can give collisions. Cases g. – l.

In what follows we need the notation of left conjugates (left parastrophes) of a given quasigroup operation $*$ i.e.

$$\mathbf{X} * \mathbf{Y} = \mathbf{Z} \quad \Leftrightarrow \quad \mathbf{X} \setminus \mathbf{Z} = \mathbf{Y}.$$

Generally, we can divide the problem of finding local collisions in two cases. The first case is the local collisions described in Table 3.13. For those collisions, we find that solving corresponding quasigroup equations is hard. For example, let us discuss the case described in Table 3.13a., with the corresponding equations given in Table 3.15a. We apply the following attack:

1. Choose some arbitrary value for $\mathbf{D}_3$.

2. Choose two distinct values $\mathbf{D}_1$ and $\mathbf{D}_2$.

3. Compute $\mathbf{X}_2 = \mathbf{D}_1 \setminus \mathbf{D}_3$.
4. Compute $\mathbf{X}_3 = \mathbf{D}_2 \setminus \mathbf{D}_3$.
5. If $(\overline{\mathbf{X}}_2 \setminus \mathbf{D}_1) = (\overline{\mathbf{X}}_3 \setminus \mathbf{D}_2)$, then set $\mathbf{X}_1 = (\overline{\mathbf{X}}_2 \setminus \mathbf{D}_1)$ else Go to Step 1.

The difficulty for solving local collision cases described in Table 3.13 lies in the fact that we are faced with a feedback information (Step 5. in the previous attack) that is coming from the reversed strings according to the definition of the compression function $\mathcal{R}$.

On the other hand, for some of the local collisions described in the cases g., h., i., j., and k., (Table 3.14) there is no need to use the feedback information in the computations. We use that fact in order to find local collisions with complexity $O(1)$. However, we want to stress the fact that in the complete computation of the compression function $\mathcal{R}$ the feedback information of the processed message bits is the essential part of its definition. In such a way the usefulness of these local collisions in attacks for finding collisions or free start collisions for the compression function $\mathcal{R}$ is diminished.

We can elaborate further the non-applicability of the attacks that use local collisions on EDON-$\mathcal{R}$ by the following discussion. The attacks that use local collisions are applied on hash functions that have many steps in the phase of their initial message expansion (see for example [30]) and have generally the following two phases:

First: Some perturbation is introduced and it is corrected (i.e. the local collision is found).

Second: Perturbation and correction vectors are found, such that the overall difference mask satisfies the message expansion.

EDON-$\mathcal{R}$ hash function does not have a message expansion part, and does not have many steps where attacker can find perturbation and correction vectors.

In what follows we are describing the algorithms with complexity $O(1)$ for finding local collisions for the cases g., h., i., j., and k.

Case g. Finding local collisions for $\mathbf{D}_1$:

1. Choose some arbitrary value for $\mathbf{D}_1$.
2. Choose two distinct values $\overline{\mathbf{X}}_3$ and $\overline{\mathbf{X}}_4$.
3. Compute $\mathbf{X}_1 = \overline{\mathbf{X}}_3 \setminus \mathbf{D}_1$.
4. Compute $\mathbf{X}_2 = \overline{\mathbf{X}}_4 \setminus \mathbf{D}_1$.

Case h. Finding local collisions for $\mathbf{D}_3$:

1. Choose some arbitrary value for $\mathbf{D}_3$.
2. Choose two distinct values $\mathbf{D}_1$ and $\mathbf{D}_2$.
3. Compute $\mathbf{X}_3 = \mathbf{D}_1 \setminus \mathbf{D}_3$.
4. Compute $\mathbf{X}_4 = \mathbf{D}_2 \setminus \mathbf{D}_3$.
5. Compute $\mathbf{X}_1 = \overline{\mathbf{X}}_3 \setminus \mathbf{D}_1$.
6. Compute $\mathbf{X}_2 = \overline{\mathbf{X}}_4 \setminus \mathbf{D}_2$.

Case i. Finding local collisions for $\mathbf{D}_3$:

1. Choose some arbitrary value for $\mathbf{D}_3$.
2. Choose two distinct values $\mathbf{D}_1$ and $\mathbf{D}_2$.
3. Compute $\mathbf{X}_3 = \mathbf{D}_1 \setminus \mathbf{D}_3$.
4. Compute $\mathbf{X}_4 = \mathbf{D}_2 \setminus \mathbf{D}_3$.
5. Compute $\mathbf{X}_1 = \overline{\mathbf{X}}_3 \setminus \mathbf{D}_1$.
6. Compute $\mathbf{X}_2 = \overline{\mathbf{X}}_4 \setminus \mathbf{D}_2$.

Case j. Finding local collisions for $\mathbf{D}_7$:

1. Choose some arbitrary value for $\mathbf{D}_7$.
2. Choose two distinct values $\mathbf{D}_5$ and $\mathbf{D}_6$.
3. Compute $\mathbf{D}_3 = \mathbf{D}_5 \setminus \mathbf{D}_7$.
4. Compute $\mathbf{D}_4 = \mathbf{D}_6 \setminus \mathbf{D}_7$.
5. Compute $\mathbf{D}_1 = \mathbf{C}_0 \setminus \mathbf{D}_5$.
6. Compute $\mathbf{D}_2 = \mathbf{C}_0 \setminus \mathbf{D}_6$.
7. Compute $\mathbf{X}_3 = \mathbf{D}_1 \setminus \mathbf{D}_3$.
8. Compute $\mathbf{X}_4 = \mathbf{D}_2 \setminus \mathbf{D}_4$.
9. Compute $\mathbf{X}_1 = \overline{\mathbf{X}}_3 \setminus \mathbf{D}_1$.
10. Compute $\mathbf{X}_2 = \overline{\mathbf{D}}_4 \setminus \mathbf{D}_2$.

Case k. Finding local collisions for $\mathbf{D}_{11}$:

1. Choose some arbitrary value for $\mathbf{D}_{11}$.
2. Choose two distinct values $\mathbf{D}_9$ and $\mathbf{D}_{10}$.
3. Compute $\mathbf{D}_8 = \mathbf{D}_{10} \setminus \mathbf{D}_{11}$.
4. Compute $\mathbf{D}_7 = \mathbf{D}_9 \setminus \mathbf{D}_{11}$.
5. Compute $\mathbf{D}_5 = \mathbf{C}_1 \setminus \mathbf{D}_9$.
6. Compute $\mathbf{D}_6 = \mathbf{C}_1 \setminus \mathbf{D}_{10}$.
7. Compute $\mathbf{D}_4 = \mathbf{D}_6 \setminus \mathbf{D}_8$.
8. Compute $\mathbf{D}_3 = \mathbf{D}_5 \setminus \mathbf{D}_7$.
9. Compute $\mathbf{D}_2 = \mathbf{C}_0 \setminus \mathbf{D}_6$.
10. Compute $\mathbf{D}_1 = \mathbf{C}_0 \setminus \mathbf{D}_5$.
11. Compute $\mathbf{X}_4 = \mathbf{D}_2 \setminus \mathbf{D}_4$.
12. Compute $\mathbf{X}_3 = \mathbf{D}_1 \setminus \mathbf{D}_3$.
13. Compute $\mathbf{X}_1 = \overline{\mathbf{X}}_3 \setminus \mathbf{D}_1$.
14. Compute $\mathbf{X}_2 = \overline{\mathbf{D}}_4 \setminus \mathbf{D}_2$.

$$\left\{\begin{array}{rl}
\mathbf{D}_{14} = & \mathbf{D}_{13} * \mathbf{D}_{11} \\
\mathbf{D}_{14} = & \mathbf{D}_{12} * \mathbf{D}_{10} \\
\mathbf{D}_{13} = & \overline{\mathbf{X}}_1 * \mathbf{D}_9 \\
\mathbf{D}_{12} = & \overline{\mathbf{X}}_1 * \mathbf{D}_8 \\
\mathbf{D}_{11} = & \mathbf{D}_7 * \mathbf{D}_9 \\
\mathbf{D}_{10} = & \mathbf{D}_6 * \mathbf{D}_8 \\
\mathbf{D}_9 = & \mathbf{D}_5 * \mathbf{C}_1 \\
\mathbf{D}_8 = & \mathbf{D}_4 * \mathbf{C}_1 \\
\mathbf{D}_7 = & \mathbf{D}_5 * \mathbf{D}_3 \\
\mathbf{D}_6 = & \mathbf{D}_4 * \mathbf{D}_3 \\
\mathbf{D}_5 = & \mathbf{C}_0 * \mathbf{D}_2 \\
\mathbf{D}_4 = & \mathbf{C}_0 * \mathbf{D}_1 \\
\mathbf{D}_3 = & \mathbf{D}_2 * \mathbf{X}_3 \\
\mathbf{D}_3 = & \mathbf{D}_1 * \mathbf{X}_2 \\
\mathbf{D}_2 = & \overline{\mathbf{X}}_3 * \mathbf{X}_1 \\
\mathbf{D}_1 = & \overline{\mathbf{X}}_2 * \mathbf{X}_1
\end{array}\right.$$

a.

$$\left\{\begin{array}{rl}
\mathbf{D}_{14} = & \mathbf{D}_{13} * \mathbf{D}_{11} \\
\mathbf{D}_{14} = & \mathbf{D}_{12} * \mathbf{D}_{10} \\
\mathbf{D}_{13} = & \overline{\mathbf{X}}_1 * \mathbf{D}_9 \\
\mathbf{D}_{12} = & \overline{\mathbf{X}}_1 * \mathbf{D}_8 \\
\mathbf{D}_{11} = & \mathbf{D}_7 * \mathbf{D}_9 \\
\mathbf{D}_{10} = & \mathbf{D}_7 * \mathbf{D}_8 \\
\mathbf{D}_9 = & \mathbf{D}_6 * \mathbf{C}_1 \\
\mathbf{D}_8 = & \mathbf{D}_5 * \mathbf{C}_1 \\
\mathbf{D}_7 = & \mathbf{D}_6 * \mathbf{D}_4 \\
\mathbf{D}_7 = & \mathbf{D}_5 * \mathbf{D}_3 \\
\mathbf{D}_6 = & \mathbf{C}_0 * \mathbf{D}_2 \\
\mathbf{D}_5 = & \mathbf{C}_0 * \mathbf{D}_1 \\
\mathbf{D}_4 = & \mathbf{D}_2 * \mathbf{X}_3 \\
\mathbf{D}_3 = & \mathbf{D}_1 * \mathbf{X}_2 \\
\mathbf{D}_2 = & \overline{\mathbf{X}}_3 * \mathbf{X}_1 \\
\mathbf{D}_1 = & \overline{\mathbf{X}}_2 * \mathbf{X}_1
\end{array}\right.$$

b.

$$\left\{\begin{array}{rl}
\mathbf{D}_{15} = & \mathbf{D}_{14} * \mathbf{D}_{12} \\
\mathbf{D}_{15} = & \mathbf{D}_{13} * \mathbf{D}_{11} \\
\mathbf{D}_{14} = & \overline{\mathbf{X}}_1 * \mathbf{D}_{10} \\
\mathbf{D}_{13} = & \overline{\mathbf{X}}_1 * \mathbf{D}_9 \\
\mathbf{D}_{12} = & \mathbf{D}_8 * \mathbf{D}_{10} \\
\mathbf{D}_{11} = & \mathbf{D}_7 * \mathbf{D}_9 \\
\mathbf{D}_{10} = & \mathbf{D}_6 * \mathbf{C}_1 \\
\mathbf{D}_9 = & \mathbf{D}_5 * \mathbf{C}_1 \\
\mathbf{D}_8 = & \mathbf{D}_6 * \mathbf{D}_4 \\
\mathbf{D}_7 = & \mathbf{D}_5 * \mathbf{D}_3 \\
\mathbf{D}_6 = & \mathbf{C}_0 * \mathbf{D}_2 \\
\mathbf{D}_5 = & \mathbf{C}_0 * \mathbf{D}_1 \\
\mathbf{D}_4 = & \mathbf{D}_2 * \mathbf{X}_3 \\
\mathbf{D}_3 = & \mathbf{D}_1 * \mathbf{X}_2 \\
\mathbf{D}_2 = & \overline{\mathbf{X}}_3 * \mathbf{X}_1 \\
\mathbf{D}_1 = & \overline{\mathbf{X}}_2 * \mathbf{X}_1
\end{array}\right.$$

c.

$$\left\{\begin{array}{rl}
\mathbf{D}_{14} = & \mathbf{D}_{13} * \mathbf{D}_{11} \\
\mathbf{D}_{14} = & \mathbf{D}_{12} * \mathbf{D}_{10} \\
\mathbf{D}_{13} = & \overline{\mathbf{X}}_2 * \mathbf{D}_9 \\
\mathbf{D}_{12} = & \overline{\mathbf{X}}_1 * \mathbf{D}_8 \\
\mathbf{D}_{11} = & \mathbf{D}_7 * \mathbf{D}_9 \\
\mathbf{D}_{10} = & \mathbf{D}_7 * \mathbf{D}_8 \\
\mathbf{D}_9 = & \mathbf{D}_6 * \mathbf{C}_1 \\
\mathbf{D}_8 = & \mathbf{D}_5 * \mathbf{C}_1 \\
\mathbf{D}_7 = & \mathbf{D}_6 * \mathbf{D}_4 \\
\mathbf{D}_7 = & \mathbf{D}_5 * \mathbf{D}_3 \\
\mathbf{D}_6 = & \mathbf{C}_0 * \mathbf{D}_2 \\
\mathbf{D}_5 = & \mathbf{C}_0 * \mathbf{D}_1 \\
\mathbf{D}_4 = & \mathbf{D}_2 * \mathbf{X}_3 \\
\mathbf{D}_3 = & \mathbf{D}_1 * \mathbf{X}_3 \\
\mathbf{D}_2 = & \overline{\mathbf{X}}_3 * \mathbf{X}_2 \\
\mathbf{D}_1 = & \overline{\mathbf{X}}_3 * \mathbf{X}_1
\end{array}\right.$$

d.

Table 3.15: Concrete systems (a. – d.) of quasigroup equations that can give collisions in the compression function $\mathcal{R}$

$$\left\{\begin{array}{ll}
\mathbf{D}_{13} = & \mathbf{D}_{12} * \mathbf{D}_{11} \\
\mathbf{D}_{12} = & \overline{\mathbf{X}}_2 * \mathbf{D}_{10} \\
\mathbf{D}_{12} = & \overline{\mathbf{X}}_1 * \mathbf{D}_9 \\
\mathbf{D}_{11} = & \mathbf{D}_8 * \mathbf{D}_{10} \\
\mathbf{D}_{11} = & \mathbf{D}_7 * \mathbf{D}_9 \\
\mathbf{D}_{10} = & \mathbf{D}_6 * \mathbf{C}_1 \\
\mathbf{D}_9 = & \mathbf{D}_5 * \mathbf{C}_1 \\
\mathbf{D}_8 = & \mathbf{D}_6 * \mathbf{D}_4 \\
\mathbf{D}_7 = & \mathbf{D}_5 * \mathbf{D}_3 \\
\mathbf{D}_6 = & \mathbf{C}_0 * \mathbf{D}_2 \\
\mathbf{D}_5 = & \mathbf{C}_0 * \mathbf{D}_1 \\
\mathbf{D}_4 = & \mathbf{D}_2 * \mathbf{X}_3 \\
\mathbf{D}_3 = & \mathbf{D}_1 * \mathbf{X}_2 \\
\mathbf{D}_2 = & \overline{\mathbf{X}}_3 * \mathbf{X}_2 \\
\mathbf{D}_1 = & \overline{\mathbf{X}}_3 * \mathbf{X}_1
\end{array}\right.$$

e.

$$\left\{\begin{array}{ll}
\mathbf{D}_{15} = & \mathbf{D}_{14} * \mathbf{D}_{12} \\
\mathbf{D}_{15} = & \mathbf{D}_{13} * \mathbf{D}_{11} \\
\mathbf{D}_{14} = & \overline{\mathbf{X}}_2 * \mathbf{D}_{10} \\
\mathbf{D}_{13} = & \overline{\mathbf{X}}_1 * \mathbf{D}_9 \\
\mathbf{D}_{12} = & \mathbf{D}_8 * \mathbf{D}_{10} \\
\mathbf{D}_{11} = & \mathbf{D}_7 * \mathbf{D}_9 \\
\mathbf{D}_{10} = & \mathbf{D}_6 * \mathbf{C}_1 \\
\mathbf{D}_9 = & \mathbf{D}_5 * \mathbf{C}_1 \\
\mathbf{D}_8 = & \mathbf{D}_6 * \mathbf{D}_4 \\
\mathbf{D}_7 = & \mathbf{D}_5 * \mathbf{D}_3 \\
\mathbf{D}_6 = & \mathbf{C}_0 * \mathbf{D}_2 \\
\mathbf{D}_5 = & \mathbf{C}_0 * \mathbf{D}_1 \\
\mathbf{D}_4 = & \mathbf{D}_2 * \mathbf{X}_3 \\
\mathbf{D}_3 = & \mathbf{D}_1 * \mathbf{X}_3 \\
\mathbf{D}_2 = & \overline{\mathbf{X}}_3 * \mathbf{X}_2 \\
\mathbf{D}_1 = & \overline{\mathbf{X}}_3 * \mathbf{X}_1
\end{array}\right.$$

f.

$$\left\{\begin{array}{ll}
\mathbf{D}_{12} = & \mathbf{D}_{11} * \mathbf{D}_9 \\
\mathbf{D}_{12} = & \mathbf{D}_{10} * \mathbf{D}_8 \\
\mathbf{D}_{11} = & \overline{\mathbf{X}}_2 * \mathbf{D}_7 \\
\mathbf{D}_{10} = & \overline{\mathbf{X}}_1 * \mathbf{D}_7 \\
\mathbf{D}_9 = & \mathbf{D}_6 * \mathbf{D}_7 \\
\mathbf{D}_8 = & \mathbf{D}_5 * \mathbf{D}_7 \\
\mathbf{D}_7 = & \mathbf{D}_4 * \mathbf{C}_1 \\
\mathbf{D}_6 = & \mathbf{D}_4 * \mathbf{D}_3 \\
\mathbf{D}_5 = & \mathbf{D}_4 * \mathbf{D}_2 \\
\mathbf{D}_4 = & \mathbf{C}_0 * \mathbf{D}_1 \\
\mathbf{D}_3 = & \mathbf{D}_1 * \mathbf{X}_4 \\
\mathbf{D}_2 = & \mathbf{D}_1 * \mathbf{X}_3 \\
\mathbf{D}_1 = & \overline{\mathbf{X}}_4 * \mathbf{X}_2 \\
\mathbf{D}_1 = & \overline{\mathbf{X}}_3 * \mathbf{X}_1
\end{array}\right.$$

g.

$$\left\{\begin{array}{ll}
\mathbf{D}_{12} = & \mathbf{D}_{11} * \mathbf{D}_{10} \\
\mathbf{D}_{11} = & \overline{\mathbf{X}}_2 * \mathbf{D}_9 \\
\mathbf{D}_{11} = & \overline{\mathbf{X}}_1 * \mathbf{D}_8 \\
\mathbf{D}_{10} = & \mathbf{D}_7 * \mathbf{D}_9 \\
\mathbf{D}_{10} = & \mathbf{D}_6 * \mathbf{D}_8 \\
\mathbf{D}_9 = & \mathbf{D}_5 * \mathbf{C}_1 \\
\mathbf{D}_8 = & \mathbf{D}_4 * \mathbf{C}_1 \\
\mathbf{D}_7 = & \mathbf{D}_5 * \mathbf{D}_3 \\
\mathbf{D}_6 = & \mathbf{D}_4 * \mathbf{D}_3 \\
\mathbf{D}_5 = & \mathbf{C}_0 * \mathbf{X}_2 \\
\mathbf{D}_4 = & \mathbf{C}_0 * \mathbf{D}_1 \\
\mathbf{D}_3 = & \mathbf{D}_2 * \mathbf{X}_4 \\
\mathbf{D}_3 = & \mathbf{D}_1 * \mathbf{X}_3 \\
\mathbf{D}_2 = & \overline{\mathbf{X}}_4 * \mathbf{X}_2 \\
\mathbf{D}_1 = & \overline{\mathbf{X}}_3 * \mathbf{X}_1
\end{array}\right.$$

h.

Table 3.16: Concrete systems (e. – h.) of quasigroup equations that can give collisions in the compression function $\mathcal{R}$

$$\left\{\begin{array}{ll}
\mathbf{D}_{14} = & \mathbf{D}_{13} * \mathbf{D}_{11} \\
\mathbf{D}_{14} = & \mathbf{D}_{12} * \mathbf{D}_{10} \\
\mathbf{D}_{13} = & \overline{\mathbf{X}}_2 * \mathbf{D}_9 \\
\mathbf{D}_{12} = & \overline{\mathbf{X}}_1 * \mathbf{D}_8 \\
\mathbf{D}_{11} = & \mathbf{D}_7 * \mathbf{D}_9 \\
\mathbf{D}_{10} = & \mathbf{D}_6 * \mathbf{D}_8 \\
\mathbf{D}_9 = & \mathbf{D}_5 * \mathbf{C}_1 \\
\mathbf{D}_8 = & \mathbf{D}_4 * \mathbf{C}_1 \\
\mathbf{D}_7 = & \mathbf{D}_5 * \mathbf{D}_3 \\
\mathbf{D}_6 = & \mathbf{D}_4 * \mathbf{D}_3 \\
\mathbf{D}_5 = & \mathbf{C}_0 * \mathbf{D}_2 \\
\mathbf{D}_4 = & \mathbf{C}_0 * \mathbf{D}_1 \\
\mathbf{D}_3 = & \mathbf{D}_2 * \mathbf{X}_4 \\
\mathbf{D}_3 = & \mathbf{D}_1 * \mathbf{X}_3 \\
\mathbf{D}_2 = & \overline{\mathbf{X}}_4 * \mathbf{X}_2 \\
\mathbf{D}_1 = & \overline{\mathbf{X}}_3 * \mathbf{X}_1
\end{array}\right.$$

i.

$$\left\{\begin{array}{ll}
\mathbf{D}_{14} = & \mathbf{D}_{13} * \mathbf{D}_{11} \\
\mathbf{D}_{14} = & \mathbf{D}_{12} * \mathbf{D}_{10} \\
\mathbf{D}_{13} = & \overline{\mathbf{X}}_2 * \mathbf{D}_9 \\
\mathbf{D}_{12} = & \overline{\mathbf{X}}_1 * \mathbf{D}_8 \\
\mathbf{D}_{11} = & \mathbf{D}_7 * \mathbf{D}_9 \\
\mathbf{D}_{10} = & \mathbf{D}_7 * \mathbf{D}_8 \\
\mathbf{D}_9 = & \mathbf{D}_6 * \mathbf{C}_1 \\
\mathbf{D}_8 = & \mathbf{D}_5 * \mathbf{C}_1 \\
\mathbf{D}_7 = & \mathbf{D}_6 * \mathbf{D}_4 \\
\mathbf{D}_7 = & \mathbf{D}_5 * \mathbf{D}_3 \\
\mathbf{D}_6 = & \mathbf{C}_0 * \mathbf{D}_2 \\
\mathbf{D}_5 = & \mathbf{C}_0 * \mathbf{D}_1 \\
\mathbf{D}_4 = & \mathbf{D}_2 * \mathbf{X}_4 \\
\mathbf{D}_3 = & \mathbf{D}_1 * \mathbf{X}_3 \\
\mathbf{D}_2 = & \overline{\mathbf{X}}_4 * \mathbf{X}_2 \\
\mathbf{D}_1 = & \overline{\mathbf{X}}_3 * \mathbf{X}_1
\end{array}\right.$$

j.

$$\left\{\begin{array}{ll}
\mathbf{D}_{13} = & \mathbf{D}_{12} * \mathbf{D}_{11} \\
\mathbf{D}_{12} = & \overline{\mathbf{X}}_2 * \mathbf{D}_{10} \\
\mathbf{D}_{12} = & \overline{\mathbf{X}}_1 * \mathbf{D}_9 \\
\mathbf{D}_{11} = & \mathbf{D}_8 * \mathbf{D}_{10} \\
\mathbf{D}_{11} = & \mathbf{D}_7 * \mathbf{D}_9 \\
\mathbf{D}_{10} = & \mathbf{D}_6 * \mathbf{C}_1 \\
\mathbf{D}_9 = & \mathbf{D}_5 * \mathbf{C}_1 \\
\mathbf{D}_8 = & \mathbf{D}_6 * \mathbf{D}_4 \\
\mathbf{D}_7 = & \mathbf{D}_5 * \mathbf{D}_3 \\
\mathbf{D}_6 = & \mathbf{C}_0 * \mathbf{D}_2 \\
\mathbf{D}_5 = & \mathbf{C}_0 * \mathbf{D}_1 \\
\mathbf{D}_4 = & \mathbf{D}_2 * \mathbf{X}_4 \\
\mathbf{D}_3 = & \mathbf{D}_1 * \mathbf{X}_3 \\
\mathbf{D}_2 = & \overline{\mathbf{X}}_4 * \mathbf{X}_2 \\
\mathbf{D}_1 = & \overline{\mathbf{X}}_3 * \mathbf{X}_1
\end{array}\right.$$

k.

$$\left\{\begin{array}{ll}
\mathbf{D}_{15} = & \mathbf{D}_{14} * \mathbf{D}_{12} \\
\mathbf{D}_{15} = & \mathbf{D}_{13} * \mathbf{D}_{11} \\
\mathbf{D}_{14} = & \overline{\mathbf{X}}_2 * \mathbf{D}_{10} \\
\mathbf{D}_{13} = & \overline{\mathbf{X}}_1 * \mathbf{D}_9 \\
\mathbf{D}_{12} = & \mathbf{D}_8 * \mathbf{D}_{10} \\
\mathbf{D}_{11} = & \mathbf{D}_7 * \mathbf{D}_9 \\
\mathbf{D}_{10} = & d6 * \mathbf{C}_1 \\
\mathbf{D}_9 = & \mathbf{D}_5 * \mathbf{C}_1 \\
\mathbf{D}_8 = & \mathbf{D}_6 * \mathbf{D}_4 \\
\mathbf{D}_7 = & \mathbf{D}_5 * \mathbf{D}_3 \\
\mathbf{D}_6 = & \mathbf{C}_0 * \mathbf{D}_2 \\
\mathbf{D}_5 = & \mathbf{C}_0 * \mathbf{D}_1 \\
\mathbf{D}_4 = & \mathbf{D}_2 * \mathbf{X}_4 \\
\mathbf{D}_3 = & \mathbf{D}_1 * \mathbf{X}_3 \\
\mathbf{D}_2 = & \overline{\mathbf{X}}_4 * \mathbf{X}_2 \\
\mathbf{D}_1 = & \overline{\mathbf{X}}_3 * \mathbf{X}_1
\end{array}\right.$$

l.

Table 3.17: Concrete systems (i. – l.) of quasigroup equations that can give collisions in the compression function $\mathcal{R}$

3.12 Used constants in EDON-$\mathcal{R}$ - avoiding fixed points for the compression function $\mathcal{R}$

The compression function of the earlier hash function Edon-R(n) had one known fixed point, since it was true that $\mathcal{R}_1(\mathbf{0}) = \mathbf{0}$, where $\mathbf{0}$ is the vector of all zero elements.

In order to avoid the existence of some trivial fixed points, in EDON-$\mathcal{R}$ we are using the constants `0x55555555` and `0xAAAAAAAA` for the 224/256 version, and the constants `0x5555555555555555` and `0xAAAAAAAAAAAAAAAA` for the 384/512 version in the affine bijective transformations $\widehat{\mathbb{A}_1}$ and $\widehat{\mathbb{A}_3}$. The reason why we chose these constants is that they are represented as sequences of alternating 0s and 1s. Having this constants in the affine bijective transformations $\widehat{\mathbb{A}_1}$ and $\widehat{\mathbb{A}_3}$ we are not aware of any point X such that

$$\mathcal{R}(\mathbf{X}) = \mathbf{X}.$$

Moreover, examining one way functions $\mathcal{R}$ defined with words of much smaller size $w = 2, 3, 4, 5$, lead us to the conclusion that finding fixed points either for the quasigroups of orders 2^{256} and 2^{512} (the case $\mathbf{X} *_q \mathbf{X} = \mathbf{X}$) or for the compression function $\mathcal{R}$ is infeasible.

3.13 Getting all the additions to behave as XORs

Having a compression function $\mathcal{R}$ defined only by additions modulo 2^{32} or modulo 2^{64}, XORs and left rotations, it is a natural idea to try to find values for which additions in $\mathcal{R}$ behave as XORs [31].

In such a case, one would have a completely linear system in the ring $(\mathbb{Z}_2^n, +, \times)$ for which collisions, preimages and second preimages can easily be found. However, getting all the additions to behave as XORs is a challenge.

Here we can point out several significant works that are related with analysis of differential probabilities of operations that combine additions modulo 2^{32}, XORs and left rotations. In 1993 Berson has made a differential cryptanalysis of addition modulo 2^{32} and applied it on MD5 [32]. In 2001 Lipmaa and Moriai have constructed efficient algorithms for computing differential properties of addition modulo 2^w (for general values of w) [33], and in 2004 Lipmaa, Wallén and Dumas have constructed a linear-time algorithm for computing the additive differential probability of exclusive-or [34].

All of these works are determining the additive differential probability of exclusive-or:

$$Pr[((x+\alpha)\oplus(y+\beta))-(x\oplus y)=\gamma]$$

and the exclusive-or differential probability of addition:

$$Pr[((x\oplus\alpha)+(y\oplus\beta))\oplus(x+y)=\gamma]$$

where probability is computed for all pairs $(x,y)\in\mathbb{Z}_{2^w}\times\mathbb{Z}_{2^w}$ and for any predetermined triplet $(\alpha,\beta,\gamma)\in\mathbb{Z}_{2^w}\times\mathbb{Z}_{2^w}\times\mathbb{Z}_{2^w}$.

In the case of EDON-$\mathcal{R}$, instead of simple combination of two w-bit variables ($w=32$ or $w=64$) once by additions modulo 2^w then by xoring, we have a linear transformation of 8, w-bit variables described by transformations defined in Definition 5. Additionally, bearing in mind that $\mathcal{R}:\{0,1\}^{32w}\rightarrow\{0,1\}^{16w}$, in this moment we do not see how the results in [34] will help in finding concrete values of arguments for the function $\mathcal{R}$ for which additions behave as XORs.

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

Claims in this Subsection were present in the original documentation, but are not correct. Khovratovich et al. have found free-start collisions for EDON-$\mathcal{R}$ in [35].

According to the conjectured one-wayness of the function $\mathcal{R}$, iterating EDON-$\mathcal{R}$ backward is infeasible. The conjecture is again 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-$\mathcal{R}$ is 2^n hash computations.

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

We elaborate our claims more concretely by the following discussion:

Definition 16. Let $(\mathbf{A_0},\mathbf{A_1},\mathbf{X_1},\mathbf{X_2}),(\mathbf{B_0},\mathbf{B_1},\mathbf{X_3},\mathbf{X_4})\in Q_q\times Q_q\times Q_q\times Q_q$. If $\mathcal{R}(\mathbf{A_0},\mathbf{A_1},\mathbf{X_1},\mathbf{X_2})=(\mathbf{D_0},\mathbf{Y})$ and $\mathcal{R}(\mathbf{B_0},\mathbf{B_1},\mathbf{X_3},\mathbf{X_4})=(\mathbf{E_0},\mathbf{Y})$ then the pair $((\mathbf{A_0},\mathbf{A_1},\mathbf{X_1},\mathbf{X_2}),(\mathbf{B_0},\mathbf{B_1},\mathbf{X_3},\mathbf{X_4}))$ is a free start collision for EDON-$\mathcal{R}$.

	$\mathbf{A}_0$	$\mathbf{A}_1$
$\overline{\mathbf{A}}_1$	$\mathbf{X}_0^{(1)}$	$\mathbf{X}_1^{(1)}$
$\mathbf{C}_0$	$\mathbf{X}_0^{(2)}$	$\mathbf{X}_1^{(2)}$
$\mathbf{C}_1$	$\mathbf{X}_0^{(3)}$	$\mathbf{X}_1^{(3)}$
$\overline{\mathbf{A}}_0$	$\mathbf{B}_0$	$\mathbf{B}_1$

a.

	$\mathbf{X}_0$	$\mathbf{X}_1$
$\overline{\mathbf{X}}_1$	$\mathbf{X}_0^{(1)}$	$\mathbf{X}_1^{(1)}$
$\mathbf{C}_0$	$\mathbf{X}_0^{(2)}$	$\mathbf{X}_1^{(2)}$
$\mathbf{C}_1$	$\mathbf{X}_0^{(2)}$	$\mathbf{X}_1^{(2)}$
$\overline{\mathbf{X}}_0$	$\mathbf{B}_0$	$\mathbf{B}_1$

b.

Table 3.18: Situations **a.** and **b.** give a free start collision.

The free start collision situation is described in the Table 3.18. In this moment we can only see two ways to find free start collisions for EDON-$\mathcal{R}$:

1. Generate a random $\mathbf{Y} \in Q_q$. Construct vectors $(\mathbf{D_0}, \mathbf{Y})$ and $(\mathbf{E_0}, \mathbf{Y})$ where $\mathbf{D_0}, \mathbf{E_0} \in Q_q$ are randomly chosen. Chose randomly $\mathbf{A_0}$, $\mathbf{A_1}$, $\mathbf{B_0}$, $\mathbf{B_1} \in Q_q$. Try to find $\mathcal{R}^{-1}(\mathbf{D_0}, \mathbf{Y})$ with the respect of chosen $\mathbf{A_0}, \mathbf{A_1}$, i.e., find $\mathbf{X_0}$ and $\mathbf{X_1}$ such that

   $$R(\mathbf{A_0}, \mathbf{A_1}, \mathbf{X_0}, \mathbf{X_1}) = (\mathbf{D_0}, \mathbf{Y})$$

   and find $\mathcal{R}^{-1}(\mathbf{E_0}, \mathbf{Y})$ with the respect of chosen $\mathbf{B_0}, \mathbf{B_1}$ i.e., find $\mathbf{X_2}$ and $\mathbf{X_3}$ such that

   $$R(\mathbf{B_0}, \mathbf{B_1}, \mathbf{X_2}, \mathbf{X_3}) = (\mathbf{E_0}, \mathbf{Y}).$$

2. Generate a random $(\mathbf{A_0}, \mathbf{A_1}, \mathbf{X_0}, \mathbf{X_1})$ and compute $\mathcal{R}(\mathbf{A_0}, \mathbf{A_1}, \mathbf{X_0}, \mathbf{X_1}) = (\mathbf{D_0}, \mathbf{Y})$. Construct vector $(\mathbf{E_0}, \mathbf{Y})$ where $\mathbf{E_0} \in Q_q$ is randomly chosen. Chose randomly $\mathbf{B_0}$, $\mathbf{B_1} \in Q_q$. Try to find $\mathcal{R}^{-1}(\mathbf{E_0}, \mathbf{Y})$ with the respect of chosen $\mathbf{B_0}$, $\mathbf{B_1}$ i.e., find $\mathbf{X_2}$ and $\mathbf{X_3}$ such that

   $$R(\mathbf{B_0}, \mathbf{B_1}, \mathbf{X_2}, \mathbf{X_3}) = (\mathbf{E_0}, \mathbf{Y}).$$

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

3.15 Statement about the cryptographic strength of EDON-$\mathcal{R}$

In summary, we can say that the design of EDON-$\mathcal{R}$ heavily uses combinations of bitwise operations of XORing, rotating and operations of addition in $\mathbb{Z}_{2^{32}}$ or in $\mathbb{Z}_{2^{64}}$ (which are mutually nonlinear operations). This strategy, combined with the conjectured one-wayness of the function $\mathcal{R}$

(reverse quasigroup string transformation) and the good differential properties of the underlying quasigroup operations used in $\mathcal{R}$ are the cornerstones of the EDON-$\mathcal{R}$ strength.

According to all this, we give a statement of the cryptographic strength of EDON-$\mathcal{R}$ against attacks for finding collisions, preimages, second preimages, the resistance to length-extension attacks, the resistance to multicollision attacks and the provable resistance to differential cryptanalysis which is summarized in Table 3.19. We also formally state that any m-bit hash function specified by taking a fixed subset of the EDON-$\mathcal{R}$s output bits meets the properties summarized in Table 3.19 when n is replaced by m.

Algorithm abbreviation	Digest size n (in bits)	Work factor for finding collision	Work factor for finding a preimage	Work factor for finding a second preimage of a message shorter than 2^k bits	Resistance to length-extension attacks	Resistance to multicollision attacks	Provable resistance to differential cryptanalysis
Edon-$\mathcal{R}$224	224	$\approx 2^{112}$	$\approx 2^{224}$	$\approx 2^{224-k}$	Yes	Yes	Yes
Edon-$\mathcal{R}$256	256	$\approx 2^{128}$	$\approx 2^{256}$	$\approx 2^{256-k}$	Yes	Yes	Yes
Edon-$\mathcal{R}$384	384	$\approx 2^{192}$	$\approx 2^{384}$	$\approx 2^{384-k}$	Yes	Yes	Yes
Edon-$\mathcal{R}$512	512	$\approx 2^{256}$	$\approx 2^{512}$	$\approx 2^{512-k}$	Yes	Yes	Yes

Table 3.19: Cryptographic strength of the EDON-$\mathcal{R}$

3.16 EDON-$\mathcal{R}$ support of HMAC

EDON-$\mathcal{R}$ is an iterative cryptographic hash function. Thus, in combination with a shared secret key it can be used in the HMAC standard as it is defined in [36–38].

As the cryptographic strength of HMAC depends on the properties of the underlying hash function, and the conjectured cryptographic strength of EDON-$\mathcal{R}$ is claimed in the Section 3.15 here we give a formal statement that EDON-$\mathcal{R}$ can be securely used with the HMAC.

In what follows we are giving 4 examples for every digest size of 224, 256, 384 and 512 bits.

Edon-R224-MAC Test Examples

```
Key:
00010203 04050607 08090A0B 0C0D0E0F 10111213 14151617 18191A1B 1C1D1E1F
20212223 24252627 28292A2B 2C2D2E2F 30313233 34353637 38393A3B 3C3D3E3F
Key_length:  64
Data:
'Sample #1'
Data_length:  9
HMAC:
71C8F2BF D3CCF225 69FC93F7 9F48750E E48620F0 003F6BEB 2A3CE6AC


Key:
30313233 34353637 38393A3B 3C3D3E3F 40414243
Key_length:  20
Data:
'Sample #2'
Data_length:  9
HMAC:
8B40C47F A7FBCE0E 6BF303A8 5F37FDD0 B48B2B9E 55F5F6FB F41BE6DF
```

```
Key:
50515253 54555657 58595A5B 5C5D5E5F 60616263 64656667 68696A6B 6C6D6E6F
70717273 74757677 78797A7B 7C7D7E7F 80818283 84858687 88898A8B 8C8D8E8F
90919293 94959697 98999A9B 9C9D9E9F A0A1A2A3 A4A5A6A7 A8A9AAAB ACADAEAF
B0B1B2B3
Key_length:  100
Data:
'The successful verification of a MAC does not completely guarantee
 that the accompanying message is authentic.'
Data_length:  110
HMAC:
E16AC9B5 4B998A93 4A39C27A A48EE2A4 95661062 CAA03DBE CDA70D21


Key:
50515253 54555657 58595A5B 5C5D5E5F 60616263 64656667 68696A6B 6C6D6E6F
70717273 74757677 78797A7B 7C7D7E7F 80818283 84858687 88898A8B 8C8D8E8F
90919293 94959697 98999A9B 9C9D9E9F A0A1A2A3 A4A5A6A7 A8A9AAAB ACADAEAF
B0B1B2B3
Key_length:   100
Data:
'The successful verification of a MAC does not completely guarantee
 that the accompanying message is authentic: there is a chance that
 a source with no knowledge of the key can present a purported MAC.'
Data_length:  200
HMAC:
92E297FF 91BB08B9 B2C68F6D 2DA82F33 353CA617 C721DA0B FC03D801
```

Edon-R256-MAC Test Examples

```
Key:
00010203 04050607 08090A0B 0C0D0E0F 10111213 14151617 18191A1B 1C1D1E1F
20212223 24252627 28292A2B 2C2D2E2F 30313233 34353637 38393A3B 3C3D3E3F
Key_length:  64
Data:
'Sample #1'
Data_length:  9
HMAC:
3673D502 83A2D52D FC0BB839 27D30386 2AC52BB9 707199D8 9A481125 6604D76B
```

```
Key:
30313233 34353637 38393A3B 3C3D3E3F 40414243
Key_length:  20
Data:
'Sample #2'
Data_length:  9
HMAC:
B9B58E62 5AF9E85D 46DDCC77 70341B32 8FD85619 3A4EAA20 E8F2D02F FE81FCEE
```

```
Key:
50515253 54555657 58595A5B 5C5D5E5F 60616263 64656667 68696A6B 6C6D6E6F
70717273 74757677 78797A7B 7C7D7E7F 80818283 84858687 88898A8B 8C8D8E8F
90919293 94959697 98999A9B 9C9D9E9F A0A1A2A3 A4A5A6A7 A8A9AAAB ACADAEAF
B0B1B2B3
Key_length:  100
Data:
'The successful verification of a MAC does not completely guarantee
 that the accompanying message is authentic.'
Data_length:  110
HMAC:
FC4B7A90 A8F5326F EADE94C5 5581AB5F 138B57DF 93B89A09 4D684C1D 76240C43
```

```
Key:
50515253 54555657 58595A5B 5C5D5E5F 60616263 64656667 68696A6B 6C6D6E6F
70717273 74757677 78797A7B 7C7D7E7F 80818283 84858687 88898A8B 8C8D8E8F
90919293 94959697 98999A9B 9C9D9E9F A0A1A2A3 A4A5A6A7 A8A9AAAB ACADAEAF
B0B1B2B3
Key_length:   100
Data:
'The successful verification of a MAC does not completely guarantee
 that the accompanying message is authentic: there is a chance that
 a source with no knowledge of the key can present a purported MAC.'
Data_length:  200
HMAC:
7CADF7EC F31BB57E 75030676 D1B38877 89CFDF1B E2DBECDC 75FE32E1 3982789D
```

Edon-R384-MAC Test Examples

```
Key:
0001020304050607 08090A0B0C0D0E0F 1011121314151617 18191A1B1C1D1E1F
2021222324252627 28292A2B2C2D2E2F 3031323334353637 38393A3B3C3D3E3F
Key_length:  64
Data:
'Sample #1'
Data_length:  9
HMAC:
0CB056EE9989BF7D 0EA50F1402992521 0683B5753965ABC6 F2C9353492234BF7
FF68D7F45AAD286F 5360E5BA091DA415


Key:
3031323334353637 38393A3B3C3D3E3F 40414243
Key_length:  20
Data:
'Sample #2'
Data_length:  9
HMAC:
5F49AF0398A7B853 A3EC7BAB13CD003E E6D6540A0B8BC5B6 6AAEE3893396D046
BA6C1290DC5DD1B2 9394E0993E2512EB
```

```
Key:
5051525354555657 58595A5B5C5D5E5F 6061626364656667 68696A6B6C6D6E6F
7071727374757677 78797A7B7C7D7E7F 8081828384858687 88898A8B8C8D8E8F
9091929394959697 98999A9B9C9D9E9F A0A1A2A3A4A5A6A7 A8A9AAABACADAEAF
B0B1B2B350515253 5455565758595A5B 5C5D5E5F60616263 6465666768696A6B
6C6D6E6F70717273 7475767778797A7B 7C7D7E7F80818283 8485868788898A8B
8C8D8E8F90919293 9495969798999A9B 9C9D9E9FA0A1A2A3 A4A5A6A7A8A9AAAB
ACADAEAFB0B1B2B3
Key_length:  200
Data:
'The successful verification of a MAC does not completely guarantee
 that the accompanying message is authentic.'
Data_length:  110
HMAC:
5B964E967CC77A65 5649A13193C72D36 E15E5D61E3171695 2FD29E265E33A1DE
AF0D7BFEB3B27557 C09FCF450FF5E4BD


Key:
5051525354555657 58595A5B5C5D5E5F 6061626364656667 68696A6B6C6D6E6F
7071727374757677 78797A7B7C7D7E7F 8081828384858687 88898A8B8C8D8E8F
9091929394959697 98999A9B9C9D9E9F A0A1A2A3A4A5A6A7 A8A9AAABACADAEAF
B0B1B2B3
Key_length:   100
Data:
'The successful verification of a MAC does not completely guarantee
 that the accompanying message is authentic: there is a chance that
 a source with no knowledge of the key can present a purported MAC.'
Data_length:  200
HMAC:
46DF6287B91B433E 421F8594D41F8501 BA074BC4C64CE0D4 8D19A5D18EE823AD
7DF488C780D54C31 20B86CD728C20A16
```

Edon-R512-MAC Test Examples

```
Key:
0001020304050607 08090A0B0C0D0E0F 1011121314151617 18191A1B1C1D1E1F
2021222324252627 28292A2B2C2D2E2F 3031323334353637 38393A3B3C3D3E3F
Key_length:  64
Data:
'Sample #1'
Data_length:  9
HMAC:
B07398A705AFF818 E332E03C788CF8C7 A5BD347CB5ED2728 1B73977716952745
CB25CFDEA9D0AD43 201160E2E96BB42D 91DA7544B2D83D64 6259DBABE3DAFC69


Key:
3031323334353637 38393A3B3C3D3E3F 40414243
Key_length:  20
Data:
'Sample #2'
Data_length:  9
HMAC:
CEDC261D9B89C31F C84B7C66C7137F96 1622FFDE2B5715B9 D3B85FFD919F4A05
4D0BBA7F6F68CE2E 7D66150DC30B5002 2B308F89BABACDDD 14680E59991E2D9A
```

```
Key:
5051525354555657 58595A5B5C5D5E5F 6061626364656667 68696A6B6C6D6E6F
7071727374757677 78797A7B7C7D7E7F 8081828384858687 88898A8B8C8D8E8F
9091929394959697 98999A9B9C9D9E9F A0A1A2A3A4A5A6A7 A8A9AAABACADAEAF
B0B1B2B350515253 5455565758595A5B 5C5D5E5F60616263 6465666768696A6B
6C6D6E6F70717273 7475767778797A7B 7C7D7E7F80818283 8485868788898A8B
8C8D8E8F90919293 9495969798999A9B 9C9D9E9FA0A1A2A3 A4A5A6A7A8A9AAAB
ACADAEAFB0B1B2B3
Key_length:  200
Data:
'The successful verification of a MAC does not completely guarantee
 that the accompanying message is authentic.'
Data_length:  110
HMAC:
1F1A500781AD0D6F 0EE5862713459C3D 06A9CB2E5750D834 E82B07900F5D1253
558E527DF778DF74 D0819CB727343D34 50DFB4D037921D56 7874E098459325FE


Key:
5051525354555657 58595A5B5C5D5E5F 6061626364656667 68696A6B6C6D6E6F
7071727374757677 78797A7B7C7D7E7F 8081828384858687 88898A8B8C8D8E8F
9091929394959697 98999A9B9C9D9E9F A0A1A2A3A4A5A6A7 A8A9AAABACADAEAF
B0B1B2B3
Key_length:   100
Data:
'The successful verification of a MAC does not completely guarantee
 that the accompanying message is authentic: there is a chance that
 a source with no knowledge of the key can present a purported MAC.'
Data_length:  200
HMAC:
A69CA444D01E741C B7683A66C060D8C0 3F33A57E62BA50B8 80DF0A63900C6C36
BEE535444C6EBB31 78F48F551D5F2447 87AD07F07E96A4B3 781A9E5EFB625F28
```

3.17 EDON-$\mathcal{R}$ support of randomized hashing

EDON-$\mathcal{R}$ can be used in the randomizing scheme proposed in [39, 40].

3.18 Resistance to SHA-2 attacks

EDON-$\mathcal{R}$ is designed to have a security strength that is at least as good as the hash algorithms currently specified in FIPS 180-2, and this security strength is achieved with significantly improved efficiency. Having in mind the fact that EDON-$\mathcal{R}$ design differs completely from the design of SHA-2 family of hash functions, we claim that any possibly successful attack on SHA-2 family of hash functions is unlikely to be applicable to EDON-$\mathcal{R}$.

CHAPTER 4

Estimated Computational Efficiency and Memory Requirements

4.1 Speed of EDON-$\mathcal{R}$ on NIST SHA-3 Reference Platform

We have developed and measured the performances of EDON-$\mathcal{R}$ on a platform with the following characteristics:

CPU: Intel Core 2 Duo,

Clock speed: 2.4 GHz,

Memory: 4GB RAM,

Operating system: Windows Vista Enterprise 64-bit (x64) Edition with Service Pack 1,

Compiler: ANSI C compiler in the Microsoft Visual Studio 2005 Professional Edition.

Compiler: ANSI C compiler in the Intel C++ v 11.0.066.

For measuring the speed of the hash function expressed as cycles/byte we have used the `rdtsc()` function and a modified version of a source code that was given to us by Dr. Brian Gladman from his optimized realization of SHA-2 hash function [41].

4.1.1 Speed of the Optimized 32–bit version of EDON-$\mathcal{R}$

In the Table 4.1 we are giving the speed of all four instances of EDON-$\mathcal{R}$ for the optimized 32–bit version obtained by Microsoft Visual Studio 2005 Professional Edition.

	Speed in cycles/byte for different lengths (in bytes) of the digested message.					
MD Size	1	10	100	1000	10,000	100,000
224	2497.00	249.70	43.21	9.01	8.66	8.64
256	781.00	76.90	13.45	9.01	8.66	8.64
384	2161.00	214.90	21.85	19.16	15.12	14.63
512	2161.00	216.10	21.97	19.19	15.13	14.63

Table 4.1: The performance of optimized 32–bit version of Edon-$\mathcal{R}$ in machine cycles per data byte on Intel Core 2 Duo for different hash data lengths, obtained by Microsoft Visual Studio 2005 Professional Edition.

	Speed in cycles/byte for different lengths (in bytes) of the digested message.					
MD Size	1	10	100	1000	10,000	100,000
224	637.00	63.70	11.41	7.19	6.34	6.26
256	589.00	58.90	10.69	6.64	6.33	6.27
384	1465.00	147.70	14.77	11.59	10.18	9.99
512	1501.00	148.90	14.89	12.57	10.25	10.04

Table 4.2: The performance of optimized 32–bit version of Edon-$\mathcal{R}$ in machine cycles per data byte on Intel Core 2 Duo for different hash data lengths, obtained by Intel C++ v 11.0.066.

In the Table 4.2 we are giving the speed of all four instances of Edon-$\mathcal{R}$ for the optimized 32–bit version obtained by Intel C++ v 11.0.066.

4.1.2 Speed of the Optimized 64–bit version of Edon-$\mathcal{R}$

In the Table 4.3 we are giving the speed of all four instances of Edon-$\mathcal{R}$ for the optimized 64–bit version.

In the Table 4.4 we are giving the speed of all four instances of Edon-$\mathcal{R}$ for the optimized 64–bit version.

	Speed in cycles/byte for different lengths (in bytes) of the digested message.					
MD Size	1	10	100	1000	10,000	100,000
224	1729.00	172.90	8.89	5.71	5.54	5.52
256	541.00	54.10	8.89	5.79	5.55	5.37
384	601.00	55.30	5.65	2.98	2.78	2.71
512	601.00	55.30	5.77	2.98	2.78	2.71

Table 4.3: The performance of optimized 64–bit version of EDON-$\mathcal{R}$ in machine cycles per data byte on Intel Core 2 Duo for different hash data lengths, obtained by Microsoft Visual Studio 2005 Professional Edition.

	Speed in cycles/byte for different lengths (in bytes) of the digested message.					
MD Size	1	10	100	1000	10,000	100,000
224	469.00	46.90	8.05	4.86	4.69	4.40
256	457.00	45.70	7.69	4.90	4.69	4.40
384	493.00	49.30	4.93	2.47	2.30	2.29
512	493.00	49.30	4.93	2.52	2.30	2.29

Table 4.4: The performance of optimized 64–bit version of EDON-$\mathcal{R}$ in machine cycles per data byte on Intel Core 2 Duo for different hash data lengths, obtained by Intel C++ v 11.0.066.

4.2 Memory requirements of EDON-$\mathcal{R}$ on NIST SHA-3 Reference Platform

When processing the message block $M^{(i)} = (M_0^{(i)}, M_1^{(i)}, \ldots, M_{15}^{(i)})$, we need the current value of the double pipe $P^{(i-1)} = (P_0^{(i-1)}, P_1^{(i-1)}, \ldots, P_{15}^{(i-1)})$, the values of the new double pipe – in the reference source code indexed as $P^{(i)} = (P_{16}^{(i)}, P_{17}^{(i)}, \ldots, P_{31}^{(i)})$ and 16 temporary variables (in the reference source code denoted as `t0,...,t15`).

The need of memory is thus:

- 16 words of $M^{(i)}$,
- 32 words of $P^{(i)}$.
- 16 temporary words `t0,...,t15`.

which is in total 64 words. That means that **EDON-$\mathcal{R}$224 and EDON-$\mathcal{R}$256 use 256 bytes** and **EDON-$\mathcal{R}$384 and EDON-$\mathcal{R}$512 use 512 bytes**.

4.3 Estimates for efficiency and memory requirements on 8-bit processors

We have used 8-bit Atmel processors ATmega16 and ATmega406 to test the implementation and performance of the compression function of the two main representatives of the EDON-$\mathcal{R}$ hash function: Edon-$\mathcal{R}$256 and Edon-$\mathcal{R}$512. We have used WinAVR – an open source software development tools for the Atmel AVR series of RISC microprocessors and for simulation we have used the AVR Studio v 4.14. In Table 4.5 we are giving the length of the produced executable code and the speed in number of cycles per byte.

Name	Code size (.text + .data + .bootloader) in bytes	Speed (cycles/byte)	8–bit MCU
Edon-$\mathcal{R}$224/256	6002	616	ATmega16
Edon-$\mathcal{R}$384/512	38798	1857	ATmega406

Table 4.5: The size and the speed of code for the compression functions for Edon-$\mathcal{R}$224/256 and Edon-$\mathcal{R}$384/512

From the analysis of the produced executable code we can project that by direct assembler programming EDON-$\mathcal{R}$ can be implemented in less than 3 Kbytes (Edon-$\mathcal{R}$256) and in less than 16 KBytes (Edon-$\mathcal{R}$512) but this claim will have to be confirmed in the forthcoming period during the NIST competition.

4.4 Estimates for a Compact Hardware Implementation

We are giving the estimates for the compact hardware implementation of the compression function of EDON-$\mathcal{R}$ in Table 4.6 having in mind the minimal memory requirements described in Section 4.2. Namely, since for

Name	Estimated gate count for the needed memory	Estimated gate count for the algorithm logic	Estimated minimal total gate count
Edon-$\mathcal{R}$224/256	12,288	≈1,000	≈13,288
Edon-$\mathcal{R}$384/512	24,576	≈2,000	≈26,576

Table 4.6: Estimated number of logic gates for realization of the compression functions for Edon-$\mathcal{R}$224/256 and Edon-$\mathcal{R}$384/512

4.5 Internal Parallelizability of EDON-$\mathcal{R}$

The design of EDON-$\mathcal{R}$ allows very high level of parallelization in computation of its compression function. This parallelism can be achieved by using specifically designed hardware, and indeed with the advent of multicore CPUs, those parts can be computed in different cores in parallel. From the specification given below, we claim that EDON-$\mathcal{R}$ can be computed after 5 "parallel" steps. Of course those 5 "parallel" steps have different hardware specification and different complexity, but can serve as a general measure of the parallelizability of EDON-$\mathcal{R}$. The high level parallel specification of EDON-$\mathcal{R}$ according to the specification of $\mathcal{R}$ given in Table 2.5 is as follows:

Computing $\mathcal{R}(\mathbf{C}_0, \mathbf{C}_1, \mathbf{A}_0, \mathbf{A}_1)$

Step 1: Compute $\mathbf{X}_0^{(1)} = \overline{\mathbf{A}}_1 * \mathbf{A}_0$

Step 2: Compute in parallel:

- $\mathbf{X}_1^{(1)} = \mathbf{X}_0^{(1)} * \mathbf{A}_1$

- $\mathbf{X}_0^{(2)} = \mathbf{C}_0 * \mathbf{X}_0^{(1)}$

Step 3: Compute in parallel:

- $\mathbf{X}_1^{(2)} = \mathbf{X}_0^{(2)} * \mathbf{X}_1^{(1)}$
- $\mathbf{X}_0^{(3)} = \mathbf{X}_0^{(2)} * \mathbf{C}_1$

Step 4: Compute in parallel:

- $\mathbf{X}_1^{(3)} = \mathbf{X}_1^{(2)} * \mathbf{X}_0^{(3)}$
- $\mathbf{B}_0 = \overline{\mathbf{A}}_0 * \mathbf{X}_0^{(3)}$

Step 5: Compute $\mathbf{B}_1 = \mathbf{B}_0 * \mathbf{X}_1^{(3)}$

Internally, every quasigroup operation $\mathbf{X} * \mathbf{Y}$ can be further parallelized and computed in 4 parallel steps. The two permutations $\pi_2 \equiv \widehat{\mathbb{A}}_1 \circ ROTL^{\mathbf{r}_{1,m}} \circ \mathbb{A}_2$ and $\pi_3 \equiv \widehat{\mathbb{A}}_3 \circ ROTL^{\mathbf{r}_{2,m}} \circ \mathbb{A}_4$ where $m = 32, 64$ can be computed in 3 parallel steps, and one step is needed for the mutual xoring. Those steps are:

Computing $\mathbf{Z} = \mathbf{X} * \mathbf{Y}$

Step 1: Compute in parallel:

- $\mathbf{Temp_1} \leftarrow \widehat{\mathbb{A}}_1(\mathbf{X})$
- $\mathbf{Temp_2} \leftarrow \widehat{\mathbb{A}}_3(\mathbf{Y})$

Step 2: Compute in parallel:

- $\mathbf{Temp_3} \leftarrow ROTL^{\mathbf{r}_{1,q}}(\mathbf{Temp_1})$
- $\mathbf{Temp_4} \leftarrow ROTL^{\mathbf{r}_{2,q}}(\mathbf{Temp_2})$

Step 3: Compute in parallel:

- $\mathbf{Temp_5} \leftarrow \mathbb{A}_2(\mathbf{Temp_3})$
- $\mathbf{Temp_6} \leftarrow \mathbb{A}_4(\mathbf{Temp_4})$

Step 4: Compute $\mathbf{Z} = \mathbf{Temp_5} \oplus \mathbf{Temp_6}$

Thus, theoretically we can digest one message block by the compression function of EDON-$\mathcal{R}$ in 20 parallel steps.

CHAPTER 5

Statements

5.1 Statement by the Submitter

I, *Danilo Gligoroski*, do hereby declare that, to the best of my knowledge, the practice of the algorithm, reference implementation, and optimized implementations that I have submitted, known as EDON-$\mathcal{R}$, may be covered by the following U.S. and/or foreign patents: **NONE**.

I do hereby declare that I am aware of no patent applications that may cover the practice of my submitted algorithm, reference implementation or optimized implementations.

I do hereby understand that my submitted algorithm may not be selected for inclusion in the Secure Hash Standard. I also understand and agree that after the close of the submission period, my submission may not be withdrawn from public consideration for SHA-3. I further understand that I will not receive financial compensation from the U.S. Government for my submission. I certify that, to the best of my knowledge, I have fully disclosed all patents and patent applications relating to my algorithm. I also understand that the U.S. Government may, during the course of the lifetime of the SHS or during the FIPS public review process, modify the algorithm's specifications (e.g., to protect against a newly discovered vulnerability). Should my submission be selected for SHA-3, I hereby agree not to place any restrictions on the use of the algorithm, intending it to be available on a worldwide, non-exclusive, royalty-free basis.

I do hereby agree to provide the statements required by Sections 5.2 and 5.3, below, for any patent or patent application identified to cover the practice of my algorithm, reference implementation or optimized implementations and the right to use such implementations for the purposes of the SHA-3 evaluation process.

I understand that NIST will announce the selected algorithm(s) and proceed to publish the draft

FIPS for public comment. If my algorithm (or the derived algorithm) is not selected for SHA-3 (including those that are not selected for the second round of public evaluation), I understand that all rights, including use rights of the reference and optimized implementations, revert back to the submitter (and other owner[s, as appropriate). Additionally, should the U.S. Government not select my algorithm for SHA-3 at the time NIST ends the competition, all rights revert to the submitter (and other owners as appropriate).

Signed: Danilo Gligoroski

Title:Prof.

Dated: 12 January 2009

Place: Trondheim, Norway

5.2 Statement by Patent (and Patent Application) Owner(s)

N/A

5.3 Statement by Reference/Optimized Implementations' Owner(s)

I, *Danilo Gligoroski* am the owner of the submitted reference implementation and optimized implementations and hereby grant the U.S. Government and any interested party the right to use such implementations for the purposes of the SHA-3 evaluation process, notwithstanding that the implementations may be copyrighted.

Signed: Danilo Gligoroski
Title: Prof.
Dated: 12 January 2009
Place: Trondheim, Norway

References

[1] *Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family*, 2007. NIST. http://csrc.nist.gov/groups/ST/hash/index.html.

[2] D. Gligoroski, S. Markovski, and L. Kocarev. Edon–$\mathcal{R}$, an infinite family of cryptographic hash functions. August 2006. http://www.csrc.nist.gov/pki/HashWorkshop/2006/Papers/GLIGOROSKI_EdonR-ver06.pdf.

[3] D. Gligoroski. On a family of minimal candidate one-way functions and one-way permutations. *International Journal of Network Security*, in print 2009.

[4] V. D. Belousov. *Osnovi teorii kvazigrup i lup*. Nauka, Moskva, 1967.

[5] J. Dénes and A. D. Keedwell. A new authentication scheme based on latin squares. *Discrete Math.*, 106-107:157–161, 1992.

[6] J. D. H. Smith. *An introduction to quasigroups and their representations*. Chapman & Hall/CRC, 2007.

[7] B.D. McKay and E. Rogoyski. Latin squares of order 10. *Electronic J. Comb.*

[8] S. Markovski, D. Gligoroski, and V. Bakeva. Quasigroup string processing. In *Part 1, Contributions, Sec. Math. Tech. Sci., MANU*, volume XX, pages 13–28, 1999.

[9] D. Gligoroski. Candidate one-way functions and one-way permutations based on quasigroup string transformations. Cryptology ePrint Archive, Report 2005/352, 2005. http://eprint.iacr.org/.

[10] D. Gligoroski and S. J. Knapskog. Edon–$\mathcal{R}(256, 384, 512)$ – an efficient implementation of edon–$\mathcal{R}$ family of cryptographic hash functions. *Comment.Math.Univ.Carolin.*, 49,2:219–239, 2008.

[11] D. Gligoroski, S. Markovski, and S. J. Knapskog. The stream cipher edon80. In M. Robshaw and O. Billet, editors, *New Stream Cipher Designs*, volume 4986 of *LNCS*, pages 152–169. Springer-Verlag, 2008.

[12] D. J. Bernstein. The salsa20 family of stream ciphers. In M. Robshaw and O. Billet, editors, *New Stream Cipher Designs*, volume 4986 of *LNCS*, pages 84–97. Springer-Verlag, 2008.

[13] D. J. Wheeler and R. M. Needham. Tea, a tiny encryption algorithm. In Bart Preneel, editor, *FSE*, volume 1008 of *LNCS*, pages 363–366. Springer, 1994.

[14] X. Lai, J. L. Massey, and S. Murphy. Markov ciphers and differential cryptanalysis. In *Advances in Cryptology, CRYPTO '91*, pages 17–38. Springer-Verlag, 1991.

[15] G. Carter, E. Dawson, and L. Nielsen. A latin square variation of des. In *In Proc. Workshop of Selected Areas in Cryptography*.

[16] J. Cooper, D. Donovan, and J. Seberry. Secret sharing schemes arising from latin squares. *Bulletin of the Institute of Combinatorics and its Applications*, (4):33–43, 1194.

[17] Claus-Peter Schnorr and S. Vaudenay. Black box cryptoanalysis of hash networks based on multipermutations. In *EUROCRYPT*, pages 47–57, 1994.

[18] C. E. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28(4):656—715, 1949.

[19] K. H. Rosen, J. G. Michaels, J. L. Gross, J. W. Grossman, and D. R. Shier. *Handbook of Discrete and Combinatorial Mathematics*. CRC Press, Boca Raton, Florida, 2000.

[20] B. McKay. Web page: Latin squares - main classes of graeco-latin squares. http://cs.anu.edu.au/people/bdm/data/latin.html.

[21] I. B. Damgård. Collision free hash functions and public key signature schemes. In *Advances in Cryptology–EUROCRYPT '87*, volume 304, pages 203–216, 1987.

[22] I. B. Damgård. A design principle for hash functions. In *Advances in Cryptology–CRYPTO '89*, pages 416–427, 1990.

[23] R. C. Merkle. One way hash functions and DES. pages 428–446, 1990. Based on unpublished paper from 1979 and his Ph.D thesis, Stanford, 1979.

[24] S. Lucks. Design principles for iterated hash functions. Cryptology ePrint Archive, Report 2004/253, 2004. http://eprint.iacr.org/.

[25] S. Lucks. A failure-friendly design principle for hash functions. In *ASIACRYPT*, pages 474–494, 2005.

[26] J.-S. Coron, Y. Dodis, C. Malinaud, and P. Puniya. Merkle–Damgård revisited: How to construct a hash function. In *Advances in cryptology: CRYPTO 2005*, volume 3621 of *LNCS*, pages 430–440. Springer-Verlag, 2005.

[27] A. Joux. Multicollisions in iterated hash functions. application to cascaded constructions. In *Advances in cryptology: CRYPTO 2004*, volume 3152 of *LNCS*, pages 430–440. Springer-Verlag, 2004.

[28] J. Kelsey and B. Schneier. Second preimages on n-bit hash functions for much less than 2^n work. In R. Cramer, editor, *EUROCRYPT*, volume 3494 of *LNCS*, pages 474–490. Springer, 2005.

[29] S. Lucks. Design principles for iterated hash functions. 2004. http://eprint.iacr.org/.

[30] A. Joux and T. Peyrin. Hash functions and the (amplified) boomerang attack. In *Advances in cryptology: CRYPTO 2007*, volume 4622 of *LNCS*, pages 244–263. Springer-Verlag, 2007.

[31] May 2007. Personal communication with S. S. Thomsen.

[32] T. A. Berson. Differential cryptanalysis mod 2^{32} with applications to md5. In *EUROCRYPT*, pages 71–80, 1992.

[33] H. Lipmaa and S. Moriai. Efficient algorithms for computing differential properties of addition. In *Proceedings of FSE 2001*, pages 336–350. Springer-Verlag, 2002.

[34] H. Lipmaa, J. Wallèn, and P. Dumas. On the Additive Differential Probability of Exclusive-Or. In Bimal Roy and Willi Meier, editors, *Fast Software Encryption 2004*, volume 3017 of *LNCS*, pages 317–331. Springer-Verlag, 2004.

[35] I. Nikolić D. Khovratovich and R.-P. Weinmann. Cryptanalysis of edon-r. http://ehash.iaik.tugraz.at/uploads/7/74/Edon.pdf.

[36] M. Bellare H. Krawczyk and R. Canetti. *RFC2104 - HMAC: Keyed-Hashing for Message Authentication*. Internet Engineering Task Force, 1997. http://www.faqs.org/rfcs/rfc2104.html.

[37] American Bankers Association. *Keyed Hash Message Authentication Code*. ANSI X9.71, Washington, D.C., 2000.

[38] National Institute of Standards and Technology. *The Keyed-Hash Message Authentication Code (HMAC), FIPS PUB 198-1*. Federal Information Processing Standards Publication, July, 2008. http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf.

[39] S. Halevi and H. Krawczyk. Strengthening digital signatures via randomized hashing. In *Proceedings of CRYPTO 2006*, volume 4117 of *LNCS*, pages 41–59, 2006.

[40] National Institute of Standards and Technology. *Randomized Hashing for Digital Signatures*. Draft NIST Special Publication 800-106, August, 2008. http://csrc.nist.gov/publications/drafts/800-106/2nd-Draft_SP800-106_July2008.pdf.

[41] B. Gladman. Sha1, sha2, hmac and key derivation in c. http://fp.gladman.plus.com/cryptography_technology/sha/index.htm.