Rune Steinsmo Ødegård

Hash Functions and Gröbner Bases Cryptanalysis

Thesis for the degree of Philosophiae Doctor

Trondheim, April 2012

Norwegian University of Science and Technology
Faculty of Information Technology,
Mathematics and Electrical Engineering
Department of Telematics

NTNU – Trondheim
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor

Faculty of Information Technology,
Mathematics and Electrical Engineering
Department of Telematics

© Rune Steinsmo Ødegård

ISBN 978-82-471-3501-3 (printed ver.)
ISBN 978-82-471-3502-0 (electronic ver.)
ISSN 1503-8181

Doctoral theses at NTNU, 2012:111

Printed by NTNU-trykk

Abstract

Hash functions are being used as building blocks in such diverse primitives as commitment schemes, message authentication codes and digital signatures. These primitives have important applications by themselves, and they are also used in the construction of more complex protocols such as electronic voting systems, online auctions, public-key distribution, mutual authentication handshakes and more. Part of the work presented in this thesis has contributed to the "SHA-3 contest" for developing the new standard for hash functions organized by the National Institute of Standards and Technology. We constructed the candidate EDON-$\mathcal{R}$, which is a hash function based on quasigroup string transformation. EDON-$\mathcal{R}$ was designed to be much more efficient than SHA-2 cryptographic hash functions, while at the same time offering same or better security. Most notably EDON-$\mathcal{R}$ was the most efficient hash function submitted to the contest.

Another contribution to the contest was our cryptanalysis of the second round SHA-3 candidate Hamsi. In our work we studied Hamsi's resistance to differential and higher-order differential cryptanalysis, with focus on the 256-bit version of Hamsi. Our main results are efficient distinguishers and near-collisions for its full (3-round) compression function, and distinguishers for its full (6-round) finalization function, indicating that Hamsi's building blocks do not behave ideally.

Another important part of this thesis is the application of Gröbner bases. In the last decade, Gröbner bases have shown to be a valuable tool for algebraic cryptanalysis. The idea is to set up a system of multivariate equations such that the solution of the system reveals some secret information of the cryptographic primitive. The system is then solved with Gröbner bases computation. Staying close to the topic of hash functions, we have applied this tool for cryptanalysis and construction of multivariate digital signature schemes, which is a major hash function application. The result of this is our cryptanalysis of the public-key cryptosystem MQQ, where we show exactly why the multivariate quadratic equation system is so easy to solve in practice. The knowledge we gained from finding the underlying weakness of the MQQ scheme was used to construct a digital signature scheme. The resulting scheme, MQQ-SIG, is a provably CMA resistant multivariate quadratic digital signature scheme based on multivariate quadratic quasigroups. The scheme is designed to be very fast both in hardware and in software. Compared to some other multivariate quadratic digital signature schemes, MQQ-SIG is much better in signing and private key size, while worse in key generation, verification and public key size. This means that MQQ-SIG is a good alternative for protocols where the constrained environment is on the side of the signer.

Preface

This thesis is submitted in partial fulfillment of the requirements for the degree of philosophiae doctor (PhD) at the Norwegian University of Science and Technology (NTNU). The work was performed at the Centre for Quantifiable Quality of Service in Communication Systems (Q2S), Centre of Excellence (CoE), during 2007-2011, and has been supervised by Professor Svein Johan Knapskog and Professor Bjarne Emil Helvik.

The document has been formatted in LaTeX using a modified version of the document class *kapproc.cls* provided by Kluwer Academic Publishers.

Acknowledgements

First, I would like to thank my supervisor Professor Svein Johan Knapskog for his support throughout my work on this thesis. A particular thanks also goes out to Professor Danilo Gligoroski who, in addition to have coauthored most of my papers, has in practice been my co-supervisor for this thesis.

I would also like to thank Research Director Jean-Charles Faugère, Assistant Professor Ludovic Perret and the rest of the SALSA-team for hosting me for 7 months. I learned a lot during my stay at the Pierre and Marie Curie University.

I also learnt a lot from collaborating with all my coauthors. In addition to the four people mentioned so far I would like to thank Jean-Philippe Aumasson, Aleš Drápal, Rune Erlend Jensen, Emilia Käsper, Vlastimil Klima, Lars Ramkilde Knudsen, Ljupco Kocarev, Smile Markovski, Marija Mihova and Thomas Peyrin for our joint papers.

I have made good friends and met a lot of very interesting people during my four years at Q2S. I would like to thank my colleagues for making Q2S a very pleasant environment to work in.

Finally, I would like to thank my friends and family. Their continued support has meant a lot to me.

Contents

List of Papers

Publications Included in the Thesis

The following are the papers included in Part II of the thesis. All papers have been subjected to minor editorial changes before their inclusion.

- PAPER A:
  Rune Steinsmo Ødegård, Marija Mihova and Danilo Gligoroski. *On Some Properties of Boolean Matrices from Latin Squares.* In Proceedings of the 1st International Workshop on Security and Communication Networks (IWSCN 2009). Trondheim, Norway. May 20-22, 2009

- PAPER B:
  Danilo Gligoroski, Rune Steinsmo Ødegård, Marija Mihova, Svein Johan Knapskog, Ljupco Kocarev, Aleš Dràpal, Vlastimil Klima. *Cryptographic Hash Function* EDON-$\mathcal{R}$. Submission to NIST 2008.

- PAPER C:
  Rune Steinsmo Ødegård and Danilo Gligoroski *On the Randomness and Regularity of Reduced* EDON-$\mathcal{R}$ *Compression Function.* In Proceedings of the 2009 International Conference on Security & Management (SAM 2009). Las Vegas, Nevada, USA. July 13-16, 2009.

- PAPER D:
  Jean-Philippe Aumasson, Emilia Käsper, Lars Ramkilde Knudsen, Krystian Matusiewicz, Rune Steinsmo Ødegård, Thomas Peyrin and Martin Schläffer. *Distinguishers for the Compression Function and Output Transformation of Hamsi-256.* In Proceedings of the 15th Australian Conference on Information Security and Privacy (ACISP 2010). Sydney, Australia. July 5-7, 2010.

- PAPER E:
  Jean-Charles Faugère, Rune Steinsmo Ødegård, Ludovic Perret and Danilo Gligoroski. *Analysis of the MQQ Public Key Cryptosystem.* In

Proceedings of the 9th International Conference on Cryptology and Network Security (CANS 2010). Kuala Lumpur, Malaysia. December 12-14, 2010.

- PAPER F:
  Danilo Gligoroski, Rune Steinsmo Ødegård, Rune Erlend Jensen, Ludovic Perret, Jean-Charles Faugère, Svein Johan Knapskog and Smile Markovski. *MQQ-SIG. An Ultra-fast and provably CMA Resistant Digital Signature Scheme.* In Proceedings of the 3rd International Conference on Trusted Systems, INTRUST 2011. Beijing, China. November 27-29, 2011.

Other Papers by the Author

The following papers were also prepared while working with this thesis.

- Danilo Gligoroski, Rune Steinsmo Ødegård, Marija Mihova, Svein Johan Knapskog, Aleš Drápal, Vlastimil Klima, Jørn Amundsen and Mohamed El-Hadedy *Cryptographic Hash Function* EDON-$\mathcal{R}'$. In Proceedings of the 1st International Workshop on Security and Communication Networks (IWSCN 2009). Trondheim, Norway. May 20-22, 2009
  –This paper presents a tweak on the hash function Edon-R of Paper B. This tweak was our planned improvment if we were selected for round two of the NIST SHA-3 competition.

- Danilo Gligoroski and Rune Steinsmo Ødegård *On the Complexity of Khovratovich et al.'s Preimage Attack on* EDON-$\mathcal{R}$. NIST mailing list. `http://eprint.iacr.org/2009/120`

- Rune Steinsmo Ødegård, Ludovic Perret, Jean-Charles Faugère and Danilo Gligoroski. *Analysis of the MQQ Public Key Cryptosystem.* In Proceedings of the Second International Conference on Symbolic Computation and Cryptography. Royal Holloway, United Kingdom. June 20-25 2010
  –An earlier version of Paper E.

- Danilo Gligoroski, Rune Steinsmo Ødegård and Rune Erlend Jensen. *OBSERVATION: An explicit form for a class of second preimages for any message M for the SHA-3 candidate Keccak.* NIST mailing list. `http://eprint.iacr.org/2011/261`

Abbreviations

ACISP	Australian Conference on Information Security and Privacy
AES	Advanced Encryption Standard
ANF	Algebraic Normal Form
BSI	Bundesamt für Sicherheit in der Informationstechnik
CANS	Conference on Cryptology and Network Security
CMA	Chosen Message Attack
CPU	Central Processing Unit
DES	Data Encryption Standard
DDT	Differential Distribution Table
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
HFE	Hidden Field Equations
HMAC	Hash function based Message Authentication Code
HW	Hamming Weight
IV	Initial Value
INSTRUST	International Conference on Trusted Systems
IWSCN	International Workshop on Security and Communication Networks
LSB	Least Significant Bits

MAC	Message Authentication Code
MCU	MicroController Unit
MD	Merkle-Damgård
MQ	Multivariate Quadratic
MQPKC	Multivariate Quadratic Public Key Cryptosystem
MQQ	Multivariate Quadratic Quasigroup
NESSIE	New European Schemes for Signatures, Integrity and Encryption
NIST	National Institute of Standards and Technology
NSA	National Security Agency
OV	Oil and Vinegar
RACE	Reseach and development in Advanced Communications technologies in Europe
RAM	Random Access Memory
RFID	Radio Frequency IDentification
RIPE	RACE Integrity Primitives Evaluation
SAT	Boolean SATisfiability
SAM	International Conference on Security & Management
Sbox	Substitution box
SHA	Secure Hash Algorithm
STS	Stepwise Triangular System
UOV	Unbalanced Oil and Vinegar
XL	eXtended Linearisation
XOR	eXclusive OR
XSL	eXtended Sparse Linearization

Part I

THESIS INTRODUCTION

1. Introduction

Ever since the Arab scholars invented cryptanalysis by breaking the monoalphabetic substitution cipher with frequency analysis around year 750 A.D. [Sin99], the science of cryptography has existed in duality with the science of cryptanalysis. In this thesis both sides of this duality are explored in two different ways. First, in the effort to contribute to the development of a new standard for cryptographic hash functions, we both construct our own candidate for this standard, and we cryptanalyse other candidates. Second, using the mathematical tool of Gröbner bases we cryptanalyse a multivariate public-key cryptosystem. We then use the knowledge we gain from this analysis together with other Gröbner bases computations to construct a multivariate quadratic digital signature scheme.

The result of this work is presented as a collection of six papers in Part II, which is the main part of this thesis. Here in Part I an introduction to the material covered in those papers is given. This part is organized as follows. In Section 2 I give the required background material. Then in Section 3 I present my research goals, before I in Section 4 discuss the research methodology we used to reach these goals. In Section 5 I present the main results of the papers included in Part II together with the my contribution to each paper. The papers are further discussed in Section 6, where the results are compared to contemporary work on the same subject. Finally, in Section 7 I conclude Part I of this thesis.

2. Background

In this section I will present the required background material for this thesis. I will give an introduction to hash functions, multivariate quadratic digital signature schemes, and the theory of Gröbner bases.

2.1 Hash Functions

2.1.1 General

A *cryptographic hash function*, $H : \{0,1\}^* \to \{0,1\}^n$, maps bit strings of arbitrary length[1] to bit strings of fixed length n. The output of the hash function is called *hash value*, or just *hash*. The input strings are called *messages*, and it should be easy to compute the hash of a message. Since the number of possible messages is much larger than the number of possible hashes, it is

[1]In most realizations of cryptographic hash functions the input length is actually bounded from above by a large constant. This constant is typically $2^r - 1$ with $r = 64$ or $r = 128$.

clear that there exist many messages that hash to the same output. Finding a pair of messages that hash to the same value is called finding a collision. Even though many such pairs exist, it is a requirement for a secure hash function that finding such a pair should be hard. Below we have formally defined this requirement together with two other characteristics that are required from a secure hash function. The definitions can for instance be found in "The Handbook of Applied Cryptography" [MVO96].

Definition 1 *A hash function H is said to be* preimage resistant *if it infeasible to find a message that maps to a given string. That is, given h it is infeasible to find M such that $H(M) = h$.*

Definition 2 *A hash function H is said to be* 2nd preimage resistant *if it is infeasible to find any second input which has the same output as any specific input. That is, given M it is infeasible to find M' such that $H(M) = H(M')$.*

Definition 3 *A hash function H is said to be* collision resistant *if it is infeasible to find two distinct messages that hash to the same output. That is, it is infeasible to find a pair (M, M') with $M \neq M'$ such that $H(M) = H(M')$.*

A *Random Oracle*, $R : \{0,1\}^* \to \{0,1\}^n$, is a deterministic entity that on input M outputs a bitstring h of length n, where h is chosen uniformly from the output domain and independently of M. This theoretical black box is very useful in cryptographic theory where it is used to model hash functions in proofs of the security of cryptographic protocols [BG81, BR93, FS86]. A particularly desirable property of a random oracle is that even if you know the hash of many different messages, you know nothing about the hash of a new message, even if the new message is related to the set of messages with known hash. This is a property which is hard to define properly. However, the random oracle model gives us an idealized way of describing this property, and how a good hash function should behave in general.

Since we want hash functions to come close to this ideal model, the security against preimage, 2nd preimage and collision attacks are required to be the same as a brute-force attack on a random oracle. This means a preimage and 2nd preimage attack should require $O(2^n)$ operations, while a collision attack should require $O(2^{\frac{n}{2}})$. The reason for this much lower complexity for collision attacks is due to the birthday attack [Yuv79]. The attack can best be explained by an analogy. In a group of 23 people most people intuitively assume that the chance that two people in that group share a birthday is quite low. The chance is however approximately 50%. The reason is that you have to consider each pair of people, and in a group of 23 there are 253 pairs. Applying this to a collision attack on a hash function, each call to the function increases the number of pairs for possible collision drastically. It can be proven that you on average only need $1.25\sqrt{2^n}$ hash function calls to find a collision.

2.1.2 NIST Hash Competition

In 2005, Wang et al. [WYY05a] presented cryptanalysis of SHA-1, the hash function that at that time was recommended by the National Institute of Standards and Technology (NIST). The theoretical collision attack required 2^{69} calculations, which is about 2000 times faster then brute-force[2]. Wang et al.'s result, together with even better collision attacks on other popular hash functions [WY05], forced NIST to take action. In addition to recommending transition to the SHA-2 family of hash functions, NIST held two public workshops to assess the status of its approved hash algorithms and to solicit public input on its cryptographic hash algorithm policy and standard. As a result of these workshops, NIST decided to develop one or more additional hash algorithms through a public competition, similar to the development process of the Advanced Encryption Standard (AES) [NIS]. On 2nd of November 2007 NIST issued a public call for a new cryptographic hash algorithm [NIS07], and launched the "SHA-3" competition. The deadline for submission of candidates was 31st of October 2008.

2.1.3 Designing Hash Functions

In this section some of the aspects of designing a secure cryptographic hash function will be discussed. Almost all hash functions are of an iterated structure built around a compression function.

Definition 4 *A* compression function, $f : \{0,1\}^r \times \{0,1\}^n \rightarrow \{0,1\}^n$, *is a function that transforms two fixed length inputs into the same size as one of the inputs.*

Note that compression functions are usually required to be *one-way*, meaning that given a particular output it should be hard to find an input that compresses to that output. The most well-known construction method for hash functions is the Merkle-Damgård construction.

Merkle-Damgård Construction. The Merkle-Damgård (MD) construction is named after Ralph Merkle and Ivan Damgård, who independently [Mer89, Dam89] proved the soundness of the structure. By sound we mean

[2]Since then the attacks have gotten better and better. The current best result is 2^{51} operations by Manuel [Man11].

that a collision in the hash function implies a collision in the compression function. This is restated in Theorem 5 below.

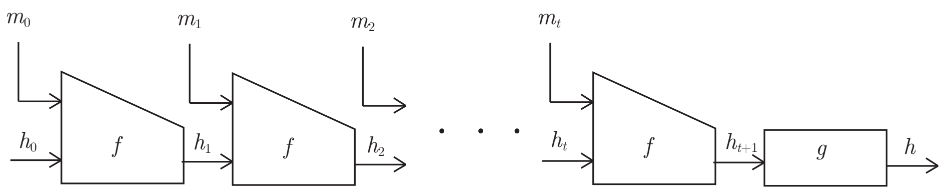

Figure 1. The Merkle-Damgård construction with initial value h_0, compression function f, optional finalization function g, padded message $M = m_0 \dots m_t$ and output h.

The MD-construction is depicted in Figure 1. A message M of arbitrary finite length b is divided into t blocks of length r, $M = m_0 m_1 \dots m_{t-1}$. To the last block a 1 followed by the appropriate number of 0-bits are appended to make it r bits long. Then an extra block m_t is appended to M, where m_t is the right-justified binary representation of b, the length of the message. This process is called *padding*, or *Merkle-Damgård strengthening*, and is a crucial part of the MD-construction. After padding, the iteration process starts from a fixed *Initial Value*, $IV = h_0$. The hash is computed iteratively using a compression function f.

$$h_i = f(h_{i-1}, m_{i-1}) \qquad i = 1, \dots t+1 \tag{1}$$

After the iteration an optional finalization function g may be applied. The final hash is then $H(M) = g(h_{t+1})$. It is important for the soundness of the construction that g is a permutation in the output bits.

THEOREM 5 *Let H be a hash function based on the Merkle-Damgård construction, where the message is padded as described above, let f be the compression function, and let g be the finalization function. Then a collision of H implies a collision of f.*

PROOF Let $M = m_0 \dots m_{t-1}$ and $M' = m'_0 \dots m'_{t-1}$ be two distinct messages that give a collision in the hash function, and let m_t and m'_t be the corresponding blocks that are appended in padding process. A collision in H means that the output of g is the same for both messages, but since g is a permutation in the output bits, we must have $h_{t+1} = f(h_t, m_t) = f(h'_t, m'_t) = h'_{t+1}$. If the length of M is different then the length of M' then $m_t \neq m'_t$ by the padding rule, which means $f(h_t, m_t) = f(h'_t, m'_t)$ is a collision of f. Assume the lengths of M and M' are equal. This means we either have a collision in f, or that $h_t = h'_t$ and $m_t = m'_t$. Assume the latter. Then we have $h_t = f(h_{t-1}, m_{t-1}) = f(h'_{t-1}, m'_{t-1}) = h'_t$, which means we either have a collision in f, or that $h_{t-1} = h'_{t-1}$ and $m_{t-1} = m'_{t-1}$. Continuing like this, we get by reverse induction that either we have a collision in f, or $m_i = m'_i$ for $0 \leq i \leq t$. Since M and M' are distinct messages we must have a collision in f. □

In the last decade generic attacks have been published which show some weaknesses of the MD-construction. All the following results are on hash functions with MD-strengthening. Joux [Jou04] showed how given a function which finds collisions in a hash function, it is easy to find sets of messages that all have the same hash value. Kelsey and Schneier [KS05] showed how one could find 2nd preimages of n-bit hash functions for much less than 2^n work using very long messages. Kelsey and Kohno [KK06] published the herding attack, in which an attacker who can find many collisions on the hash function by brute force can first provide the hash of a message, and later "herd" any given starting part of a message to that hash value by the choice of an appropriate suffix. There is also the well-known length-extension attack, where given $H(M)$ and the length of M, the attacker can calculate $H(M||M')$ for a suitable M' without knowing M.

To counter these inherit weaknesses of the MD-construction a lot of tweaks and improvements have been proposed. Lucks proposed increasing the internal state of the hash functions [Luc05]. Instead of using compression functions with the same size as the n-bit hash output, Lucks proposed using compression functions with output size w for iteration, and then another compression function $f' : \{0,1\}^w \rightarrow \{0,1\}^n$ as a finalization function. With $w > n$, this is called the *wide-pipe* construction, and *double-pipe* if $w = 2n$. Conversely, constructions with $w = n$ are called *narrow-pipe*. Nandi and Paul proposed fast wide-pipe construction as a more efficient version of Lucks' design [NP10a].

Coron et al. proposed the prefix-free Merkle-Damgård construction [CDMP05]. They showed that if the underlying compression function is viewed as a random oracle or an ideal block-cipher, then any crypto-system proven secure in the random oracle model, remains secure if one instead plugs in the Prefix-free MD construction. This is a nice property, since most crypto-systems that use hash functions are proven secure in the random oracle model.

Another notable MD-based iterated constructions is HAIFA [BD06] proposed by Biham and Dunkelman. Their proposed framework fixes many of the found flaws of the MD-construction while supporting several additional properties such as randomized hashing and variable hash size. The SHA-3 finalist Blake [AHMP08] uses this construction.

Cryptographic Sponges. An interesting and relatively new method of designing hash functions is the *sponge construction* developed by Bertoni et al. [BDPA07]. The sponge construction is an iterated construction for building a function $F : \{0,1\}^* \to \{0,1\}^*$ with variable input length and arbitrary output length based on a fixed-length transformation (or permutation) $f : \{0,1\}^b \to \{0,1\}^b$ operating on b bits. Here b is referred to as the *width* of the sponge function. Fixing the output size of the sponge construction F to n bits gives us an n-bit hash function.

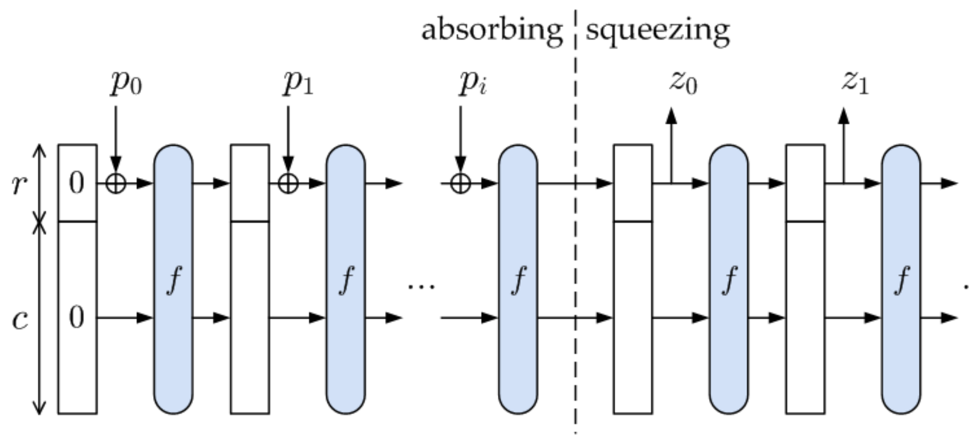

Figure 2. The sponge construction [BDPA07].

The construction is depicted in Figure 2. The sponge function operates on a state of $b = r + c$ bits, where c is called the *capacity*, and r is called the *bitrate*. The input message is divided into blocks of r bits, where the last message block is appropriately padded to reach the desired length. All the bits of the state are initialized to zero, before the two stages of the sponge function begins. First, in the *absorbing phase*, the r-bit message blocks are XORed into the first r-bits of the state, interleaved with applications of the function f. After all message blocks are absorbed the sponge construction switches to the *squeezing phase*. In this phase the first r-bits are returned as output blocks, interleaved with applications of the function f. For n-bit hash functions the number of output blocks is $\frac{n}{r}$.

Unlike the MD-construction, the security of the hash functions using the sponge construction does not depend on the number of output bits, but instead of the capacity of the permutation f. For a random transformation (or permutation) f, Bertoni et al. [BDPA07] have proved that the sponge only differs from a random oracle by existence of inner collisions. They use this together with some other observations on the sponge construction to define the flat sponge claim.

Definition 6 *Given a capacity c_{claim}, the success probability of any attack should be not higher than the sum of that for a random oracle and*

$$1 - e^{-N^2 2^{-(c_{claim}+1)}}, \qquad (2)$$

with the workload of the attack having the computational equivalent of N calls to f (or its inverse).

For collision resistance the above security claim translates into $2^{\frac{n}{2}}$ if the output length is $n \leq c_{\text{claim}}$, and $2^{\frac{c_{\text{claim}}}{2}}$ if the output length is $n \geq c_{\text{claim}}$. For preimage and 2nd preimage attacks the resistance is 2^n if $n \leq c_{\text{claim}}$, and $2^{c_{\text{claim}}}$ if $n \geq c_{\text{claim}}$.

Dedicated Hash Functions. Dedicated hash functions are algorithms that are designed specifically for hashing. Looking at the hash functions that have been and are used in practice, this is the most commonly employed type of hash functions. The reason for this is that these functions are usually much more efficient then hash functions based on block ciphers, or those based on some hard problem. Most dedicated hash functions are based on variants of the Merkle-Damgård construction. Examples include two of the first published dedicated hash functions, namely was Merkle's Snefru [Mer90b] and Rivest's MD2 [Kal92]. MD2 was later superseded by two improved algorithms. First MD4 [Riv90] and later MD5 [Riv92]. MD5 was for many years one of the most used hash functions, and is still widely employed today, even though it is broken [dBB93, WY05, SLdW07]. Another interesting proposal around this time is Zhen et al.'s HAVAL [ZPS93], which has variable output length and a parameter that controls the level of security by changing the number of passes per message block.

In 1992, under the European RACE Integrity Primitives Evaluation (RIPE) project, den Boer and others developed RIPEMD [BP95]. The construction was a strengthened version of MD4 designed to counter known attacks. Later Dobbertin et al. developed RIPEMD-160 [DBP96] because of concern about both possible attacks and the inherent limitation of 128-bit output digest.

In 1993, NIST proposed SHA [NIS93b] (now called SHA-0) as the U.S government standard for cryptographic hash functions. The hash function was later withdrawn by the NSA because of unspecified weaknesses, and superseded by SHA-1 [NIS95] in 1995. The only difference between SHA-0 and SHA-1 is the inclusion of a 1-bit rotation in the block expansion part of the algorithm. The design of SHA-1 is based on the Merkle-Damgård construction and was partially inspired by MD5. Similarly to MD5, the SHA-1 hash function is still widely employed today, even though it is broken [WYY05a, CMR07]. Unlike MD5 all breaks published so far are only theoretical. Most experts believe, however, that a collision for SHA-1 will be found in the near future. The life cycles of some popular employed hash functions is depicted in Figure 3.

Life cycles of popular cryptographic hashes																						
Function	90	91	92	93	94	95	96	97	98	99	00	01	02	03	04	05	06	07	08	09	10	11
Snefru	🟩	🟩	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
MD2	🟩	🟧	🟧	🟧	🟧	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
MD4			🟩	🟩	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
MD5			🟩	🟩	🟩	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
RIPEMD			🟩	🟩	🟩	🟩	🟩	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
HAVAL-128			🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟧	🟧	🟧	🟧	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
SHA-0				🟩	🟩	🟩	🟩	🟩	🟧	🟧	🟧	🟧	🟧	🟧	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
SHA-1						🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧
RIPEMD-128							🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
RIPEMD-160							🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
SHA-2 family											🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩

Figure 3. Life cycle of popular cryptographic hash functions [Aur09]. Green square indicates the hash function is unbroken, orange indicates the function is weakened, while red indicates the function is broken.

Among the SHA-3 finalists, Blake [AHMP08] and Grøstl [GKM$^+$08] are examples of dedicated hash functions which use variants of the MD construction. While Keccak [BDPA08] and JH [Wu09] are examples of dedicated hash functions which use the sponge construction.

Block Cipher Based Hash Functions. The construction of hash functions based on block ciphers is a well-known approach dating back to when DES [NIS93a] was used as the underlying block cipher in the Davies-Meier construction. This construction was later proven secure in the ideal cipher model [Win84]. The general construction of block cipher based hash constructions was studied by Preneel, Govaerts and Vandewalle [PGV93], from here on referred to as PGV. They considered turning a block cipher $E : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ into a hash function $H : \{0,1\}^* \to \{0,1\}^n$ using a compression function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ derived from E. For a fixed n bit constant v, PGV considers all 64 compressions functions of the form $f(h_{i-1}, m_{i-1}) = E_a(b) \oplus c$, where $a, b, c \in \{h_{i-1}, m_{i-1}, h_{i-1} \oplus m_{i-1}, v\}$. This description of PGVs work is by Black et al. [BRS02], and is depicted in

Figure 4. The iterated hash of m is then defined as

$$h_i = f(h_{i-1}, m_{i-1}) \qquad i = 1, \ldots t \tag{3}$$

Here $m_0, \ldots, m_{t-1}$ are the message blocks of size n where the last block is appropriately padded. The constant h_0 is some specified initial value, and h_t is the final output of the hash function. PGV found that 12 of the 64 possible compression function constructions are secure using an attack based approach. Later Black et.al [BRS02] proved the same 12 schemes to be secure in the ideal cipher model. In addition, when stepping outside the MD approach to analysis, they found 8 other PGV schemes that are just as just as collision resistant as those 12 schemes.

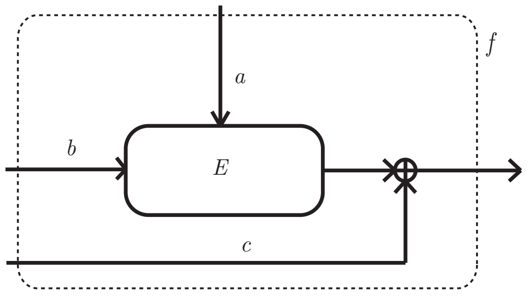

Figure 4. A compression function f based on a block cipher E. The plaintext b, key a and feed forward c can all be from the set $\{h_{i-1}, m_{i-1}, h_{i-1} \oplus m_{i-1}, v\}$.

Even though the security of block cipher based hash functions is well studied, the approach has been less widely used. The reasons for this include export restrictions on block ciphers, a preponderance of 64-bit block lengths, problems attributable to "weak keys", and the lack of popular block ciphers with per-byte speeds comparable to that of dedicated hash functions [BRS02].

An example of a block cipher based hash function is the SHA-3 finalist Skein [FLS+08], which is built on the block cipher Treefish using Unique Block Iteration, a variant of the Matyas-Meyer-Oseas hash mode [MMO85]. This is one of the twelve modes found secure by PGV.

Provably Secure Cryptographic Hash Functions. This class of cryptographic hash functions is based on some hard mathematical problem. A hash function is said to be *provably secure* if breaking it is as difficult as breaking the hard problem on which they are based. There are many different examples of such functions. Breaking FSB [AFG+08] is at least as difficult as solving the

NP-complete problem of regular syndrome decoding. VHS [CLS06] is based on the problem of finding nontrivial modular square roots modulo a composite number. Damgård considered hash functions based on the knapsack problem [Dam89].

The downside of this construction is that most provably secure hash functions are too inefficient to be used in practice. FSB was for instance one of the slowest submissions to the SHA-3 competition [Be11b].

2.1.4 Hash Function Cryptanalysis

A hash function is said to be broken if there exists an attack with shorter expected running time than the security claim of the hash function. For instance, a hash function with claimed collision resistance level of $2^{\frac{n}{2}}$ is said to be broken if there exists a collision attack with expected running time $2^{\frac{n}{2}-t}$ with $t \geq 1$. Attacks are called *practical* if the computation necessary to break the hash function is feasible for a high resource attacker. If not, the attack is called *theoretical*. Today, an attack needs to be at least under 2^{60} to be considered practical. Most new hash functions have a claimed security against collision attacks of 2^{128} or higher. Since it is hard to find an attack on a well designed function that is 2^{70} times faster then the security claim, almost all cryptanalytical results starts out as theoretical, pseudo attacks, attack on reduced functions, or some combination of these three.

Pseudo Attack. A *pseudo attack* is an attack that uses more degrees of freedom then what is given by the hash function. A common example of this are attacks where the attacker is free to choose the initial value of the hash function. This type of attacks is also referred to as *free start attacks.* Even though pseudo attacks (usually) do not violate the security claims of the hash function, they are still considered interesting cryptanalytical results. The reason for this is that they give a better understanding of the attacked hash function, and its security margin.

Cryptanalysis of Reduced Functions. Cryptanalysis of hash functions often starts by looking at some reduced version of the hash function, and if there are any results, see if these can be extended to the full version. Most hash functions come with some security parameter which is natural to reduce. This can for instance be the number of rounds the compression function does per message block iteration. Other types of reductions can be reducing the size of the state in a natural way, or removing a small part of the compression function.

Similar to pseudo attacks, cryptanalytical results on reduced hash functions are considered interesting, even if they cannot be extended to the full version, since they give a better understanding of the attacked hash function, and its security margin.

Differential Cryptanalysis. Differential cryptanalysis is the main technique used to attack hash functions. The concept was first introduced by Biham and Shamir [BS91], and used to attack the block cipher DES and DES like crypto-systems.

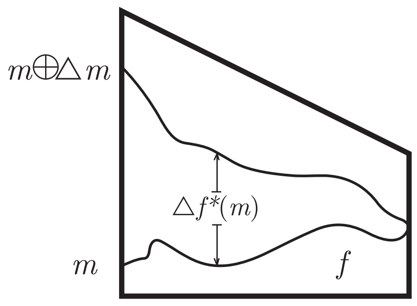

Figure 5. Example of a differential path on a compression function f leading to an output difference of zero. Here f^* represents some intermediate evaluation of f.

The main idea when applied to hash functions is to look at input differences, $\Delta m = m \oplus m'$, trying to find one that leads to an output difference of zero of the compression function, $\Delta f(m) = f(m) \oplus f(m') = 0$, in other words a collision. A symbolism of this is depicted in Figure 5. The process of differential cryptanalysis is to analyze the internal structure of the compression function, trying to trace a path of highly probable differences through various stages of compression. A sequence of input and output difference so that the output difference from one stage of compression corresponds to the input difference for the next stage is called *differential characteristic*, or a *differential path*. A good differential path for hash functions is one that optimizes the probability of the path while minimizing the effort to find m. A differential path with probability $p > 2^{-\frac{n}{2}}$ on an n-bit hash function can be used to find a collision by trying (on average) $\frac{1}{p}$ random message pairs $(m, m \oplus \Delta m)$ checking if $\Delta f(m) = 0$. This collision attack has complexity less then $2^{\frac{n}{2}}$.

A well-known differential cryptanalytical results of hash functions is Wang et al.'s on SHA-1 [WYY05a] and other well known hash functions [WY05].

Truncated Differential Cryptanalysis. Truncated differential cryptanalysis can be seen as a generalization of differential cryptanalysis. Instead of predicting the output difference between two input states of the compression function, a truncated differential path predicts only part of the difference in the output.

By not considering all bits in the state it is possible to find (truncated) differential paths with higher probability. Such paths can be used for key recovery attacks (on keyed hash functions), statistical distinguishers, or they might even be extended to collision attacks.

Truncated differential cryptanalysis was introduced by Knudsen [Knu95], and has for instance been used by Peyrin [Pey07] to attack the hash function Grindahl [KRT07].

Higher-Order Differential Cryptanalysis. Higher-order differential cryptanalysis was introduced by Lai [Lai92], and first applied to hash functions by Knudsen [Knu94]. The difference between differential and higher-order differential cryptanalysis is that instead of looking at the propagation of input differences, higher-order differential cryptanalysis exploits the propagation of the difference between differences. This technique has for instance been applied to reduced SHA-256 [LM11] and Keccak and Luffa [BCC10]. The recently proposed cube attacks [DS09] and cube testers [ADMS09] are also forms of higher-order differential cryptanalysis techniques.

k-Sum and Zero-Sum Distinguishers. Another form of higher order differential cryptanalysis are k-sums and zero-sums. The k-sum problem, applied to a hash function H, is to find k distinct messages such that their image sum to zero.

$$\bigoplus_{i=1}^{k} H(M_i) = 0 \tag{4}$$

With $k = 2$ this is essentially the collision problem. If we also require that the messages sum to zero, $\bigoplus_{i=1}^{k} M_i = 0$, we have the zero-sum problem. Finding k-sums and zero-sum is hard for an ideal random function, so the finding of such sets can be seen as a distinguisher for the hash function.

The notion was first introduced by Aumasson and Meier in [AM09], where they applied it to Keccak, Luffa and Hamsi.

Meet-in-the-Middle Attack. A meet-in-the-middle attack is a time-memory tradeoff originally proposed by Diffie and Helman [DH77] to attack double DES, a proposed security improvement of the Data Encryption Standard [NIS93a]. We will describe the general idea with a preimage attack on BadHash-n, a narrow pipe Merkle-Damgård iterated hash function with invertible compression function f and n-bit output. To find a preimage for h the attacker computes $(h_1)_i = f(IV, m_i)$ in forward direction, and $(h_2)_j = f^{-1}(m_j, h)$ in backward direction, for $2^{\frac{n}{2}}$ random message blocks m_i and m_j. Then, by the birthday paradox in two groups [NS90], there will with high probability be a match $(h_1)_i = (h_2)_j$ for some i, j. The corresponding $m_i m_j$ is a preimage for h. This attack requires $O(2^{\frac{n}{2}+1})$ time and $O(2^{\frac{n}{2}+1})$ memory, as opposed to $O(2^n)$ time and $O(1)$ memory of a generic attack.

Even though the attack on this toy example cannot be extended to most hash functions, the general idea is the same; i.e. to decompose the hash or compression function into two (or more) parts, and compute with random inputs in forward and backward direction trying to find a match in the middle. The best choices for how and where to make the decomposition vary from hash function to hash function. Successful meet-in-the-middle attacks have for instance been applied to reduced round SHA-0 and SHA-1 [AS09] and MD4, Tiger and reduced SHA-2 [GLRW10].

Rebound Attack. The rebound attack is a state-of-the-art technique for cryptanalysis of hash functions. The technique can be described as a combination between differential cryptanalysis and the meet-in-the-middle attack, and was introduced by Mendel et al. [MRST09]. The technique can be applied to both block cipher based and permutation based compression functions. Let E be a block cipher, for permutation based functions just replace E with P. The idea is to decompose the cipher into three smaller parts $E = E_{bw} \circ E_{in} \circ E_{fw}$ as shown in Figure 6. The attack is then a two step process with a possible third step.

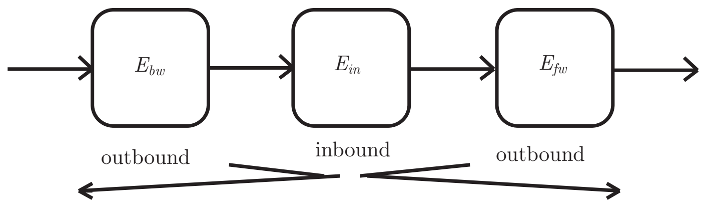

Figure 6. Rebound attack on a cipher $E = E_{bw} \circ E_{in} \circ E_{fw}$ [MRST09].

1 Inbound phase: This phase is a meet-in-the-middle phase in E_{in}, which is aided by the degrees of freedom that are available to a hash function cryptanalyst.

2 Outbound phase: In this phase, truncated differentials are used in both forward- and backward direction through E_{fw} and E_{bw} to obtain desired collisions or near-collisions

3 If the truncated differentials have a low probability in E_{fw} and E_{bw}, the inbound phase can be repeated to obtained more starting points for the outbound phase.

The rebound attack was first applied to reduced Whirpool and reduced round version of SHA-3 finalist Grøstl [MRST09]. Since then it has for instance been used to attack reduced round version of SHA-3 finalists Skein [KNR10] and JH [RTV10, NP10b], and full compression function of Lane [MNPN$^+$09].

Algebraic Cryptanalysis. The basic principle of algebraic cryptanalysis is to model a cryptographic primitive by a set of algebraic equations. The system of equations is constructed in such a way that there is a correspondence between the solution of the system and some secret information of the cryptographic primitive [Fau09]. For block ciphers the solution can for instance reveal the secret key, for hash functions the solutions can give a collision, or a preimage. After the equation system is constructed, some method for finding a solution is applied. The current state-of-the-art is the Gröbner bases approach (see Section 2.3), eXtended Linearisation (XL) approach[3] [CKPS00], and Boolean SATisfiability (SAT) solvers [ZM02].

There are not many published results on algebraic cryptanalysis of hash functions. Sugita et al. [SKPI07] have applied a combination of differential and algebraic cryptanalysis to SHA-1, while Morawiecki and Srebrny [MS10] applied a SAT based preimage attack on Keccak. One of the reasons there are so few results is that when applying algebraic techniques to hash functions the corresponding algebraic systems are so huge (thousands of variables and equations) that nobody is able to predict correctly the complexity of solving such polynomial systems [Fau09].

Statistical Cryptanalysis. Statistical cryptanalysis is the process of looking at the output distribution of the hash function and trying to distinguish it from the distribution of an ideal random function. Although being a pseudo random function is not always in the security claim of the hash function, it is often considered an important property. The reason for this is that hash functions are often employed as a heuristic substitute for random oracles in

[3]XL has been shown to be equivalent to Gröbner bases by Ars et al. [AFI$^+$04]. However, the XL approach is still the preferred choice in some research communities.

cryptographic protocols proven secure in the random oracle model, and the output of random oracles are by definition uniform in the image and independent of the input.

The main process of statistical cryptanalysis is applying a wide range of statistical tools to the hash function (see for instance [RSN$^+$01, Fil02, BJV04]) trying to find some statistical anomaly. This anomaly might then be used to construct a distinguisher for the hash function. Given two possible distributions, a *distinguisher* is an algorithm which takes a sequence of realization of a distribution and determines which of the two possible distributions the sequence is from. When applied to hash functions the two possible distributions are the output hash under evaluation and the output of a random oracle.

2.1.5 Application of Hash Functions

Hash functions are often referred to as the swiss army knife of cryptographic tools. The reason for this is, of course, that hash functions are a versatile primitive which are used in many very different cryptographic systems. I will here mention some of the more common uses of cryptographic hash functions.

Commitment Scheme. A *Commitment scheme* is a protocol between to players A and B where player A chooses a value v from some (usually) finite set V and commits to this choice. Later, player A may choose to reveals his choice to player B.

Commitments are used to bind participants of cryptographic protocols to a value, such that they later cannot change that value in order to gain some unfair advantage. This has applications in for example interactive proof systems, verifiable secret sharing and auctions.

Given a cryptographic hash function H, player A can commit to a value $v \in V$ by generating a random string r and then compute $c = H(v||r)$. He then sends the commitment c to player B. To reveal the value v player A send (r, v) to player B, and B checks if $c \overset{?}{=} H(v||r)$. Note that after sending c to player B, player A can no longer change the value v without the change being detectable by B, because this implies finding a collision of the hash function $H(v||r) = c = H(v'||r')$. This is called the *binding* property of the commitment scheme. On the other hand, player B cannot guess the value v before player A reveals (v, r) since there are infinite combinations of v and r that hash to c.[4] This is called the *hiding* property of the commitment scheme.

[4]Note that this is only true for most hash functions. It is possible to construct a hash function where $H(v||r)$ is not hiding. A more complex hash function based commitment scheme that is hiding for all hash functions can be found in [DPP98].

The example above is based on Damgård et al. [DPP98], who proved that if collision-intractable hash functions exist, then there exist commitment schemes with unconditional hiding and computational binding. This was an extension of the bit-commitment result of Naor and Yung [NY89].

Password Protection. System providers should not store their users passwords in plaintext. The main reason for this is that if someone hacks the systems database they should not gain access to the login and passwords of all users. An easy solution to this is to instead store the hash of the password together with a salt and the user identification, $H(\text{salt}, U_{ID}, \text{password})$. When the user logs in, this hash is computed and compared to the hash stored in the database. The idea of storing the encryption of the password was first described by Wilkes [Wil75].

Key Derivation. In situations where two parties share a symmetric key it is often wise for them to generate session keys, and instead use these session keys when communicating. One of the reasons for this is that if a session key is compromised the adversary will only be able to read the communication for that particular session. Another reason is that it is a good cryptographic principle to use different keys for different cryptographic purposes. Using key derivation one can easily obtain separate keys for encryption and integrity protection from one master key. This principle is for instance used in 3G Partnership Project [NN03]. The user and the phone company share a key stored in the SIM card of the user, and this key is used to derive new cipher keys, integrity keys and anonymity keys each time the user wants to make a call. See for instance [Che09] for how hash functions may be used for key derivation.

Key Stretching. In many real-world applications there may be a need to increase the entropy of the key-space to avoid the possibility of brute force attacks. The reason for this can be poorly chosen passwords, or that the system is outdated and only accepts keys of 40-bits. Kelsey et al. [KSHW98] explain how hash functions can be used to stretch s bit keys such that the complexity required to brute-force search a $s+t$-bit key-space is the same as the time required to brute-force search a s-bit key stretched by t bits.

Message Authentication. In cryptography it is often important to check the authenticity and integrity of information transmitted over, or stored on, an unreliable medium. That is, when Alice receives a message from Bob, she wants to be sure both that the message is indeed from Bob, and that the message received is exactly as sent. For this purpose Message Authentication

Codes (MAC) are used. One way to construct a MAC from a hash function H was proposed by Bellare et al. [BCK96]:

$$\mathrm{HMAC}(K, M) = H(K \oplus OP || H(K \oplus IP || M)) \tag{5}$$

Where K is the key, $\oplus$ is bitwise XOR, $OP = $ 0x5C repeated B times, $IP =$ 0x36 repeated B times, M is the message and B is the block length of the hash function. Note that hash function based MACs are also called *keyed hash functions.*

Digital Signatures. Loosely speaking, digital signatures can be thought of as the electronic equivalent of hand written signatures. They can be used to give evidence of both the provenance of the document, and the intention of an individual with regards to that document. However, digital signatures offer much stronger security guarantees. For instance, forging a handwritten signature is relatively easy with a bit of practice, while forging a digital signature requires solving a hard mathematical problem. In addition, digital signatures offer desirable properties such as non-repudiation and public verifiability.

Today, digital signatures are accepted as legally binding in many countries, where they can be used for certifying contracts or notarizing documents, and for authentication of individuals or cooperations [Kat10]. Across the internet digital signatures are for instance used to issue software patches and updates in an authenticated manner. Another important online application is the secure distribution and transmission of public-keys. Because of this, digital signatures can be thought of as the foundation of all public-key cryptography [Kat10].

The concept of digital signatures was introduced by Diffie and Hellman [DH76a, DH76b] in 1976. However, in their papers they only conjectured that such a scheme existed. The first concrete realization came two years later with the design of the RSA public key algorithm by Rivest, Shamir and Adleman [RSA78]. This algorithm, together with the ElGamal signature scheme [EG84], are the two most widely used signature schemes today. A more thorough history of digital signatures can be found in [MVO96], while Katz [Kat10] has written a good introductory book on the concept.

Following the informal description by Katz [Kat10], a digital signature scheme is typically used by a *signer* and a set of *verifiers.* The scheme starts with the signer using a *key-generating algorithm* to produce a set of keys (pk, sk) The key pk is called the *public key* and is published by the signer through some publicly authenticated channel. We assume that any potential verifiers can get hold of the signers public key. The key sk is called the *secret key* or *private key.* To sign a message M the signer uses a signature algorithm together with his private key to produce a signature $\sigma = \mathrm{Sign}_{sk}(M)$. This is done in such a way that any other party who knows pk can verify that the message originated from the signer and has not been modified in any way. To do this the verifier inputs the public key, message and signature into a verification algorithm $\mathrm{Vrfy}_{pk}(M, \sigma) = b$. The verifier accepts the signature if $b = 1$, and rejects if $b = 0$.

Most digital signature schemes can only sign messages of some fixed short length. A natural way of handling longer messages is to hash all messages before signing them. Given a keyed hash function H_s this can be done in the following way [Kat10]: Generate a fresh k bit key s. The signature σ is then computed as

$$\sigma = (s, \mathrm{Sign}_{sk}(s||H_s(M))). \tag{6}$$

The downside of such a construction is that an adversary who can find collisions in the hash function easily can forge signatures. It is therefore important to use a cryptographically secure hash function.

There exist some digital signatures schemes that are considered both efficient and provably secure. These include schemes based on the RSA assumption [DN94, CD96, HW09], strong RSA assumption [CS00, Fis03, GHR99], and on bilinear maps [BB04, Wat05]. However, these schemes are not widely used. The reason for this is that they are less efficient than heuristically secure digital signature schemes. The construction of such schemes uses the random oracle model (see Section 2.1.1). This means that they are proven secure assuming the existence of a random oracle, while in practice the oracle is replaced with a hash function. Example of such schemes includes variants of the Full Domain Hash Signature Scheme ([BR93, BLSS04, RDS98, KW03]), and variants of identification scheme based signature schemes ([FS86, KR00, GMR89, GQ88, OS90, NIS09]),

2.2 Multivariate Quadratic Digital Signature Schemes

Ever since the first published realization of a public-key cryptosystem by Rivest, Shamir and Adelman [RSA78], a lot of research has been put into public-key primitives based on different hard mathematical problems. The reason for this is that cryptography is a forever evolving cycle of cryptanalysis followed by new or improved constructions. Even though RSA is currently not broken we want to make sure we have an alternative if somebody breaks it. This concept became even more important when Shor [Sho97] showed that there exist polynomial time quantum algorithms for factoring numbers and solving the discrete logarithm problem. These are the two hard problems on which most public-key cryptosystems are based on today. As a consequence, the search for post-quantum public-key cryptosystems is currently an important line of research.

At a first glance, public-key cryptosystems based on multivariate quadratic equations (MQPKC) are very appealing. One reason for this is that the encryption and decryption procedures are very efficient, and can also be performed in constrained environments [BERW08, CCC$^+$09]. Another reason is that solving systems of multivariate equations is a know NP-complete problem [GJ79], and it is a problem which is likely to be hard even assuming the existence of quantum computers [BBBV97]. Solving a system of random multivariate quadratic (MQ) equations with the tools we have today has exponential complexity, both in theory and practice. However, as soon as we introduce some structure into the equation system the complexity is much harder to predict. Algebraic tools, such as Gröbner bases, seem to be very good at exploiting such underlying structures, and thereby solving the equation system in much less than expected time. Because of this, the history of MQPKC is full of broken proposals, and new cryptosystems constructed to counter the attacks on these broken proposals. We will now give a short overview of this history based on the following three surveys [WP05, **?**, BD09]. Note that we will mention both public-key encryption systems and signature schemes, since they are often closely related.

Let $\mathbb{K}$ be a field. Generally, a multivariate quadratic public-key cryptosystem is a system of multivariate quadratic equations over $\mathbb{K}[x_1, \ldots, x_n]$,

$$\begin{aligned} y_1 = p_1(x_1, \ldots, x_n) &= \sum_{1 \leq j \leq k \leq n} a_{1,j,k} x_j x_k + \sum_{j=1}^{n} b_{1,j} x_i + c_1 \\ &\vdots \\ y_i = p_i(x_1, \ldots, x_n) &= \sum_{1 \leq j \leq k \leq n} a_{i,j,k} x_j x_k + \sum_{j=1}^{n} b_{i,j} x_i + c_i \qquad (7) \\ &\vdots \\ y_m = p_m(x_1, \ldots, x_n) &= \sum_{1 \leq j \leq k \leq n} a_{m,j,k} x_j x_k + \sum_{j=1}^{n} b_{m,j} x_i + c_m \end{aligned}$$

where $(y_1, \ldots, y_m)$ is the message digest bits to be signed (or the encryption of a message), and $(x_1, \ldots, x_m)$ is the signature (or decryption of a ciphertext). Hidden in the equation system is some kind of trapdoor such that solving the system is very easy if you know the private key. However, if you do not have access to the private key, solving the system should be as hard as solving a random system of equations with the same size.

The construction of the trapdoor consist of finding an easily invertible mapping over an extension field L that can be described as a system of polynomials over the ground field K: $\mathcal{P}' : \mathbb{K}^n \to \mathbb{K}^m$. The structure of this system is hidden from the attacker by two linear invertible affine transformations, $S : \mathbb{K}^n \to \mathbb{K}^n$ and $T : \mathbb{K}^m \to \mathbb{K}^m$, as follows

$$(x_1, \ldots, x_n) = x \underbrace{\overset{S}{\longmapsto} x' \overset{\mathcal{P}'}{\longmapsto} y' \overset{T}{\longmapsto}}_{\mathcal{P}} y = (y_1, \ldots, y_m). \tag{8}$$

Since we only consider multivariate quadratic equations this trapdoor is the only one possible [WP05]. The equation system $\mathcal{P}$ is the public key, while the triple $(\mathcal{P}', S, T)$ is the private key. To sign a message $(y_1, \ldots, y_n)$ the signer uses his knowledge of the private key to solve the equation system $\mathcal{P}(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$ for $(x_1, \ldots, x_n)$. Any person who know the public key of the signer can then verify if $\mathcal{P}(x_1, \ldots, x_n) \overset{?}{=} (y_1, \ldots, y_n)$.

The first MQPKC called MIA (and C^*) was proposed by Matsumoto and Imai [MI88]. The scheme was broken by Patarin [Pat95] who then proposed the Hidden Field Equations (HFE) scheme [Pat96] and the Oil and Vinegar (OV) scheme [Pat97]. Both these schemes were broken by Kipnis and Shamir [KS98, KS99]. Kipnis et al. then proposed Unbalanced Oil and Vinegar (UOV) as an improvement of OV [KHPG99]. Cryptanalysis by Braeken et al. [BWP05] and Faugère and Perret [FP09] show that not all parameters of UOV are secure. Another family of trapdoors is the Stepwise Triangular Scheme (STS) introduced in [WBP04]. STS can be considered a generalization of the Tame Transformation Method [Moh99] and the Triangle Plus Minus [GC00]; and is also related to early work by Shamir [Sha93]. This whole family is considered broken [WBP04]. Gligoroski et al. proposed using Multivariate Quadratic Quasigroups (MQQ) as the trapdoor [GMK08b]. This scheme was broken by Mohamed et al. [MDBW09].

Many modifiers have been proposed to improve the security of the different MQPKCs. The minus modifier, which consists of removing one or more equations from the public key, was originally proposed by Shamir [Sha93]. Patarin suggested to use the method to strengthen HFE in the context of signatures. A variant of this is called Quartz [PCG01b]. The method has also been applied to MIA [PGC98]. A variation of this scheme called SFLASH [PCG01a] which was selected in 2003 by the NESSIE European Consortium as one of the three recommended public key signature schemes. SFLASH has been successfully cryptanalyzed taking advantage of the underlying monomial structure [DFSS07]. However, the minus modifier is still considered secure against algebraic cryptanalysis.

An overview of other modifiers is presented in Table 1 from [Wol06]. A detailed description of the modifiers can be found in [WP05] .

Table 1. An overview over generic modifications of MQ schemes.

Symbol	Long name	Security
-	Minus	secure
+	Plus	mostly no effect
/	Subfield	insecure
$\perp$	Branching	insecure
f	fixing	open
h	homogenizing	no effect
i	internal	open
m	masking	open
s	sparse	open
v	vinegar	slightly more secure

2.3 Gröbner bases

2.3.1 General Theory

In this section I will give an introduction to Gröbner bases aligned with the presentation in "Using Algebraic Geometry" by Cox, Little and O'Shea [CLO05]. Let $\mathbb{K}$ be a field and $\mathbb{K}[x_1, \ldots, x_n]$ be the polynomial ring over $\mathbb{K}$ in the variables $x_1, \ldots, x_n$. A *monomial* in a collection of variables is a product $x^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ where $\alpha_i \geq 0$. The *total degree* of a monomial x^α is the sum of its exponents $|\alpha| = \alpha_1 + \cdots + \alpha_n$. A *polynomial*, f, is a finite linear combination of monomials with coefficients in $\mathbb{K}$. That is,

$$f = \sum_\alpha c_\alpha x^\alpha, \tag{9}$$

where $c_\alpha \in \mathbb{K}$ for each α and there are finitely many terms $c_\alpha x^\alpha$ with $c_\alpha \neq 0$. We are now ready to define an ideal.

Definition 7 *([CLO05]) Let $I \subset \mathbb{K}[x_1, \ldots, x_n]$ be a non-empty subset. I is said to be an* ideal *if*

1 $f + g \in I$ whenever $f \in I$ and $g \in I$, and

2 $pf \in I$ whenever $f \in I$ and $p \in \mathbb{K}[x_1, \ldots, x_n]$ is an arbitrary polynomial.

Proposition 8 *([CLO05]) Let $f_1, \ldots f_s \in \mathbb{K}[x_1, \ldots, x_n]$. Then the collection*

$$\langle f_1, \ldots f_s \rangle = \{p_1 f_1 + \cdots + p_s f_s \mid p_i \in \mathbb{K}[x_1, \ldots, x_n] \ \forall \ i = 1, \ldots s\}, \tag{10}$$

is an ideal and it is called the ideal generated by $f_1, \ldots f_s$

Recall the division algorithm for polynomials $f, g \in \mathbb{K}[x]$ in one variable, with $f, g \neq 0$. Let the degree of f be larger than the degree of g. Then there exists unique q and r in $\mathbb{K}[x]$ such that $f = qg+r$ and either $r = 0$ or the degree of r is less than the degree of g [BJN94]. In addition, $r = 0$ if and only if $f \in \langle g \rangle$. We wish to find a similar algorithm for polynomials in more than one variable. To do this we need some way to determine which monomial is largerst of $x^2y^2z^2$ and xy^4z for instance. This is where the concept of monomial order comes in.

Definition 9 *([CLO05]) A* monomial order *on* $\mathbb{K}[x_1, \ldots, x_n]$ *is any relation* $>$ *on the set of monomials* x^α *in* $\mathbb{K}[x_1, \ldots, x_n]$ *satisfying:*

1 $>$ *is a total (linear) ordering relation,*

2 $>$ *is compatible with multiplication in* $\mathbb{K}[x_1, \ldots, x_n]$*, in the sense that if* $x^\alpha > x^\beta$ *and* x^γ *is a monomial, then* $x^\alpha x^\gamma = x^{\alpha+\gamma} > x^{\beta+\gamma} = x^\beta x^\gamma$*, and*

3 $>$ *is a well-ordering. That is, every nonempty collection of monomials has a smallest element under* $>$*.*

Lexicographic ordering and graded reverse lexicographic ordering are two well known and frequently used monomial orders.

Definition 10 *([CLO05]) Let* x^α *and* x^β *be monomials in* $\mathbb{K}[x_1, \ldots, x_n]$*. We say* $x^\alpha >_{lex} x^\beta$ *if in the difference* $\alpha - \beta \in \mathbb{Z}^n$*, the leftmost nonzero entry is positive. This is called* lexicographic ordering.

Definition 11 *([CLO05]) Let* x^α *and* x^β *be monomials in* $\mathbb{K}[x_1, \ldots, x_n]$*. We say* $x^\alpha >_{grevlex} x^\beta$ *if* $\sum_{i=1}^n \alpha_i > \sum_{i=1}^n \beta_i$*, or if* $\sum_{i=1}^n \alpha_i = \sum_{i=1}^n$*, and in the difference* $\alpha - \beta \in \mathbb{Z}^n$*, the rightmost nonzero entry is negative. This is called the* graded reverse lexicographic ordering.

Example 12 *From the definitions we see that* $x^2y^2z^2 >_{lex} xy^4z$*, and* $x^2y^4z >_{lex} x^2y^2z^2$*, since in both examples the leftmost nonzero entry of* $\alpha - \beta$ *is positive. In this case* $\alpha - \beta$ *is equal to (1,-2,1) and (0,2,-1) respectively.*

For the graded reverse ordering we have that $x^2y^2z^3 >_{grevlex} xy^4z$*, and* $xy^4z >_{grevlex} x^2y^2z^2$*. This is because in the first example the sum of exponents is bigger on the left side (7) than on the right (6). In the second example the sum of exponents are equal, so the rightmost nonzero entry of* $\alpha - \beta$ *must be negative. In this case the difference is (-1,2,-1).*

Given a monomial order $>$ we can define the *leading term* of a polynomial f, denoted $LT_>(f)$ as the largest monomial of f under the ordering $>$. We are now ready to state the division algorithm for polynomials in more than one variable. Fix any monomial order $>$ in $\mathbb{K}[x_1, \ldots, x_n]$, and let $F =$

$(f_1, \dots, f_s)$ be an ordered s-tuple of polynomials in $\mathbb{K}[x_1, \dots, x_n]$. Then every $f \in \mathbb{K}[x_1, \dots, x_n]$ can be written as

$$f = q_1 f_1 + \cdots + q_s f_s + r, \tag{11}$$

where $q_i, r \in \mathbb{K}[x_1, \dots, x_n]$, for all i, $q_i f_i = 0$ or $LT_>(f) \geq LT_> q_i f_i$, and either $r = 0$, or r is a linear combination of monomials, none of which is divisible by any of $LT_>(f_1), \dots, LT_>(f_s)$. We call r a *remainder* of f on division by F and is denoted $\overline{f}^F$ [CLO05].

Unfortunately, the division algorithm for multivariate polynomials does not have all the nice properties that the one variable version have. For instance, for any f and F there are always *different* expressions of the form in equation 11 depending on the particular algorithm used for division. Reordering F, or changing the monomial order, may also produce different q_i and r. A particularly undesirable property is that $r \neq 0$ does not necessarily mean that $f \notin \langle F \rangle$. The reason why this might happen is because the remainder r may also be an element in the ideal $\langle F \rangle = \langle f_1, \dots, f_s \rangle$, and $r \neq 0$ because it contains terms that cannot be removed by these particular generators of $\langle F \rangle$. In order for a division to produce zero remainders for all elements of an ideal, we need to be able to remove *all* leading terms of elements of that ideal using the leading terms of the divisors [CLO05]. That is the motivation for the definition of Gröbner bases.

Definition 13 *([CLO05]) Fix a monomial order $>$ on $\mathbb{K}[x_1, \dots, x_n]$, and let $I \subset \mathbb{K}[x_1, \dots, x_n]$ be an ideal. A Gröbner basis for I (with respect to $>$) is a finite collection of polynomials $G = \{g_1, \dots, g_t\} \subset I$ with the property that for every nonzero $f \in I$, $LT_>(f)$ is divisible by $LT_>(g_i)$ for some i.*

Gröbner bases behave much more nicely with respect to the division algorithm for multivariate polynomials. In particular we have the following two results.

Proposition 14 *([CLO05]) Let $G = \{g_1, \dots, g_t\}$ be a Gröbner basis for an ideal I. Then $I = \langle g_1, \dots, g_t \rangle$, and for any $f \in I$ the remainder on division of f by G is zero.*

Theorem 15 *([CLO05]) Fix a monomial order $>$ and let $I \subset \mathbb{K}[x_1, \dots, x_n]$ be an ideal. Division of $f \in \mathbb{K}[x_1, \dots, x_n]$ by a Gröbner basis for I produces an expression $f = g + r$ where $g \in I$ and no term in r is divisible by any element of $LT(I)$. If $f = g' + r'$ is any other such expression then $r = r'$*

We are interested in using Gröbner bases as a tool for solving systems of polynomial equations. Let

$$f_1(x_1, \dots, x_n) = \cdots = f_m(x_1, \dots, x_n) = 0 \tag{12}$$

be a system of m polynomials in n unknowns over the field $\mathbb{K}$. The set of solutions in $\mathbb{K}$, which is the *algebraic variety*, is defined as

$$V = \{(z_1, \dots, z_n) \in \mathbb{K}^n \mid f_i(z_1, \dots, z_n) = 0 \ \forall \ 1 \le i \le m\}. \tag{13}$$

For cryptographic applications we are often interested in multivariate quadratic equations. For this purpose we have the following results.

Proposition 16 *([FJ03]) Let G be a Gröbner basis of $\langle f_1, \dots, f_m, x_1^2 - x_1, \dots, x_n^2 - x_n\rangle \subset GF(2)[x_1, \dots, x_n]$. Then the following holds:*

1 $V = \emptyset$ (no solution) iff $G = \{1\}$.

2 V has exactly one solution iff $G = \{x_1 - a_1, \dots, x_n - a_n\}$ where $a_i \in GF(2)$. Then $(a_1, \dots, a_n)$ is the solution in $GF(2)$ of the algebraic system $f_1 = \cdots = f_m = 0$.

2.3.2 Algorithms for Computing Gröbner Bases

I will in this section describe three different algorithms for computing Gröbner bases, starting with the classical algorithm by Buchberger, before presenting two different improvements due to Faugère.

Buchberger's Algorithm. The classical algorithm for computing Gröbner bases was introduced by Bruno Buchberger in his PhD-thesis [Buc65]. In addition to providing the algorithm, he proved that the algorithm, which takes an arbitrary generating set $\{f_1, \dots, f_s\}$ for an ideal I as input, always terminates and produces a Gröbner basis for I. Before we state the algorithm we need to introduce the S-polynomial.

Definition 17 *([CLO05]) Let $f, g \in \mathbb{K}[x_1, \dots, x_n]$ be nonzero. Fix a monomial order and let $LT(f) = cx^\alpha$ and $LT(g) = dx^\beta$, where $c, d \in \mathbb{K}$. Let x^γ be the least common multiple of x^α and x^β. Then the S-polynomial of f and g, denoted $S(f, g)$, is the polynomial*

$$S(f, g) = \frac{x^\gamma}{LT(f)} \cdot f - \frac{x^\gamma}{LT(g)} \cdot g. \tag{14}$$

The algorithm works as a generalization of the Euclidean division of polynomials in one variable, and is based on the following theorem called Buchberger's criterion.

Theorem 18 *([CLO05] A finite set $G = \{g_1, \dots, g_t\}$ is a Gröbner basis of $I = \langle g_1, \dots, g_t\rangle$ if and only if $\overline{S(g_i, g_j)}^G = 0$ for all pairs $i \ne j$.*

We are now ready to present the algorithm. We will use the description found in [CLO05].

```
**Algorithm** *Buchberger's algorithm*(F)
**Input:** $F = \{f_1, \dots, f_s\}$
**Output:** a Gröbner basis $G = \{g_1, \dots, g_t\}$ for $\langle F \rangle$
1.  $G \leftarrow F$;
2.  repeat
3.      $G' \leftarrow G$;
4.      for each pair $p \neq q \in G'$
5.          do $S := \overline{S(p,q)}^{G'}$;
6.              if $S \neq 0$ then $G \leftarrow G \cup \{S\}$;
7.  until $G = G'$
8.  return $G$;
```

The algorithm is easy to understand, and a direct implementation would be fairly straightforward. Unfortunately, such an implementation would be very inefficient in practice. The main reason for this is that, in general, more than 90% of the computations in step 5 of the algorithm results in 0, and the running time of the algorithm is highly correlated with the number of such computations [Fau02]. The pairs considered in step 4 are called critical pairs, and the computation of the remainder in step 5 is called the reduction of that pair. If the reduction of a critical pair is 0, that pair is called useless, since it provides no information about the Gröbner basis under computation. Buchberger [Buc65, Buc79, Buc85], Gebauer and Möller [GM86], and Mora et al. [MMT92] have all suggested improvements trying to diminish the number of useless critical pairs. According to Faugère [Fau02], the efficiency of those algorithms is not yet satisfactory in theory and in practice, because a lot of useless critical pairs are not removed. Faugère has suggested two different strategies for improving the computation of Gröbner bases. These will be discussed in the next two paragraphs.

Faugère's F_4 Algorithm. The idea behind Faugère's F_4 Algorithm [Fau99] can be described by first looking at Euclidean division of polynomials in one variable (as done in [AACW08]). Let $f = f_d x^d + \cdots + f_1 x + f_0$ and $g_{d'} x^{d'} + \cdots + g_1 x + g_0$ be two polynomials in $\mathbb{K}[x]$ with $d' \leq d$. Consider the following matrix.

$$\begin{array}{l c}
 & \begin{array}{cccccccc} x^d & x^{d-1} & & & & & & x^0 \end{array} \\
\begin{array}{l} f \\ x^{d-d'}g \\ x^{d-d'-1}g \\ x^{d-d'-2}g \\ \vdots \\ g \end{array} &
\begin{pmatrix}
f_d & f_{d-1} & \cdots & & & & & f_0 \\
g_{d'} & g_{d'-1} & \cdots & g_0 & & & & \\
0 & g_{d'} & g_{d'-1} & \cdots & g_0 & & & \\
0 & 0 & g_{d'} & g_{d'-1} & \cdots & g_0 & & \\
\vdots & & & \ddots & & & \ddots & \\
0 & 0 & 0 & 0 & g_{d'} & g_{d'-1} & \cdots & g_0
\end{pmatrix}
\end{array} \tag{15}$$

The remainder of f on division by g can be computed by successive row echelon reduction of the first row by the remaining rows. Similarly the multivariate division algorithm can be described in a matrix fashion. The F_4 algorithm takes advantage of this. At each iteration of the F_4 algorithm every critical pair (under a selection strategy to remove pairs reducing to zero), together with all already known polynomials, and the current basis G are multiplied by fitting monomials and written in a global matrix M_{F_4}. The multiplication with fitting monomials is done in the same way as the polynomial g is shifted in equation 15. The matrix M_{F_4} is then reduced to row echelon form in one huge computation. This corresponds to one iteration of the Buchberger algorithm, so roughly speaking the F_4 algorithm is the Buchberger algorithm in matrix form.

It has been shown experimentally that the implementation of the F_4 algorithm is much faster then most other algorithms for computing Gröbner bases (except F_5) [Fau99]. An implementation of the F_4 algorithm can for instance be found in Magma [MAG].

Faugère's F_5 Algorithm. Loosely speaking, the strategy of the F_5 algorithm [Fau02] for computing the Gröbner basis of a set of polynomials $(f_1, \dots, f_s)$ is to only consider trivial combinations of the form $f_i f_j - f_j f_i$. This strategy implies two major differences from the Buchberger algorithm. The first is that we need to iteratively compute all the Gröbner bases of the following ideals $(f_s), (f_{s-1}, f_s), \dots, (f_1, \dots, f_s)$. The second difference is that some reductions are not allowed. As a result of this, the reduction of one polynomial by a list of polynomials may be several polynomials. Faugère has proven that this strategy avoids all useless critical pairs if the input is a regular sequence, and has shown experimentally that most examples have no reductions to zero in practice [Fau02]. This is also supported with experimental results where the F_5 algorithm is sometimes up to one order of magnitude faster then F_4. In addition to this, the algorithm was used to compute the Gröbner basis of Cyclic 10 for the first time [Fau02]. A public implementation has been done by Till Stegers [Ste06], however the implementation is not designed for efficiency.

2.3.3 Complexity of Computing Gröbner Bases

For cryptanalysis it is often important to know the complexity of a computation. Often the computation may be way to big for all practical purposes, but finding the expected running time for the computation is still interesting. The reason for this is that the expected running time might be lower than a brute-force attack on the cryptosystem on which the computational problem is based. We would then have a theoretical break of that cryptosystem.

The complexity of computing a Gröbner basis of an ideal I depends on the maximum degree of the polynomials appearing during the computation. This degree, called *degree of regularity*, is the key parameter for understanding the complexity of a Gröbner basis computation [BFS04]. Indeed, the complexity of the computation is exponential in the degree of regularity D_{reg}, more precisely the complexity is:

$$\mathcal{O}(n^{\omega D_{\mathrm{reg}}}), \tag{16}$$

which basically correspond to the complexity of reducing a matrix of size $\approx n^{D_{\mathrm{reg}}}$. Here $2 < \omega \leq 3$ is the "linear algebra constant", and n the number of variables of the system. Note that D_{reg} is also a function of the number of variables, n, and the number of equations m. The relation between D_{reg} and m, n depends on the specific system of equations. This relation is well understood for regular (and semi-regular) systems of equations [Bar04, BFS04, BFS02, BFSY05]. However, as soon as the system has some kind of structure, this degree is much more difficult to predict. Since all cryptosystems have some underlying structure, this means that all Gröbner bases based cryptanalysis should try to find some bound on the degree of regularity. This has for instance been done in the works on HFE ([FJ03, GJS06]).

2.3.4 Gröbner Bases Cryptanalysis

Since it is very natural to use Gröbner bases to attack cryptosystems based on multivariate quadratic equations a lot of work has been done in this area. According to [BD09] Dobbertin reported successfully solving instances of MIA using Gebauer and Möller's version of Buchberger's algorithm for computing Gröbner bases while working at the German federal office for information security (BSI). However, the first published algebraic attack on MIA is due to Patarin [Pat95]. The attack is very instructive in the sense that it shows that there exist bilinear equations relating the input and the output of the system.

These low degree equations help explain why Gröbner bases can successfully solve the equation system.

A lot of work has been done on Gröbner bases based cryptanalysis of different instances of HFE. Faugère and Joux [FJ03] used Gröbner bases to solve the 80-bit HFE challenge. They explained this by showing how the complexity of computing Gröbner bases of HFE was bound only by the degree d of the secret polynomial S, and not the size of the field $\mathbb{F}_{2^n}$. In [FJ03] the degree d is viewed as a constant, while later work by Granboulan et al. [GJS06] on HFE considers instances where d increases with the size of the field. Setting $d = n^\alpha, \alpha \geq 1$, they consider the complexity of Gröbner bases based decryption attacks. By bounding the degree of regularity they showed this has quasipolynomial complexity. Courtois et al. [CD03] used Gröbner bases to analyze the security of different parameters of HFEv-. Using the Buchberger algorithm they looked at smaller instances of HFEv- and compared the running time of the algorithm with the running time of the same algorithm on a random system of equations. They used the results to analyze the security of Quartz, and recommend parameters resulting in a higher degree of security. In [BFJT09], Bouillaguet et al. showed that when the secret key was defined in the base field instead of the extension field, the secret key can be recovered. Bouillaguet et al. used Gröbner bases as part of this attack.

Studying the security of UOV, Braeken et al. [BWP05] looked at the running time of Magma implementation of Faugères F_4 algorithm when fixing some of the unknown variables to random bits. They found that some choices of parameters are particularly vulnerable to this attack. Later work by Faugère and Perret [FP09] extends on the analysis of [BWP05]. This is done by analyzing the effect of fixing $r = 1, 2, 3$ of the unknown variables to a bit chosen at random on the degree of regularity. The conclusion is that UOV is insecure when the parameter choice is as suggested by [KHPG99].

Gröbner bases have also been used to improve other types of attacks on MQ schemes. In [FP06b], Perret and Faugère use Gröbner bases to improve on the isomorphism of polynomials problem. This technique is subsequently used to break several challenges in multivariate cryptography. In [FP06a] Faugère and Perret use Gröbner bases in a new algorithm for solving the functional decomposition problem, and in effect breaks the MQ signature scheme $2R^-$. In [FdVP08] Levy-dit-Vehel et al. uses Gröbner bases to improve on Kipnis and Shamir's MinRank attack [KS99]. The MinRank problem is NP-complete, and is considered a very important problem in multivariate cryptography [FdVP08]. In the paper, the improved attack is used to break a multivariate authentication scheme.

According to a survey on block ciphers and algebraic cryptanalysis by Cid and Weinmann [CW09], the first documented use of Gröbner bases in analysis of symmetric-key cryptography was an improvement of the linear cryptanalysis of DES [SK98]. In other works on block ciphers Cid et al. [CMR05] applied Gröbner bases to the analysis of a scaled down version of AES to give some insight into solving block ciphers with algebraic techniques. Buchmann et al. [BPW06a] describe how to compute a Gröbner basis of a zero-dimensional ideal describing the key-recovery problem from a single plaintext/ciphertext pair for the full AES-128. However, the described method is not efficient enough for successful cryptanalysis. Another paper by Buchman et al. [BPW06b] showed how there exist block ciphers that are resistant to classical attacks, but are susceptible to Gröbner bases cryptanalysis. This was done by constructing a Feistel network based cipher Flurry, and a substitution-permutation network based cipher Curry; both with the described characteristics[5].

On the subject of stream ciphers, Ars and Faugère [AF03] used Gröbner bases to successfully attack non-linear filter generators. In the paper they take a closer look at ciphers satisfying Goli̒c‘s design criteria. For these ciphers they show how the algebraic degree is much lower then the theoretical bound. Because of this, the secret key of many 80-bit stream ciphers can be recovered. In another paper by the same authors they introduce a concept of algebraic immunities and show how it can be used to attack stream and block ciphers [AF05]. More information can be found in a survey of Gröbner bases and stream ciphers written by Armknecht and Ars [AA09].

Work on Gröbner bases applied to other cryptographic primitives has for instance been done by Faugère et al. [FOPT10]. In the paper they solved variants of the McEliece public-key cryptosystem [McE78], which is based on the difficulty of decoding linear code. We also recall that Sugita et al. [SKPI07] have applied a combination of differential and algebraic cryptanalysis to 58-round SHA-1, as mentioned in Section 2.1.4.

3. Research Goals

In Section 2.1.5 we saw how important secure cryptographic hash functions are in the construction of cryptographic systems. Hash functions are being used as a building block in such diverse primitives as commitment schemes, message authentication codes and digital signatures. These primitives have important applications by themselves, and they can also be used in the construction of more complex protocols such as electronic voting systems, online auctions, public-key distribution, mutual authentication handshakes and more.

[5]These two papers by Buchman et al. was also part of Pyshkins PhD thesis [Pys08].

Because of the importance of hash functions in cryptography, and because of the prestige of winning the SHA-3 competition organized by NIST, a lot of the cryptographic communities focus is directed towards hash functions in the duration of the contest. The writing of the research plan for this thesis coincided with NIST's public call for cryptographic hash functions. It was a natural choice to have contributing to the SHA-3 competition, both with submission of a candidate and with cryptanalysis of candidates, as one of the research goals for this thesis.

In 2002 Courtois and Pieprzyk presented a new algebraic technique based on the XL method [CP02]. The new method they called XSL uses the sparsity of the equations and their specific structure to solve the equation system. In the paper they suggest that the XSL attack might be able to break Rijndael (AES) 256 bits [DR98] and Serpent [ABK98] for key lengths 192 and 256 bits. Even though the XSL method today is widely recognized as being incorrect, its publication can be considered as a key event that fueled the interest of the cryptographic community in algebraic methods applied to cryptography [CW09]. This interest, together with Faugère's improved algorithms (F_4 and F_5) for computing Gröbner bases, is probably one of the reasons for the numerous and comprehensive results on Gröbner bases as a cryptanalytical tool in the last decade (see Section 2.3.4). All these results, and the knowledge of how powerful Gröbner bases are as a tool for solving systems of multivariate equations, were the motivation for adding Gröbner bases cryptanalysis of SHA-3 candidates as a research goal of this thesis.

During the work on this thesis, we learned that direct Gröbner bases cryptanalysis of hash functions is difficult since the corresponding algebraic systems are so huge that it is very difficult to predict the complexity of solving such polynomial systems. The current state of the art uses Gröbner bases to improve on other cryptanalysis results, such as differential cryptanalysis. Because of the nature of the SHA-3 competition, the existence of differential cryptanalysis results on a candidate will most likely exclude it from the list of viable contenders. Since the focus of the cryptographic community is (and should be) on the remaining candidates, improvement of results of discarded candidates is not a priority. Because of this, and because we still wanted applications of Gröbner bases to be a research goal of this thesis, we found it natural to extend this goal to include hash function applications. For this purpose we decided to focus on cryptanalysis and construction of multivariate quadratic signature schemes.

To summarize, the following are the research goals of this thesis.

1 Contribute to the National Institute of Standards and Technologies public competition to develop a new cryptographic hash algorithm; both with submission of a candidate, and with cryptanalysis of candidates.

2 To apply Gröbner bases as a tool for cryptanalysis of hash functions and hash function applications.

4. Research Methodology

To reach the research goals I have applied a combination of mathematical and experimental approach. In the experimental approaches I followed a traditional scientific methodology starting with a hypothesis which is tested before the results from the test are validated.

For our construction of a canditate for NIST's hash competition we based ourselves on the known theory of hash function design presented in Section 2.1.3. Using this and quasigroup theory we were able to construct a hash function that is provably resistant to differential cryptanalysis. However, by the nature of hash functions (that are not based on some hard problem), we were not able to generally prove mathematically the security of the proposed hash function. Because of this we also chose an experimental approach. In one of the experiments we hypotesized that our hash function is an ideal random function, and then we performed statistical experiments to test this hypothesis.

For our cryptanalysis of other SHA-3 candidates we chose a mathematical approach based on the theory presented in Section 2.1.4. In particular we constructed a search algorithm for finding differential paths based on some mathematical observations we made when analyzing the internal structure of one of the canditates.

For our cryptanalysis of a multivariate quadratic signature scheme we started with the hypothesis that the equation system behaved as a set of random quadratic equations. This hypothesis was proven wrong by using experiments measuring the degree of regularity when solving the system with Gröbner bases. Using this method we found that the degree of regularity did not behave as expected. We then used a mathematical approach to explain this observed degree of regularity.

For our construction of a multivariate quadratic signature scheme we based our work on the theory presented in Section 2.2 and the theory of quasigroups. We also performed experiments with Gröbner bases. In these experiments we measured the degree of regularity when solving different systems with Gröbner bases. For these experiments we wanted to find the best choice of quasigroups with respect to Gröbner bases analysis, and we wanted to show that our resulting system behaved as a random system of equations with respect to Gröbner bases.

5. Contributions

This section presents the main results of each of the six papers included in Part II of this thesis. The author's contribution to each paper is specified.

5.1 Paper A

On Some Properties of Boolean Matrices from Latin Squares

The paper presents an interesting class of Boolean matrices we found during construction and cryptanalysis of the hash function EDON-$\mathcal{R}$. Let $\mathbb{1}$ be the all 1 Boolean matrix of appropriate size. We found that under a special map we defined from Latin squares to a pair of Boolean matrices $(\mathbb{A}_1, \mathbb{A}_2)$, $\mathbb{A}_1$ is non-singular if and only if $\mathbb{A}_2$ is non-singular. In addition they have the property that $\mathbb{A}_1 + \mathbb{A}_2 = \mathbb{1}$, and if the matrices are non-singular the same holds for the inverse matrices as well $\mathbb{A}_1^{-1} + \mathbb{A}_2^{-1} = \mathbb{1}$. Note that this is not necessarily true if we randomly produce a non-singular matrix $\mathbb{A}_1$ and set $\mathbb{A}_2 = \mathbb{A}_1 + \mathbb{1}$.

The author's contribution to Paper A, in addition to writing the paper, was mathematically proving both lemmas and the theorem presented under results. Danilo Gligoroski presented the problem to the author. Both Danilo Gligoroski and Marija Mihova provided valuable feedback during the author's work on the paper.

5.2 Paper B

Cryptographic Hash Function EDON-$\mathcal{R}$

The paper is the supporting document for the cryptographic hash function EDON-$\mathcal{R}$ which was submitted as a candidate for SHA-3 hash competition organized by NIST, according to the public call [NIS07].

EDON-$\mathcal{R}$ is a dedicated wide piped Merkle-Damgård iterated hash function. The compression function is based on a form for quasigroup string transformation, where the underlying quasigroup operation is a permutation based on the three basic operations addition, rotation and eXclusive OR.

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: $2^{\frac{n}{2}}$ hash computations for finding collisions, 2^n hash computations for finding preimages, 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-$\mathcal{R}$ was designed to be much more efficient than SHA-2 cryptographic hash functions, while at the same time offering the same or better security.

The author's contributions to Paper B are listed below.

- Helped in the preparation and writing of the paper.
- Provided insight and analysis to some of the underlying mathematical properties, including the part also presented in paper A.
- Performed experiments on the effect on the choice different Latin squares.
- Performed experiments on the avalanche properties of the hash function.
- Performed statistical analysis which provided confidence in the randomness of the hash function, including the analysis presented in paper C.
- Performed experiments and mathematical analysis on a special class of the compression function of EDON-$\mathcal{R}$, where for some initial values the compression function behaved like a one-way permutation. This property was only found in scaled down versions of the compression function.

5.3 Paper C

On the Randomness and Regularity of Reduced EDON-$\mathcal{R}$ *Compression Function*

The paper presents results from statistical analysis performed on the compression function of EDON-$\mathcal{R}$, where the underlying quasigroup operations are reduced to sizes 2^8 and 2^{16}. The results show that the reduced compression function $\mathcal{R}_8$ is regular and that its output distribution is similar to that of an ideal random function. We also show that distribution of the output of $\mathcal{R}_{16}$ is very similar to an ideal random function, and that the Bellare-Khono balance of $\mathcal{R}_{16}$ is high.

The author's contribution to this paper was writing the paper, coding and performing the experiments, and the statistical analysis. Danilo Gligoroski presented valuable feedback throughout the work on this paper.

5.4 Paper D

Distinguishers for the Compression Function and Output Transformation of Hamsi-256

The paper presents a study of Hamsi's resistance to differential and higher-order differential cryptanalysis, with focus on the 256-bit version of Hamsi. The main results are efficient distinguishers and near-collisions for its full (3-round) compression function, and distinguishers for its full (6-round) finalization function, indicating that Hamsi's building blocks do not behave ideally. Note that when the paper was presented at ACISP 2010, Hamsi was one of 14 remaining candidates in NIST's Hash Competition for the future hash standard SHA-3.

The author was part of a group that started to work on differential cryptanalysis of Hamsi and the randomized search algorithm at a hash workshop in Graz, Austria. In preparation of this paper the author independently wrote a program that confirmed the differential path, found the 4 messages satisfying the relaxed first round differential, and computed the corresponding chaining values which satisfy the differential path up to four rounds. The author also presented the paper at ACISP 2010.

5.5 Paper E

Analysis of the MQQ Public Key Cryptosystem

The initial motivation for this paper was our desire to design a fast and secure public key cryptosystem based on multivariate quadratic quasigroups. To achieve this we first needed to understand why Gligoroski et al.'s previous attempt [GMK08b] was so easy to solve in practice [MDBW09].

The paper presents an algebraic cryptanalysis of the MQQ Public Key Cryptosystem. Using a Gröbner bases based approach we explain why systems arising in MQQ are so easy to solve in practice. To do so, we consider the so-called degree of regularity; which is the exponent in the complexity of a Gröbner basis computation. For MQQ systems, we show that this degree is bounded from above by a small constant. This is due to the fact that the complexity of solving the MQQ system is the minimum complexity of solving just one quasigroup block or solving the Dobbertin transformation. Furthermore, we show that the degree of regularity of the Dobbertin transformation is bounded from above by the same constant as the bound observed on MQQ system. We then investigate the strength of a tweaked MQQ system where the input of the Dobbertin transformation is replaced with random linear equations. It appears that the degree of regularity of this tweaked system varies both with the size of the quasigroups and the number of variables. We conclude that if a suitable replacement for the Dobbertin transformation is found, MQQ can possibly be made strong enough to resist pure Gröbner attacks for adequate choices of quasigroup size and number of variables.

The author's contribution to this paper was constructing the quasigroups, constructing the various MQQ systems, and performing all experiments with Gröbner bases. Faugère contributed with his expertise on Gröbner bases, and suggested various approaches and experiments. Perret contributed significantly during the writing of the paper. He also contributed with knowledge on Gröbner bases computation and coding in Magma. Gligoroski contributed with his knowledge of quasigroups and the MQQ cryptosystem.

5.6 Paper F

MQQ-SIG. An Ultra-fast and provably CMA Resistant Digital Signature Scheme

The paper presents a digital signature scheme based on multivariate quadratic quasigroups. The signature scheme is designed to be very fast in both software and hardware. Signing can be performed in about 1100 cycles on a modern Intel processor. This is three orders of magnitude faster then RSA and ECDSA.

From a security point of view the signature schemes offers a conjectured $\frac{n}{2}$ bits of security, with $n = 160, 192, 224, 256$. To ensure this security the well-known minus modifier for multivariate quadratic schemes is used. In addition, the paper presents the results of extensive set of experiments investigating the properties of systems of $n-k, 0 \leq k \leq \frac{n}{2}$, MQQ equations in n variables. These results give confidence that the equation system behaves as random multivariate equations when enough equations is removed. The signature scheme is proven to be secure under a chosen message attack based on this assumption.

The author's contribution to this paper was extensive research and experiments with Gröbner bases. This research is summarized below:

- Analysis of the broken MQQ public key cryptosystem finding the underlying weaknesses of the old scheme. This work was presented in Paper E.

- Analysis of many different MQQ systems. In these systems the effects of using different types and size of MQQs, and different numbers of unique MQQs was tested. The conclusion was that after removing enough equations, these choices had no impact on the security of the schemes with respect to Gröbner bases cryptanalysis.

- The experiments which give confidence in the signature scheme's resilience to Gröbner bases attacks when removing half of the equations. These results are presented in Table 5 in the paper.

In addition the author wrote the proof showing that MQQ-SIG is secure under chosen message attack in the random oracle model assuming that solving $\frac{n}{2}$ MQQ equations with n variables is as hard as solving systems of $\frac{n}{2}$ random multivariate quadratic equations.

Danilo Gligoroski was the main designer of both the old MQQ scheme, and this signature scheme. Rune Erlend Jensen wrote the implementation of MQQ-SIG in C. He also contributed to the design phase with ideas on how to keep public-key size low and software speed high. Ludovic Perret and Jean-Charles Faugère both contributed with their knowledge of Gröbner bases computation. Svein Johan Knapskog and Smile Markovski where both contributors to the old MQQ scheme.

6. Results and Discussion

I will in this section further discuss this thesis contribution.

6.1 Boolean Matrices from Latin Squares

For our construction of Edon-$\mathcal{R}$, the relationship presented in Paper A was important. The reason for this is that the maps are used to define the underlying quasigroup operation in Edon-$\mathcal{R}$'s compression function. The property that $\mathbb{A}_1 + \mathbb{A}_2 = \mathbb{1}$ plays an important role in the diffusion of bit differences in the compression function. Having $\mathbb{A}_1^{-1} + \mathbb{A}_2^{-1} = \mathbb{1}$ as well means that we would have the same desirable diffusion properties for the inverse quasigroup operation no matter what Latin squares we chose for the compression function. This made our search for the best possible latin square easier.

This interesting class of Boolean matrices might have applications in other areas of cryptography or applied mathematics in general. To our knowledge there has been no published work on the relationship between Latin squares and Boolean matrices.

6.2 Dedicated Hash Function Edon-$\mathcal{R}$

Most of the aspects of hash functions design has been discussed in Section 2.1.3. A nice overview of the general history of hash function design can for instance be found in Section 9.8 of [MVO96]. We will here discuss Edon-$\mathcal{R}$ and some contemporary work on dedicated hash functions.

As discussed in Section 2.1.2, NIST has recommended transition to the SHA-2 family [NIS02] of hash functions because of the reported weaknesses of SHA-1. In addition, they are currently holding a competition, called the "SHA-3 contest," to develop the new hash function standard. Comparing with the thesis author's contribution Edon-$\mathcal{R}$ we see that of the 51 submission accepted for round 1 of the contest, NaSHA [MM08] is the only other hash function based on quasigroup operations. BLAKE [AHMP08], Blender [Bra08], Blue Midnight Wish [GKK+08], CubeHash [Ber08b], Dynamic SHA2 [Xu08], EnRUPT [ONH08], Sgáil [Max08], Skein [FLS+08] and Tangle [AMZ08] are the only other submissions based on the three basic operations addition, rotation and eXclusive OR. A nice overview of all submission can be found at the "SHA-3 zoo" [sha]. An overall classification of all candidates has been done by Fleischmann et al. [FFG08].

In addition to the standard security requirements of cryptographic hash functions, NIST wanted the SHA-3 candidates to be significantly faster then SHA-2 [NIS07]. This requirement was added to ease the transition from the currently used hash functions to the winner of the SHA-3 contest. It is speculated that the industry's unwillingness to move away from the respectively broken and weakened hash functions MD5 and SHA-1 is the much slower speed of SHA-2 (see for instance [GKAJ11, GKS11]). If we take a closer look at the implementation of all SHA-3 candidate hash functions from round 1, Edon-$\mathcal{R}$ is the fastest hash function submitted to the contest (on most platforms) [Be11b]. This is shown in Table 2 where speed of Edon-$\mathcal{R}$ in cycles/byte is compared to the SHA-3 finalists[6]. Other round 1 candidates where either the 32-bit or 64-bit implementation is significantly faster then SHA-2 are ARIRANG [CHK+08], Blue Midnight Wish, EnRUPT, Luffa [CSW08], Lux [NBK08] and MeshHash [Fay08].

Of the 51 submissions accepted to round 1 of the competition Blue Midnight Wish, CubeHash, ECHO [BBG+08], Fugue [HHJ08], Hamsi [KÖ8], Luffa, Shabal [BCCM+08], SHAvite-3 [BD08] and SIMD [LBF08] were the nine hash functions accepted for round 2 of the contest that did not make it to round 3. At the time of writing of this thesis, BLAKE, Grøstl [GKM+08], JH [Wu09], Keccak and Skein are the five remaining candidates in the contest. NIST made this selection primarily based on the following three criteria: Security, cost and performance, and algorithm and implementation characteristics. A detailed report on the selection of round 3 candidates has been published by NIST [NIS11].

6.3 Cryptanalysis of Edon-$\mathcal{R}$

The work presented in Paper C is the only published statistical analysis of Edon-$\mathcal{R}$. For this paper we chose to work on reduced versions of the hash

[6]Note that the implementation of Edon-$\mathcal{R}$ is pure C-code, while the implementation all others in the table are heavily optimized assemblers. We could expect up to 50% speed gains for an optimized assembler code of Edon-$\mathcal{R}$.

Table 2. Speed comparison EDON-$\mathcal{R}$ and the SHA-3 finalists in cycles/byte. The computations was performed on the following computer: amd64; Sandy Bridge (206a7); 2011 Intel Core i7-2600K; 4 x 3400MHz [Be11b].

Hash function	cycles/byte
EDON-$\mathcal{R}$-512	2.70
EDON-$\mathcal{R}$-256	5.22
Skein-512	7.83
Blake-256	7.87
Blake-512	7.94
Skein-256	9.65
Grøstl-256	11.51
Keccak-256	12.84
JH-512	13.70
JH-256	13.71
Grøstl-512	15.59
Keccak-512	23.98

function. The reason for this is that we wanted a better understanding of the general structure of the design and the complete output distribution of the construction.

In other work on EDON-$\mathcal{R}$, Khovratovic et al. [KNW08] presented various free-start attacks together with a meet-in-the-middle preimage attack requiring $O(2^{\frac{2n}{3}})$ time and $O(2^{\frac{2n}{3}})$ memory. The validity of the model in which Khovratovic's meet-in-the-middle attack is compared to generic attacks was later disputed by Gligoroski and the author [GØ09]. In another work, which partially overlaps Khovratovic et al.'s cryptanalysis, Klima [Kli08] presented a multi-collision attack together with some other observation on EDON-$\mathcal{R}$. Leurent [Leu09] showed how the secret key leaks if EDON-$\mathcal{R}$ is used in an outdated secret-prefix MAC construction. In [NF09] Novotney and Ferguson used differential cryptanalysis to produce detectable output bit biases.

6.4 Cryptanalysis of Hamsi

All techniques used in our cryptanalysis of the Hamsi compression function have been discussed in Section 2.1.4. This paper is the only published work on Hamsi's resistance to differential cryptanalysis, and played a partial role why Hamsi was not selected for round three of the SHA-3 competition [NIS11]. Below we mention other cryptanalytical results on Hamsi that also influenced this results.

Starting with results closely related to our work Nikolić [Nik09], Turan and Uyan [TU10], and Wang et al. [WWJW09] have all presented results on pseudo-near-collisions of Hamsi, while Aumasson and Meier [AM09] and Boura and Canteau[CB10] have presented results on Zero-Sum distinguishers of the algorithm.

In other work on Hamsi, Çalik and Turan [ÇT10] presented message recovery and pseudo-preimage attacks on the compression function of Hamsi-256. Other work on second preimage attacks of Hamsi-256 was presented by Fuhr in [Fuh10], where he showed how to find second preimages for short messages. Dinur and Shamir [DS10] exploited the low algebraic degree of Hamsi-256 in an algebraic second preimage attack. In [Gli10], Gligoroski showed how all narrow-pipe SHA-3 candidates, including Hamsi, differ significantly from ideal random functions defined over big domains.

6.5 Gröbner Bases Based Cryptanalysis of MQQ

The MQQ [GMK08b] cryptosystem had already been shown weak by Mohamed et al. [MDBW09]. However [MDBW09] only solved the MQQ equation system using both XL and Gröbner bases approach, and did not explain why the system was so easy to solve in practice. The results presented in Paper E shows exactly why the system is susceptible to algebraic cryptanalysis. This result is crucial in our attempt to construct a secure digital signature scheme based on MQQ, which is the work presented in Paper F.

We know of no other published results on cryptanalysis of the MQQ cryptosystem. Gröbner bases based cryptanalysis of MQ systems in general was discussed in Section 2.3.4.

6.6 Multivariate Quadratic Signature Scheme MQQ-SIG

Most aspects of MQ signature schemes have been discussed in Section 2.2. The proposed signature scheme in Paper F is the first published security improvement in the MQQ family of MQPKC. The improvements are partially based on the security problems we found during our cryptanalysis of the old MQQ cryptosystem [GMK08b], which is the work presented in Paper E.

In Table 3 we compare MQQ-SIG to the MQ schemes enhanced TTS [YC05] and 3ICP [DWY07]. We see that MQQ-SIG is much better in signing and private key size, while worse in key generation, verification and public key size. This means that MQQ-SIG is a good alternative for protocols where the constrained environment is on the side of the signer.

Table 3. Comparison of selected MQ schemes with security level 2^{128}. Signing and verification is in CPU cycles for a 59 byte message. Private and public key represent size in bytes.

Algorithm	Key Generation	Signing	Verification	Private key	Public key
MQQSIG256	4,839,469,440	4,948	138,324	593	526,368
TTS6440	60,827,704	84,892	76,224	16,608	57,600
3ICP	15,520,100	1,641,032	60,856	12,768	35,712

7. Summary and Conclusion

Recall that the research goals for this thesis were to contribute to the SHA-3 competition, and to use Gröbner bases as a tool for cryptanalysis of hash function applications. An overview of all papers included in this thesis, and how they relate to each other and the research goals, is presented in Figure 7. The hash function presented in Paper B was our candidate for the SHA-3 competition, and both Paper A and Paper C are strongly connected to this construction. Paper D presents our cryptanalysis of Hamsi, and as mentioned, the paper played a partial role for why Hamsi was not selected for round three of the SHA-3 competition. Both Paper E and Paper F present work strongly connected with Gröbner bases cryptanalysis of digital signature schemes, which we recall is one of the more common and important hash function applications. Paper E presents Gröbner bases based cryptanalysis of the public-key encryption and signature scheme MQQ. The work in Paper E, together with more experiments with Gröbner bases, resulted in the signature scheme presented in Paper F. The dashed line in Figure 7 indicates that EDON-$\mathcal{R}'$ [GØM+09a], which is an improved version of the hash function in Paper B, can be used as the underlying hash function in the signature scheme of Paper F.

I have shown how each paper have helped me reach the research goals. In addition, all papers presented in Part II of this thesis, except Paper B, have been accepted to peer reviewed conferences. Paper B was submitted to NIST for the SHA-3 competition. Note that a paper closely related to Paper B was also accepted to a peer reviewed conference [GØM+09a]. In conclusion I have reached the reseach goals of this thesis as they are described in Section 3.

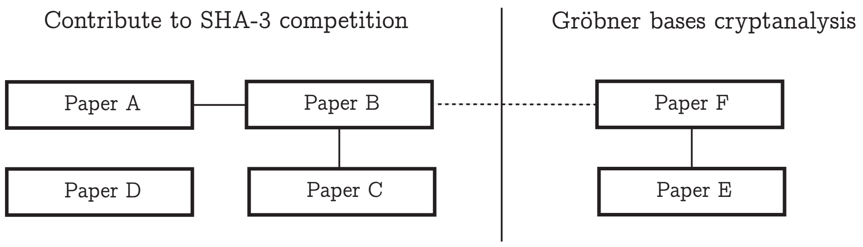

Figure 7. Overview of the papers included in this thesis. A line between two papers indicates a strong connection, while a dashed line indicates a weak connection.

Part II

INCLUDED PAPERS

PAPER A

On Some Properties of Boolean Matrices from Latin Squares

Rune Steinsmo Ødegård, Marija Mihova and Danilo Gligoroski

In Proceedings of the 1st International Workshop on Security and Communication Networks

Trondheim, Norway. May 20-22, 2009

ON SOME PROPERTIES OF BOOLEAN MATRICES FROM LATIN SQUARES

Rune Steinsmo Ødegård
Centre for Quantifiable Quality of Service in Communication Systems.
Norwegian University of Science and Technology
Trondheim, Norway
rune.odegard@q2s.ntnu.no

Marija Mihova
Institute of Informatics
Faculty of Natural Sciences and Mathematics
Ss Cyril and Methodius University in Skopje, Macedonia
marija@ii.edu.mk

Danilo Gligoroski
Department of Telematics
Faculty of Information Technology, Mathematics and Electrical Engineering
Norwegian University of Science and Technology
danilog@item.ntnu.no

Abstract EDON-$\mathcal{R}$ is one of the candidate hash functions for the ongoing NIST competition for the next cryptographic hash standard called SHA-3. Its construction is based on algebraic properties of non-commutative and non-associative quasigroups of orders 2^{256} and 2^{512}. In this paper we present some results from our investigation of the mathematical properties of the quasigroups used in EDON-$\mathcal{R}$.

Keywords: Quasigroups, Boolean matrices, EDON-$\mathcal{R}$, SHA-3

1. Introduction

Recently Gligoroski et.al submitted the hash function EDON-$\mathcal{R}$ [GØM$^+$09b] to the NIST hash competition [NIS07]. The conjectured one-wayness of EDON-$\mathcal{R}$ is based on the problem of solving a system of quasigroup equations in quasigroups of order 2^{256} and 2^{512}. The underlying binary operation of these quasigroups are constructed using a map from Latin squares to Boolean matrices. As a part of our cryptanalysis of the EDON-$\mathcal{R}$ hash function we present here several interesting mathematical properties for these Boolean matrices. 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-$\mathcal{R}$.

This paper is organized as follows. We first give the required mathematical background in Section 2, and then show the connection to EDON-$\mathcal{R}$ in Section 3. In Section 4 we present our results. Then we discuss some open questions in Section 5, before Section 6 concludes the paper.

2. Preliminaries

In this section we give the required mathematical background. We first give a definition of quasigroups. The modern cryptography started actually with a strong connection to quasigroups trough the fundamental work of Shannon [Sha49] almost 60 years ago. Then, many cryptographic primitives were designed by the use of quasigroups. For example an authentication scheme has been proposed by Dénes and Keedwell [DK92], a secret sharing scheme was proposed by Cooper et al. [CDS94], and a version of the block cipher DES defined by Latin squares was proposed by Carter et al. [CDN95]. Moreover, different constructions of hash functions using quasigroups have been proposed by several authors [SV94, GØM+09b]. For a good introduction to quasigroups the reader is adviced to see [Bel67, Smi07] and [MR95].

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

$$\begin{aligned} a * x = b \\ y * a = b \end{aligned} \tag{1}$$

has unique solutions x and y in Q.

Closely related to quasigroups are Latin squares. The reason for this is that the multiplication table of a quasigroup is just a Latin square.

DEFINITION 2 *A* Latin square *of size n is an $n \times n$-matrix whose elements are the numbers $0, \dots, n-1$ and each number appears exactly one time in each row and each column.*

The map used to construct the underlying binary operation of the quasigroups in EDON-$\mathcal{R}$ is the following map from the set of Latin squares to the set of Boolean matrices.

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

Figure 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 start the enumerating of the rows and columns of the matrices from 0.

DEFINITION 3 *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*

$$\begin{array}{rccc} F_k^n: & \mathcal{L}^n & \rightarrow & M(\mathbb{Z}_2) \times M(\mathbb{Z}_2) \\ & L & \mapsto & (\mathbb{A}_1, \mathbb{A}_2) \end{array} \tag{2}$$

where

$$(\mathbb{A}_1)_{ij} = \begin{cases} 1 & \textit{if } j \in \{L_{1i}, \ldots, L_{ki}\} \\ 0 & \textit{else} \end{cases} \tag{3}$$

and

$$(\mathbb{A}_2)_{ij} = \begin{cases} 1 & \textit{if } j \in \{L_{k+1,i}, \ldots, L_{ni}\} \\ 0 & \textit{else} \end{cases} \tag{4}$$

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 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, otherwise it is zero.

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 $F_{n-k}^n(L') = (\mathbb{A}_2, \mathbb{A}_1)$. The Latin square L' is just the square L where the top k-rows and the bottom $(n-k)$-rows has exchanged place. This fact will be used to prove some of the results this paper.

$$L_1 = \begin{bmatrix} 0&7&1&3&2&4&6&5\\ 4&1&7&6&3&0&5&2\\ 7&0&4&2&5&3&1&6\\ 1&4&0&5&6&2&7&3\\ 2&3&6&7&1&5&0&4\\ 5&2&3&1&7&6&4&0\\ 3&6&5&0&4&7&2&1\\ 6&5&2&4&0&1&3&7 \end{bmatrix} \qquad L_2 = \begin{bmatrix} 0&4&2&3&1&6&5&7\\ 7&6&3&2&5&4&1&0\\ 5&3&1&6&0&2&7&4\\ 1&0&5&4&3&7&2&6\\ 2&1&0&7&4&5&6&3\\ 3&5&7&0&6&1&4&2\\ 4&7&6&1&2&0&3&5\\ 6&2&4&5&7&3&0&1 \end{bmatrix}$$

Figure 2. Two mutually orthogonal Latin squares used to define the permutations π_2 and π_3

3. Concrete example

We will now give a short description of the connection to EDON-$\mathcal{R}$. The operation $*$ of the quasigroups $(Q, *)$ used in EDON-$\mathcal{R}$ is defined by three permutations by the following formula:

$$\mathbf{X} * \mathbf{Y} \equiv \pi_1(\pi_2(\mathbf{X}) +_8 \pi_3(\mathbf{Y})) \tag{5}$$

The permutations π_2 and π_3 are defined using the two orthogonal Latin squares of order 8 given in Figure 2. This is done by computing

$$\begin{aligned} F_8^5(L_1) &= F_8^5(\begin{bmatrix} 0&7&1&3&2&4&6&5\\ 4&1&7&6&3&0&5&2\\ 7&0&4&2&5&3&1&6\\ 1&4&0&5&6&2&7&3\\ 2&3&6&7&1&5&0&4\\ 5&2&3&1&7&6&4&0\\ 3&6&5&0&4&7&2&1\\ 6&5&2&4&0&1&3&7 \end{bmatrix}) \\ &= (\begin{bmatrix} 1&1&1&0&1&0&0&1\\ 1&1&0&1&1&0&0&1\\ 1&1&0&0&1&0&1&1\\ 0&0&1&1&0&1&1&1\\ 0&1&1&1&0&1&1&0\\ 1&0&1&1&1&1&0&0\\ 1&1&0&0&0&1&1&1\\ 0&0&1&1&1&1&1&0 \end{bmatrix}, \begin{bmatrix} 0&0&0&1&0&1&1&0\\ 0&0&1&0&0&1&1&0\\ 0&0&1&1&0&1&0&0\\ 1&1&0&0&1&0&0&0\\ 1&0&0&0&1&0&0&1\\ 0&1&0&0&0&0&1&1\\ 0&0&1&1&1&0&0&0\\ 1&1&0&0&0&0&0&1 \end{bmatrix}) \\ &= (\mathbb{A}_1, \mathbb{A}_2) \end{aligned} \tag{6}$$

and

$$\begin{aligned} F_8^5(L_2) &= F_8^5(\begin{bmatrix} 0&4&2&3&1&6&5&7\\ 7&6&3&2&5&4&1&0\\ 5&3&1&6&0&2&7&4\\ 1&0&5&4&3&7&2&6\\ 2&1&0&7&4&5&6&3\\ 3&5&7&0&6&1&4&2\\ 4&7&6&1&2&0&3&5\\ 6&2&4&5&7&3&0&1 \end{bmatrix}) \\ &= (\begin{bmatrix} 1&1&1&0&0&1&0&1\\ 1&1&0&1&1&0&1&0\\ 1&1&1&1&0&1&0&0\\ 0&0&1&1&1&0&1&1\\ 1&1&0&1&1&1&0&0\\ 0&0&1&0&1&1&1&1\\ 0&1&1&0&0&1&1&1\\ 1&0&0&1&1&0&1&1 \end{bmatrix}, \begin{bmatrix} 0&0&0&1&1&0&1&0\\ 0&0&1&0&0&1&0&1\\ 0&0&0&0&1&0&1&1\\ 1&1&0&0&0&1&0&0\\ 0&0&1&0&0&0&1&1\\ 1&1&0&1&0&0&0&0\\ 1&0&0&1&1&0&0&0\\ 0&1&1&0&0&1&0&0 \end{bmatrix}) \\ &= (\mathbb{A}_3, \mathbb{A}_4) \end{aligned} \tag{7}$$

.

The Boolean matrices $\mathbb{A}_1, \mathbb{A}_2$ are then used to define the permutation π_2, while $\mathbb{A}_3, \mathbb{A}_4$ are used to define the permutation π_3. For more detail of the construction of these permutation we refer the reader to [GØM$^+$09b].

As mentioned, from the definition of the map F_n^k, it is obvious that

$$\mathbb{A}_1 + \mathbb{A}_2 = \mathbb{1} \tag{8}$$

and

$$\mathbb{A}_3 + \mathbb{A}_4 = \mathbb{1}. \tag{9}$$

We also find that it is an interesting fact that similar properties hold for their inverse matrices:

$$\mathbb{A}_1^{-1} + \mathbb{A}_2^{-1} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix} = \mathbb{1} \tag{10}$$

and

$$\mathbb{A}_3^{-1} + \mathbb{A}_4^{-1} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix} = \mathbb{1}. \tag{11}$$

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.

4. Results

In this section we show that if both n and $n-k$ is odd, the map from Section 2 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 $F_k^n(L) = (\mathbb{A}_1, \mathbb{A}_2)$. Then det$(\mathbb{A}_1) = 1$ mod 2 if and only if det$(\mathbb{A}_2) = 1$ mod 2.*

Proof Let $\det(\mathbb{A}_1) = 1 \bmod 2$. 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 $\det(\mathbb{A}_2) = 0 \bmod 2$. This means that there exists some $\tilde{v}_l$ and $b_1, \dots b_n \in \{0,1\}$ such that

$$\begin{aligned} \mathbb{1} + v_l &= \tilde{v}_l = \sum_{i \neq l} b_i \tilde{v}_i = \sum_{i \neq l} b_i(\mathbb{1} + v_i) \\ &= \begin{cases} \mathbb{1} + \sum_{i \neq l} b_i v_i & \text{if } \sum b_i = 1 \\ \sum_{i \neq l} b_i v_i & \text{if } \sum b_i = 0 \end{cases} \end{aligned} \tag{12}$$

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 $v_1, \dots, v_n$ 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. The reason for this is that a sum of an even number of vectors with the same weight must have even weight. This means that the vector on right side of the equation has even weight, while the vector on the left side of the equation has odd weight since $n - k$ is odd. Since neither of the two cases in the equation above is possible, the assumption that $\det(\mathbb{A}_2) = 0 \bmod 2$ must be wrong.

This means $\det(\mathbb{A}_1) = 1 \bmod 2 \Rightarrow \det(\mathbb{A}_2) = 1 \bmod 2$. To prove $\det(\mathbb{A}_2) = 1 \bmod 2 \Rightarrow \det(\mathbb{A}_1) = 1 \bmod 2$ notice 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 $F_{k'}^n(L') = (\mathbb{A}_2, \mathbb{A}_1)$. □

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

PROOF Let the $l_{i,1}, \dots l_{i,k}$ 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 $\mathbb{A}_1 \mathbb{A}_1^{-1} = I$ we have the following equations for any column j of $\mathbb{A}_1^{-1}$

$$\begin{aligned} b_{l_{1,1}j} + \cdots + b_{l_{1k}j} &= 0 \\ &\vdots \\ b_{l_{j-1,1}j} + \cdots + b_{l_{j-1,k}j} &= 0 \\ b_{l_{j,1}j} + \cdots + b_{l_{j,k}j} &= 1 \\ b_{l_{j+1,1}j} + \cdots + b_{l_{j+1,k}j} &= 0 \\ &\vdots \\ b_{l_{n,1}j} + \cdots + b_{l_{n,k}j} &= 0 \end{aligned} \tag{13}$$

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 paper.

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*

$$\mathbb{A}_2^{-1} = \mathbb{A}_1^{-1} + \mathbb{1} \tag{14}$$

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

$$\begin{aligned}
\mathbb{A}_1 &= \mathbb{1} + \mathbb{A}_2 & (15)\\
\mathbb{A}_1\mathbb{A}_1^{-1} &= \mathbb{1}\mathbb{A}_1^{-1} + \mathbb{A}_2\mathbb{A}_1^{-1} & (16)\\
I &= \mathbb{1}\mathbb{A}_1^{-1} + \mathbb{A}_2\mathbb{A}_1^{-1} & (17)\\
\mathbb{A}_2^{-1} &= \mathbb{A}_2^{-1}\mathbb{1}\mathbb{A}_1^{-1} + \mathbb{A}_1^{-1} & (18)\\
\mathbb{A}_2^{-1} &= \mathbb{1} + \mathbb{A}_1^{-1} & (19)
\end{aligned}$$

Where equality 15 follows from the definition of Latin squares, and the step from equality 18 to equality 19 follows from Lemma 5. □

5. Open questions

The quasigroups used in EDON-$\mathcal{R}$ are obtained from Latin Squares, which by themselves also can be considered as quasigroups. It would be interesting to extend our analysis to investigate similar object that are obtained by the principles defined in EDON-$\mathcal{R}$, but on algebraic objects such as left or right quasigroups, loops and groups.

6. Conclusion

We have shown that when k and $n - k$ is odd the map from Definition 3 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.

References

[Bel67] V. D. Belousov. Osnovi teorii kvazigrup i lup (in russian), 1967. Nauka, Moscow.

[CDN95] G. Carter, E. Dawson, and L. Nielsen. A latin square variation of DES. In *Proc. Workshop of Selected Areas in Cryptography*, Ottawa, Canada, 1995.

[CDS94] J. Cooper, D. Donovan, and J. Seberry. Secret sharing schemes arising from latin squares. *Bulletin of the Institute of Combinatorics and its Applications*, 12(4):33–43, 1994.

[DK92] J. Dénes and A. D. Keedwell. A new authentication scheme based on latin squares. *Discrete Math.*, 106-107:157–161, 1992.

[GØM$^+$09] Danilo Gligoroski, Rune Steinsmo Ødegård, Marija Mihova, Svein Johan Knapskog, Ljupco Kocarev, Aleš Drápal, and Vlastimil Klima. Cryptographic hash function EDON-$\mathcal{R}$. Submission to NIST, 2009.

[MR95] B.D. McKay and E. Rogoyski. Latin squares of order 10. *Electronic J. Comb.*, 2(3), 1995. `http://ejc.math.gatech.edu:8080/Journal/journalhome.html`.

[NIS07] NIST. Announcing request for candidate algorithm nominations for a new cryptographic hash algorithm (SHA-3) family. Federal Register Notice, 72(112), November 2007. `http://csrc.nist.govgroups/ST/hash/documents/FR_Notice_Nov07.pdf`.

[Sha49] C. E. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28(4):656—715, 1949.

[Smi07] J. D. H. Smith. *An introduction to quasigroups and their representations.* Chapman & Hall/CRC, 2007.

[SV94] C.-P. Schnorr and S. Vaudenay. Black Box Cryptoanalysis of Hash Networks Based on Multipermutations. In *Proceedings of EUROCRYPT 1994*, volume 950 of *LNCS*, pages 47–57. Springer, 1994.

PAPER B

Cryptographic Hash Function Edon-$\mathcal{R}$

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

Submission to NIST 2008

CRYPTOGRAPHIC HASH FUNCTION Edon-$\mathcal{R}$

Danilo Gligoroski
Department of Telematics
Faculty of Information Technology, Mathematics and Electrical Engineering
Norwegian University of Science and Technology
danilog@item.ntnu.no

Rune Steinsmo Ødegård
Centre for Quantifiable Quality of Service in Communication Systems.
Norwegian University of Science and Technology
Trondheim, Norway
rune.odegard@q2s.ntnu.no

Marija Mihova
Institute of Informatics
Faculty of Natural Sciences and Mathematics
Ss Cyril and Methodius University in Skopje, Macedonia
marija@ii.edu.mk

Svein Johan Knapskog
Centre for Quantifiable Quality of Service in Communication Systems.
Norwegian University of Science and Technology
Trondheim, Norway
svein.knapsjog@q2s.ntnu.no

Ljupco Kocarev
University of California
San Diego, USA and
Macedonian Academy of Sciences and Arts
lkocarev@ucsd.edu

Aleš Drápal
Charles University, Czech Republic
drapal@karlin.mff.cuni.cz

Vlastimil Klima
Independent cryptologist - consultant, Czech Republic
v.klima@volny.cz

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 [NIS07].

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 $n = 384, 512$.

Keywords: Hash function, SHA-3, quasigroup string transformation

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

Table 1. Little-endian representation of the 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`

Table 2. Little-endian representation of the 64–bit integer value: "`0xA103FE2332EF301A`.

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 and Table 2 respectivly. 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 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 $M^{(i)} = (M_0^{(i)}, \ldots, M_{15}^{(i)})$ 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 $M^{(i)} = (M_0^{(i)}, \ldots, M_{15}^{(i)})$.
$P^{(i)}$	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 $P^{(i)} = (P_0^{(i)}, \ldots, P_{15}^{(i)})$ 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. $\|Q\| = 2^q$.
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 3. 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 3, 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.

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

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.

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

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, $M^{(1)}$, $M^{(2)}$, …, $M^{(N)}$. 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, $M^{(1)}$, $M^{(2)}$, …, $M^{(N)}$. 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 subsubsections 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 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`.

$P_0^{(0)}$ = 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 4. 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 5. Initial double pipe $P^{(0)}$ for EDON-$\mathcal{R}$256

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 4.

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 5.

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 6.

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 7.

$P_0^{(0)} =$ 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 6. 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 7. Initial double pipe $P^{(0)}$ for EDON-$\mathcal{R}$512

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 [GMK06, Gli09]. 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 $* : Q^2 \to Q$ with the property that each of the equations*

$$\begin{array}{c} a * x = b \\ y * a = b \end{array} \tag{1}$$

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 [Bel67, DK92, Smi07, MR95, MGB99].

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^{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	$T_0 \leftarrow ROTL^0(\texttt{0xAAAAAAAA} + X_0 + X_1 + X_2 + X_4 + X_7);$
	$T_1 \leftarrow ROTL^4(X_0 + X_1 + X_3 + X_4 + X_7);$
	$T_2 \leftarrow ROTL^8(X_0 + X_1 + X_4 + X_6 + X_7);$
	$T_3 \leftarrow ROTL^{13}(X_2 + X_3 + X_5 + X_6 + X_7);$
	$T_4 \leftarrow ROTL^{17}(X_1 + X_2 + X_3 + X_5 + X_6);$
	$T_5 \leftarrow ROTL^{22}(X_0 + X_2 + X_3 + X_4 + X_5);$
	$T_6 \leftarrow ROTL^{24}(X_0 + X_1 + X_5 + X_6 + X_7);$
	$T_7 \leftarrow ROTL^{29}(X_2 + X_3 + X_4 + X_5 + X_6);$
2	$T_8 \leftarrow T_3 \oplus T_5 \oplus T_6;$
	$T_9 \leftarrow T_2 \oplus T_5 \oplus T_6;$
	$T_{10} \leftarrow T_2 \oplus T_3 \oplus T_5;$
	$T_{11} \leftarrow T_0 \oplus T_1 \oplus T_4;$
	$T_{12} \leftarrow T_0 \oplus T_4 \oplus T_7;$
	$T_{13} \leftarrow T_1 \oplus T_6 \oplus T_7;$
	$T_{14} \leftarrow T_2 \oplus T_3 \oplus T_4;$
	$T_{15} \leftarrow T_0 \oplus T_1 \oplus T_7;$
3	$T_0 \leftarrow ROTL^0(\texttt{0x55555555} + Y_0 + Y_1 + Y_2 + Y_5 + Y_7);$
	$T_1 \leftarrow ROTL^5(Y_0 + Y_1 + Y_3 + Y_4 + Y_6);$
	$T_2 \leftarrow ROTL^9(Y_0 + Y_1 + Y_2 + Y_3 + Y_5);$
	$T_3 \leftarrow ROTL^{11}(Y_2 + Y_3 + Y_4 + Y_6 + Y_7);$
	$T_4 \leftarrow ROTL^{15}(Y_0 + Y_1 + Y_3 + Y_4 + Y_5);$
	$T_5 \leftarrow ROTL^{20}(Y_2 + Y_4 + Y_5 + Y_6 + Y_7);$
	$T_6 \leftarrow ROTL^{25}(Y_1 + Y_2 + Y_5 + Y_6 + Y_7);$
	$T_7 \leftarrow ROTL^{27}(Y_0 + Y_3 + Y_4 + Y_6 + Y_7);$
4	$Z_5 \leftarrow T_8 + (T_3 \oplus T_4 \oplus T_6);$
	$Z_6 \leftarrow T_9 + (T_2 \oplus T_5 \oplus T_7);$
	$Z_7 \leftarrow T_{10} + (T_4 \oplus T_6 \oplus T_7);$
	$Z_0 \leftarrow T_{11} + (T_0 \oplus T_1 \oplus T_5);$
	$Z_1 \leftarrow T_{12} + (T_2 \oplus T_6 \oplus T_7);$
	$Z_2 \leftarrow T_{13} + (T_0 \oplus T_1 \oplus T_3);$
	$Z_3 \leftarrow T_{14} + (T_0 \oplus T_3 \oplus T_4);$
	$Z_4 \leftarrow T_{15} + (T_1 \oplus T_2 \oplus T_5);$

Table 8. An algorithmic description of a quasigroup of order 2^{256}.

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

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

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

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

Table 9. 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

$$\Big(\left(\mathbb{Z}_{2^w}\right)^8, +_8\Big),\ w = 32, 64,$$

where $+_8$, is the operation of componentwise addition of two 8–dimensional vectors in $\left(\mathbb{Z}_{2^w}\right)^8$. We shall thus define three permutations $\pi_i : \mathbb{Z}_2^q \to \mathbb{Z}_2^q$, $1 \leq i \leq 3$ 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. $ROTL^{\mathbf{r}}(\mathbf{X}) = \mathbf{D^r} \cdot \mathbf{X}$ where $\mathbf{D^r} \in \mathbb{Z}_2^q \times \mathbb{Z}_2^q$,

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

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:

- $\widehat{\mathbb{A}}_1, \widehat{\mathbb{A}}_3 : \left(\mathbb{Z}_{2^w}\right)^8 \to \left(\mathbb{Z}_{2^w}\right)^8$ 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.

- $\mathbb{A}_2, \mathbb{A}_4 : \left(\mathbb{Z}_{2^w}\right)^8 \to \left(\mathbb{Z}_{2^w}\right)^8$ 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, the matrices

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 10. Rotation vectors for definition of π_2 and π_3.

$\mathbb{A}_2$ 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 $\mathbb{B}_{i,j} \in \mathbb{Z}_2^w \times \mathbb{Z}_2^w,\ 1 \le i,j \le 8$ 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 $\pi_1 : Q_q \to Q_q$ ($q = 256, 512$) 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 5 *Transformations π_1 are permutations.* □

Definition 6 *Transformations $\pi_2 : Q_q \to Q_q$ and $\pi_3 : Q_q \to Q_q$ 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 10, and the information about $\widehat{\mathbb{A}}_1$, $\mathbb{A}_2$, $\widehat{\mathbb{A}}_3$ and $\mathbb{A}_4$ is given in Table 11. 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 7 *Transformations π_2 and π_3 are permutations on $Q_q,\ q = 256, 512$.*

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 $(\mathbb{Z}_2, +, \times)$.

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

Table 11. Matrices $\widehat{\mathbb{A}}_1$, $\mathbb{A}_2$, $\widehat{\mathbb{A}}_3$ and $\mathbb{A}_4$.

THEOREM 8 *Operations $*_q : Q_q^2 \to Q_q$ 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 9

1 $\mathcal{R}_{256} : Q_{256}^4 \rightarrow Q_{256}^2 \equiv \mathcal{R}$ *where* $\mathcal{R}$ *is defined as in Definition 11 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 11 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 [Gli05], and a generic hash function with reverse quasigroup string transformation has been described in [GMK06, Gli09]. A concrete hash function with similar name: Edon-R(n) for $n = 256, 384, 512$ has been described in [GK08]. 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 [GMK08a]. 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 10 *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 11 *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:

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

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 11 can be easily understood by looking at Figure 1a. 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 "=".

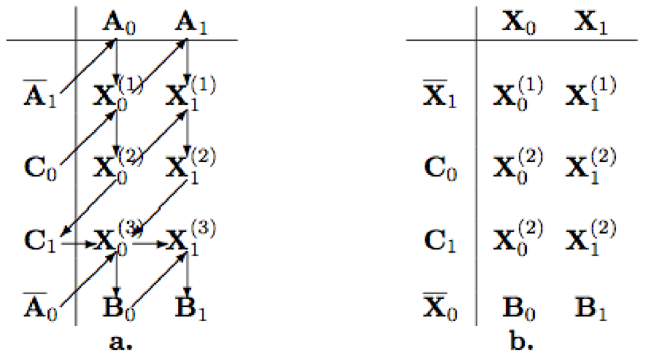

Figure 1. **a.** Schematic representation of the function $\mathcal{R}$, **b.** Conjectured one-wayness of $\mathcal{R}$ comes from the difficulty to solve a system of two equations where $\mathbf{B}_0$, $\mathbf{B}_1$, $\mathbf{C}_0$ and $\mathbf{C}_1$ are given, and $\mathbf{A}_0 = X_0$ and $\mathbf{A}_1 = X_1$ are indeterminate variables.

The conjectured one-wayness of $\mathcal{R}$ can be explained by Figure 1b. 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 11 and obtain the following relations for the elements of Figure 1b:

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

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

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

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 12.

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, $M^{(1)}$, $M^{(2)}$, ..., $M^{(N)}$.
    (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 12. A generic description of the EDON-$\mathcal{R}$ hash algorithm

In the generic description the words of the initial double pipe $P_0^{(i-1)}$, $P_1^{(i-1)}$, ..., $P_{15}^{(i-1)}$ are represented as two vectors of length 8 i.e.

$$(P_0^{(i-1)}, P_1^{(i-1)}, \ldots, P_{15}^{(i-1)}) \equiv (\mathbf{P}_0^{(0)}, \mathbf{P}_1^{(0)}) \equiv P^{(0)}.$$

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

$$M' = (M^{(1)}, M^{(2)}, \ldots, M^{(N)}).$$

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 Figure 2.

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:

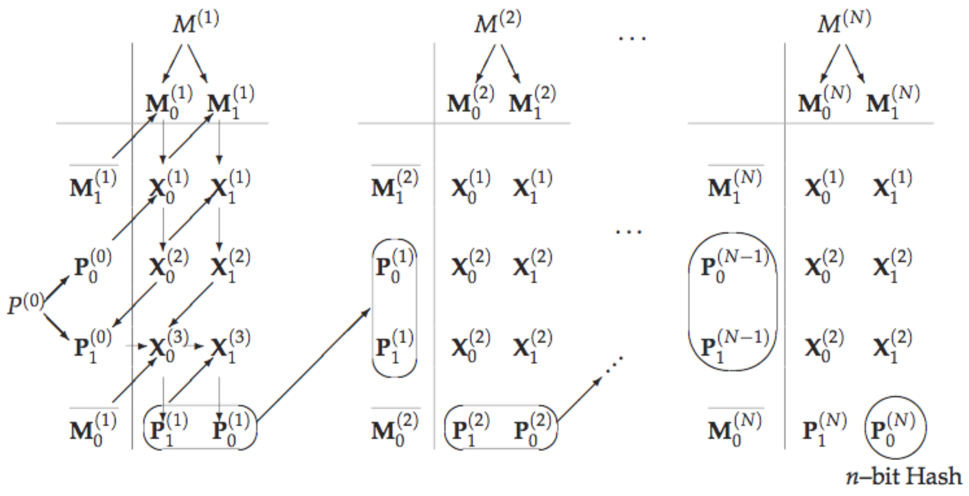

Figure 2. A graphic representation of the EDON-$\mathcal{R}$ hash algorithm.

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 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 4 for BWM224, or as defined in Table 5 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 6 for BWM384, or as defined in Table 7 for BWM512.

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 [Ber08a], The Tiny Encryption Algorithm [WN94], IDEA [LMM91], 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 [GMK06]. 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 $\{0,1\}^{512}$.

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

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

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

Table 13. Two mutually orthogonal Latin squares used to define the permutations π_2 and π_3

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 [CDN95, CDS94, DK92, SV94, Sha49].

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 13.

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 [RMG$^+$00]). 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 10, and the information about $\widehat{\mathbb{A}}_1$, $\mathbb{A}_2$, $\widehat{\mathbb{A}}_3$ and $\mathbb{A}_4$ is given in Table 11.

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 13 ($\mathbb{A}_1$ 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 13 ($\mathbb{A}_3$ 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 12 *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., $|R_{\mathbb{A}_i,j}| = 5,\ i = 1,3,\ \forall j \in \{0,\dots,7\}$.

DEFINITION 13 *For the Latin rectangles* $L_{1,2} = (l^{(1)}_{i,j}),\ L_{2,2} = (l^{(2)}_{i,j})\ i = 0,1,2,\ j = 0,1,\dots 7$, *we denote the sets of elements of their* j*-th column as* $L^{(j)}_{1,2} = \{l^{(1)}_{0,j}, l^{(1)}_{1,j}, l^{(1)}_{2,j}\}$ *and as* $L^{(j)}_{2,2} = \{l^{(2)}_{0,j}, l^{(2)}_{1,j}, l^{(2)}_{2,j}\}$.

DEFINITION 14 *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*

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

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*

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

Diffusion matrices for π_2 and π_3 are given in Table 14.

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

Table 14. 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 15 $\mathbf{Diff}_{\pi_2} = \Big(\mathbb{A}_1 \cdot \mathbb{A}_2\Big)^T$ *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 11 and "T" is a transposition of a matrix.* $\square$

In Table 15 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

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

Table 15. Eigenvalues and the eigenvectors for the diffusion matrices $\mathbf{Diff}_{\pi_2}$ and $\mathbf{Diff}_{\pi_3}$.

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 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 16 *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 10 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 17 *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 17 for proving the following properties of the differential characteristics of the quasigroup operation $*_q$.

Corollary 18 *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 16, 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 17.

COROLLARY 19 *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 20 *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 $\mathcal{D}_{(\mathbf{\Delta_X}, \mathbf{\Delta_Y})} = (\delta_0, \ldots, \delta_7) \in (\mathbb{Z})^8$ 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 21

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

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 16, Corollary 18 and Corollary 19.

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 16a). 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 16 and in Table 17.

In Table 18 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 19

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

a.

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

b.

Table 16. **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 $\mathbf{D}_i,\ i = 1, \dots, 8$. **b.** Counting the minimal number of bit flips $\mathcal{D}_i,\ i = 1, \dots, 8$ when applying the quasigroup operation $*_q$ on $\mathbf{X}$ and $\mathbf{Y}$ that differ by difference vectors $\mathbf{\Delta_X}$ and $\mathbf{\Delta_Y}$. Corresponding difference vectors for the fixed values $\mathbf{C}_0$ and $\mathbf{C}_1$ are the zero vector $\mathbf{0}$.

$$\mathcal{D}_1 = \mathbf{Diff}_{\pi_2} \cdot \overline{\mathbf{\Delta_Y}} + \mathbf{Diff}_{\pi_3} \cdot \mathbf{\Delta_X}$$
$$\mathcal{D}_2 = \mathbf{Diff}_{\pi_2} \cdot (\mathbf{Diff}_{\pi_2} \cdot \overline{\mathbf{\Delta_Y}} + \mathbf{Diff}_{\pi_3} \cdot \mathbf{\Delta_X}) + \mathbf{Diff}_{\pi_3} \cdot \mathbf{\Delta_Y}$$
$$\mathcal{D}_3 = \mathbf{Diff}_{\pi_3} \cdot (\mathbf{Diff}_{\pi_2} \cdot \overline{\mathbf{\Delta_Y}} + \mathbf{Diff}_{\pi_3} \cdot \mathbf{\Delta_X})$$
$$\mathcal{D}_4 = \mathbf{Diff}_{\pi_2} \cdot \Big(\mathbf{Diff}_{\pi_3} \cdot (\mathbf{Diff}_{\pi_2} \cdot \overline{\mathbf{\Delta_Y}} + \mathbf{Diff}_{\pi_3} \cdot \mathbf{\Delta_X})\Big) + \mathbf{Diff}_{\pi_3} \cdot \Big(\mathbf{Diff}_{\pi_2} \cdot (\mathbf{Diff}_{\pi_2} \cdot \overline{\mathbf{\Delta_Y}} + \mathbf{Diff}_{\pi_3} \cdot \mathbf{\Delta_X}) + \mathbf{Diff}_{\pi_3} \cdot \mathbf{\Delta_Y}\Big)$$
$$\mathcal{D}_5 = \mathbf{Diff}_{\pi_2} \cdot \Big(\mathbf{Diff}_{\pi_3} \cdot (\mathbf{Diff}_{\pi_2} \cdot \overline{\mathbf{\Delta_Y}} + \mathbf{Diff}_{\pi_3} \cdot \mathbf{\Delta_X})\Big)$$
$$\mathcal{D}_6 = \mathbf{Diff}_{\pi_2} \cdot \bigg(\mathbf{Diff}_{\pi_2} \cdot \Big(\mathbf{Diff}_{\pi_3} \cdot (\mathbf{Diff}_{\pi_2} \cdot \overline{\mathbf{\Delta_Y}} + \mathbf{Diff}_{\pi_3} \cdot \mathbf{\Delta_X})\Big) + \mathbf{Diff}_{\pi_3} \cdot \Big(\mathbf{Diff}_{\pi_2} \cdot (\mathbf{Diff}_{\pi_2} \cdot \overline{\mathbf{\Delta_Y}} + \mathbf{Diff}_{\pi_3} \cdot \mathbf{\Delta_X}) + \mathbf{Diff}_{\pi_3} \cdot \mathbf{\Delta_Y}\Big)\bigg) + \mathbf{Diff}_{\pi_3} \cdot \bigg(\mathbf{Diff}_{\pi_2} \cdot \Big(\mathbf{Diff}_{\pi_3} \cdot (\mathbf{Diff}_{\pi_2} \cdot \overline{\mathbf{\Delta_Y}} + \mathbf{Diff}_{\pi_3} \cdot \mathbf{\Delta_X})\Big)\bigg)$$
$$\mathcal{D}_7 = \mathbf{Diff}_{\pi_2} \cdot \overline{\mathbf{\Delta_X}} + \mathbf{Diff}_{\pi_3} \cdot \bigg(\mathbf{Diff}_{\pi_2} \cdot \Big(\mathbf{Diff}_{\pi_3} \cdot (\mathbf{Diff}_{\pi_2} \cdot \overline{\mathbf{\Delta_Y}} + \mathbf{Diff}_{\pi_3} \cdot \mathbf{\Delta_X})\Big)\bigg)$$
$$\mathcal{D}_8 = \mathbf{Diff}_{\pi_2} \cdot \Bigg(\mathbf{Diff}_{\pi_2} \cdot \overline{\mathbf{\Delta_X}} + \mathbf{Diff}_{\pi_3} \cdot \bigg(\mathbf{Diff}_{\pi_2} \cdot \Big(\mathbf{Diff}_{\pi_3} \cdot (\mathbf{Diff}_{\pi_2} \cdot \overline{\mathbf{\Delta_Y}} + \mathbf{Diff}_{\pi_3} \cdot \mathbf{\Delta_X})\Big)\bigg)\Bigg) + \mathbf{Diff}_{\pi_3} \cdot \Bigg(\mathbf{Diff}_{\pi_2} \cdot \bigg(\mathbf{Diff}_{\pi_2} \cdot \Big(\mathbf{Diff}_{\pi_3} \cdot (\mathbf{Diff}_{\pi_2} \cdot \overline{\mathbf{\Delta_Y}} + \mathbf{Diff}_{\pi_3} \cdot \mathbf{\Delta_X})\Big) + \mathbf{Diff}_{\pi_3} \cdot \Big(\mathbf{Diff}_{\pi_2} \cdot (\mathbf{Diff}_{\pi_2} \cdot \overline{\mathbf{\Delta_Y}} + \mathbf{Diff}_{\pi_3} \cdot \mathbf{\Delta_X}) + \mathbf{Diff}_{\pi_3} \cdot \mathbf{\Delta_Y}\Big)\bigg) + \mathbf{Diff}_{\pi_3} \cdot \bigg(\mathbf{Diff}_{\pi_2} \cdot \Big(\mathbf{Diff}_{\pi_3} \cdot (\mathbf{Diff}_{\pi_2} \cdot \overline{\mathbf{\Delta_Y}} + \mathbf{Diff}_{\pi_3} \cdot \mathbf{\Delta_X})\Big)\bigg)\Bigg)$$

Table 17. The relations between the vectors of minimal number of bit flips for the function $\mathcal{R}$ when the initial difference vectors are $\mathbf{\Delta_X}$ and $\mathbf{\Delta_Y}$.

	$\mathbf{\Delta_X} = (1,0,0,0,0,0,0,0)$	$\mathbf{\Delta_Y} = (0,0,0,0,0,0,0,0)$
$\overline{\mathbf{\Delta_Y}}$	(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)
$\overline{\mathbf{\Delta_X}}$	(6330, 6331, 6330, 6330, 6332, 6328, 6329, 6330)	(379716, 379715, 379721, 379713, 379716, 379717, 379715, 379712)

Table 18. 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 $\mathbf{\Delta_Y} = (0,0,0,0,0,0,0,0)$.

	$\mathbf{\Delta_X} = (0,0,0,0,0,0,0,0)$	$\mathbf{\Delta_Y} = (1,0,0,0,0,0,0,0)$
$\overline{\mathbf{\Delta_Y}}$	(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)
$\overline{\mathbf{\Delta_X}}$	(6328, 6328, 6330, 6328, 6329, 6328, 6327, 6327)	(386016, 386011, 386017, 386016, 386016, 386018, 386017, 386014)

Table 19. 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 $\mathbf{\Delta_Y} = (1,0,0,0,0,0,0,0)$.

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 $\mathcal{D}_i,\ i = 1,\dots,8$ in Table 18 and Table 19. 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.

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

Let $\lambda_1, \lambda_2, \dots, \lambda_n$ be the eigenvalues of some matrix $\mathbb{M}$, arranged such that $|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_n|$, where $|x + iy| = \sqrt{x^2 + y^2}$. The eigenvector corresponding to the eigenvalue λ_i is denoted by $\mathbf{s}_i = [s_{1i}, s_{2i}, \dots, s_{ni}]^T$. 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. $\mathbb{S} = [\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_n]$. Let the i-th vector-row of the matrix $\mathbb{S}^{-1}$ be denoted by $\mathbf{s}'_i = [s'_{1i}, s'_{2i}, \dots, s'_{ni}]^T$, such that $(\mathbb{S}^{-1})^T = [\mathbf{s}'_1, \mathbf{s}'_2, \dots, \mathbf{s}'_n]$. 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,

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

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.

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

PROPOSITION 22 *Let $|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_n|$ be the eigenvalues of $\mathbb{M}$. If the vector $\mathbb{1} = (1,1,\dots,1)$ is the eigenvector for the greatest eigenvalue, λ_1. Then for each vector $\mathbf{a}$*

$$\lim_{n\to\infty} \frac{(\mathbb{M}^n\mathbf{a})_i}{\lambda_1^n} = (\mathbf{s}'_1)^T\mathbf{a},\ \forall i = 1,\dots,n, \tag{3}$$

PROOF From (2) we have,

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

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

Directly from this holds (3).

PROPOSITION 23 *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* $\mathbf{a}$

$$\lim_{n\to\infty} \frac{Var(\mathbb{M}^n\mathbf{a})}{\min_i(\mathbb{M}^n\mathbf{a})_i} = 0. \tag{4}$$

PROOF Let $b_j^{(n)} = (\mathbb{M}^n\mathbf{a})_j$. Then from (2) 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

$$Avr(b^{(n)}) = \frac{1}{n}\sum_{j=1}^{n} b_j^{(n)} = \lambda_1^n\mathbf{s}'_1\mathbf{a} + \frac{1}{n}\sum_{j=1}^{n}\sum_{i=2}^{n} \lambda_i^n s_{ji}\mathbf{s}'_i\mathbf{a}$$

$$= \lambda_1^n\mathbf{s}'_1\mathbf{a} + \sum_{i=2}^{n} \lambda_i^n \left(\frac{1}{n}\sum_{j=1} s_{ji}\right)\mathbf{s}'_i\mathbf{a}$$

$$= \lambda_1^n\mathbf{s}'_1\mathbf{a} + \sum_{i=2}^{n} \lambda_i^n Avr(s_i)\mathbf{s}'_i\mathbf{a}.$$

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 Proposition 22 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 24 *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 15.

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

Theorem 25 *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 26 *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:

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

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

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 $\Delta_{D_\mu}^{(i)}$ 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 $\big(1-\frac{1}{w}\big)$. Then if we have $\delta_\mu^{(i)}$ experiments, the probability that the number of bit flips for that particular bit is even can be computed as:

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

Similarly we can compute the probability:

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

Now if take into account that

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

and

$$\lim_{\delta_\mu^{(i)}\to\infty} |Pr\big(\Delta_j^{(i)} = 0\big) - Pr\big(\Delta_j^{(i)} = 1\big)| = 0$$

then we can rewrite the last limit as:

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

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

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

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

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

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

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

Table 20. 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 20. 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 26 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 21.

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 [McK]. 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 $\mathbb{A}_i$

For the readability of the thesis we have removed this section which had a lot of similarities with Paper A.

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 [Dam87, Dam89, Mer90a], 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 [Luc04, Luc05] and Coron et al. [CDMP05] 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 [Jou04] and Kelsey and Schneier [KS05], 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 [Dam89, Mer90a] works as follows:

Let $M = M_1||M_2||\ldots||M_N$ 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

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

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 21. Criteria for choosing the Latin squares

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 [Luc04]).

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 22. 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 23. 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.

$Min = 15$ $Avr = 28.86$ $Max = 50$	$Min = 92$ $Avr = 117.66$ $Max = 139$
$Min = 96$ $Avr = 120.06$ $Max = 143$	$Min = 99$ **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 22. **a.** Avalanche propagation of the Hamming distance between two 256–bit words M_1 and M_2 that initially differs in one bit and where $M_1 = 0$ (minimum, average and maximum) **b.** Avalanche propagation of the Hamming distance between two 256–bit words M_1 and M_2 that initially differs in one bit (minimum, average and maximum)

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

a.

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

b.

Table 23. **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)

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.

$*_q$	$B_1 = \{\mathbf{B_1}\}$	$B_2 = \{\mathbf{B_1}, \mathbf{B_2}\}$
$A_1 = \{\mathbf{A_1}\}$	$C_1 = \{\mathbf{C_1}\}$ 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 24. Definition of quasigroup operation between one or two-element sets.

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 27 *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 24.*

Following directly from the properties of the unique solutions of equations of type (1) it is easy to prove the following two propositions:

PROPOSITION 28 *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* $\mathbf{C_1} \neq \mathbf{C_2}$. $\square$

PROPOSITION 29 *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* $\mathbf{C_1} \neq \mathbf{C_2}$. $\square$

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 30 *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* $\{\mathbf{C_1}\}$. $\square$

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

Definition 31 *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 27 and Definition 31 we can trace all possible paths that can produce collisions in the compression function $\mathcal{R}$. That is formulated in the following theorem:

Theorem 32 *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 25 and Table 26.* $\square$

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

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 25. 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 27, Table 28, Table 29 and Table 30.

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

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 26. 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 25. For those collisions, we find that solving corresponding quasigroup equations is hard. For example, let us discuss the case described in Table 25a., with the corresponding equations given in Table 27a. 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 25 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 26) 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 [JP07]) 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}{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}}_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.
\qquad
\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}}_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.
\qquad
\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}}_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.
$$

a. b. c.

Table 27. Concrete systems (a. – c.) of quasigroup equations that can give collisions in the compression function $\mathcal{R}$

$$\left\{\begin{array}{ll}
\mathbf{D}_{14} = & \mathbf{D}_{13} * \mathbf{D}_{11} \\
\mathbf{D}_{14} = & \mathbf{D}_{12} * \mathbf{D}_{10} \\
\mathbf{D}_{13} = & \overline{\mathbf{X}}_{2} * \mathbf{D}_{9} \\
\mathbf{D}_{12} = & \overline{\mathbf{X}}_{1} * \mathbf{D}_{8} \\
\mathbf{D}_{11} = & \mathbf{D}_{7} * \mathbf{D}_{9} \\
\mathbf{D}_{10} = & \mathbf{D}_{7} * \mathbf{D}_{8} \\
\mathbf{D}_{9} = & \mathbf{D}_{6} * \mathbf{C}_{1} \\
\mathbf{D}_{8} = & \mathbf{D}_{5} * \mathbf{C}_{1} \\
\mathbf{D}_{7} = & \mathbf{D}_{6} * \mathbf{D}_{4} \\
\mathbf{D}_{7} = & \mathbf{D}_{5} * \mathbf{D}_{3} \\
\mathbf{D}_{6} = & \mathbf{C}_{0} * \mathbf{D}_{2} \\
\mathbf{D}_{5} = & \mathbf{C}_{0} * \mathbf{D}_{1} \\
\mathbf{D}_{4} = & \mathbf{D}_{2} * \mathbf{X}_{3} \\
\mathbf{D}_{3} = & \mathbf{D}_{1} * \mathbf{X}_{3} \\
\mathbf{D}_{2} = & \overline{\mathbf{X}}_{3} * \mathbf{X}_{2} \\
\mathbf{D}_{1} = & \overline{\mathbf{X}}_{3} * \mathbf{X}_{1}
\end{array}\right.$$

d.

$$\left\{\begin{array}{ll}
\mathbf{D}_{13} = & \mathbf{D}_{12} * \mathbf{D}_{11} \\
\mathbf{D}_{12} = & \overline{\mathbf{X}}_{2} * \mathbf{D}_{10} \\
\mathbf{D}_{12} = & \overline{\mathbf{X}}_{1} * \mathbf{D}_{9} \\
\mathbf{D}_{11} = & \mathbf{D}_{8} * \mathbf{D}_{10} \\
\mathbf{D}_{11} = & \mathbf{D}_{7} * \mathbf{D}_{9} \\
\mathbf{D}_{10} = & \mathbf{D}_{6} * \mathbf{C}_{1} \\
\mathbf{D}_{9} = & \mathbf{D}_{5} * \mathbf{C}_{1} \\
\mathbf{D}_{8} = & \mathbf{D}_{6} * \mathbf{D}_{4} \\
\mathbf{D}_{7} = & \mathbf{D}_{5} * \mathbf{D}_{3} \\
\mathbf{D}_{6} = & \mathbf{C}_{0} * \mathbf{D}_{2} \\
\mathbf{D}_{5} = & \mathbf{C}_{0} * \mathbf{D}_{1} \\
\mathbf{D}_{4} = & \mathbf{D}_{2} * \mathbf{X}_{3} \\
\mathbf{D}_{3} = & \mathbf{D}_{1} * \mathbf{X}_{2} \\
\mathbf{D}_{2} = & \overline{\mathbf{X}}_{3} * \mathbf{X}_{2} \\
\mathbf{D}_{1} = & \overline{\mathbf{X}}_{3} * \mathbf{X}_{1}
\end{array}\right.$$

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.

Table 28. Concrete systems (d. – f.) of quasigroup equations that can give collisions in the compression function $\mathcal{R}$

$$\left\{\begin{array}{ll}
\mathbf{D}_{12} = & \mathbf{D}_{11} * \mathbf{D}_{9} \\
\mathbf{D}_{12} = & \mathbf{D}_{10} * \mathbf{D}_{8} \\
\mathbf{D}_{11} = & \overline{\mathbf{X}}_{2} * \mathbf{D}_{7} \\
\mathbf{D}_{10} = & \overline{\mathbf{X}}_{1} * \mathbf{D}_{7} \\
\mathbf{D}_{9} = & \mathbf{D}_{6} * \mathbf{D}_{7} \\
\mathbf{D}_{8} = & \mathbf{D}_{5} * \mathbf{D}_{7} \\
\mathbf{D}_{7} = & \mathbf{D}_{4} * \mathbf{C}_{1} \\
\mathbf{D}_{6} = & \mathbf{D}_{4} * \mathbf{D}_{3} \\
\mathbf{D}_{5} = & \mathbf{D}_{4} * \mathbf{D}_{2} \\
\mathbf{D}_{4} = & \mathbf{C}_{0} * \mathbf{D}_{1} \\
\mathbf{D}_{3} = & \mathbf{D}_{1} * \mathbf{X}_{4} \\
\mathbf{D}_{2} = & \mathbf{D}_{1} * \mathbf{X}_{3} \\
\mathbf{D}_{1} = & \overline{\mathbf{X}}_{4} * \mathbf{X}_{2} \\
\mathbf{D}_{1} = & \overline{\mathbf{X}}_{3} * \mathbf{X}_{1}
\end{array}\right.$$

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.

$$\left\{\begin{array}{ll}
\mathbf{D}_{14} = & \mathbf{D}_{13} * \mathbf{D}_{11} \\
\mathbf{D}_{14} = & \mathbf{D}_{12} * \mathbf{D}_{10} \\
\mathbf{D}_{13} = & \overline{\mathbf{X}}_{2} * \mathbf{D}_{9} \\
\mathbf{D}_{12} = & \overline{\mathbf{X}}_{1} * \mathbf{D}_{8} \\
\mathbf{D}_{11} = & \mathbf{D}_{7} * \mathbf{D}_{9} \\
\mathbf{D}_{10} = & \mathbf{D}_{6} * \mathbf{D}_{8} \\
\mathbf{D}_{9} = & \mathbf{D}_{5} * \mathbf{C}_{1} \\
\mathbf{D}_{8} = & \mathbf{D}_{4} * \mathbf{C}_{1} \\
\mathbf{D}_{7} = & \mathbf{D}_{5} * \mathbf{D}_{3} \\
\mathbf{D}_{6} = & \mathbf{D}_{4} * \mathbf{D}_{3} \\
\mathbf{D}_{5} = & \mathbf{C}_{0} * \mathbf{D}_{2} \\
\mathbf{D}_{4} = & \mathbf{C}_{0} * \mathbf{D}_{1} \\
\mathbf{D}_{3} = & \mathbf{D}_{2} * \mathbf{X}_{4} \\
\mathbf{D}_{3} = & \mathbf{D}_{1} * \mathbf{X}_{3} \\
\mathbf{D}_{2} = & \overline{\mathbf{X}}_{4} * \mathbf{X}_{2} \\
\mathbf{D}_{1} = & \overline{\mathbf{X}}_{3} * \mathbf{X}_{1}
\end{array}\right.$$

i.

Table 29. Concrete systems (g. – i.) of quasigroup equations that can give collisions in the compression function $\mathcal{R}$

$$\underset{\text{j.}}{\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.}
\qquad
\underset{\text{k.}}{\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.}
\qquad
\underset{\text{l.}}{\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.}$$

Table 30. Concrete systems (j. – 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 [Tho07].

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 [Ber92]. In 2001 Lipmaa and Moriai have constructed efficient algorithms for computing differential properties of addition modulo 2^w (for general values of w) [LM02], and in 2004 Lipmaa, Wallén and Dumas have constructed a linear-time algorithm for computing the additive differential probability of exclusive-or [LWD04].

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 6. Additionally, bearing in mind that $\mathcal{R} : \{0,1\}^{32w} \to \{0,1\}^{16w}$, in this moment we do not see how the results in [LWD04] 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 [KNW08].

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

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

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 31. Situations **a.** and **b.** give a free start collision.

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 33 *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-$\mathcal{R}$.

The free start collision situation is described in the Table 31. 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 and second preimages which is summarized in Table 32. 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 32 when n is replaced by m. In addition we formally state that all 4 algorithms in Table 32 are resistante to length-extension attacks, resistante to multicollision attacks and the provable resistante to differential cryptanalysis.

Algorithm	Digest size n (in bits)	Work factor for finding collision	Work factor for finding a preimage	Word factor for finding a 2nd preimage of a message $\leq 2^k$ bits
Edon-$\mathcal{R}$224	224	$\approx 2^{112}$	$\approx 2^{224}$	$\approx 2^{224-k}$
Edon-$\mathcal{R}$256	256	$\approx 2^{128}$	$\approx 2^{256}$	$\approx 2^{256-k}$
Edon-$\mathcal{R}$384	384	$\approx 2^{192}$	$\approx 2^{384}$	$\approx 2^{384-k}$
Edon-$\mathcal{R}$512	512	$\approx 2^{256}$	$\approx 2^{512}$	$\approx 2^{512-k}$

Table 32. 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 [HKC97, Ass00, NIS08b].

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 [HK06, NIS08a].

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-$\mathcal{R}$.

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 [Gla].

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

In the Table 33 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.

In the Table 34 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 35 we are giving the speed of all four instances of EDON-$\mathcal{R}$ for the optimized 64–bit version.

In the Table 36 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	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 33. 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 34. 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.

	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 35. 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 36. 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 37 we are giving the length of the produced executable code and the speed in number of cycles per byte.

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

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

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 38 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 38. 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 Figure 1 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.

References

[Ass00] American Bankers Association. *Keyed Hash Message Authentication Code.* ANSI X9.71, Washington, D.C., 2000.

[Bel67] V. D. Belousov. Osnovi teorii kvazigrup i lup (in russian), 1967. Nauka, Moscow.

[Ber92] T. A. Berson. Differential Cryptanalysis Mod 2^{32} with Applications to MD5. In *Advances in Cryptology - EUROCRYPT '92*, volume 658 of *LNCS*, pages 71–80, 1992.

[Ber08] 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.

[CDMP05] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård Revisited: How to Construct a Hash Function. In *Advances in Cryptology: CRYPTO'05*, volume 3621 of *LNCS*, pages 430–448. Springer Berlin / Heidelberg, 2005.

[CDN95] G. Carter, E. Dawson, and L. Nielsen. A latin square variation of DES. In *Proc. Workshop of Selected Areas in Cryptography*, Ottawa, Canada, 1995.

[CDS94] J. Cooper, D. Donovan, and J. Seberry. Secret sharing schemes arising from latin squares. *Bulletin of the Institute of Combinatorics and its Applications*, 12(4):33–43, 1994.

[Dam87] I. B. Damgård. Collision free hash functions and public key signature schemes. In *Advances in Cryptology–EUROCRYPT '87*, volume 304 of *LNCS*, pages 203–216. Springer-Verlag, 1987.

[Dam89] Ivan Damgård. A design principle for hash functions. In Gilles Brassard, editor, *CRYPTO*, volume 435 of *LNCS*, pages 416–427. Springer, 1989.

[DK92] J. Dénes and A. D. Keedwell. A new authentication scheme based on latin squares. *Discrete Math.*, 106-107:157–161, 1992.

[GK08] 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.

[Gla] B. Gladman. SHA1, SHA2, HMAC and Key Derivation in C. url:`http://fp.gladman.plus.com/cryptography_technology/sha/index.htm`.

[Gli05] D. Gligoroski. Candidate one-way functions and one-way permutations based on quasigroup string transformations. Cryptology ePrint Archive, Report 2005/352, 2005.

[Gli09] D. Gligoroski. On a family of minimal candidate one-way functions and one-way permutations. *International Journal of Network Security*, in print 2009.

[GMK06] D. Gligoroski, S. Markovski, and L. Kocarev. Edon–$\mathcal{R}$, an infinite family of cryptographic hash functions. In *Second NIST Cryptographic Hash Workshop*, August 2006.

[GMK08] 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.

[HK06] S. Halevi and H. Krawczyk. Strengthening digital signatures via randomized hashing. In *Proceedings of CRYPTO 2006*, volume 4117 of *LNCS*, pages 41–59, 2006.

[HKC97] M. Bellare H. Krawczyk and R. Canetti. *RFC2104 - HMAC: Keyed-Hashing for Message Authentication.* Internet Engineering Task Force, 1997. url:`http://www.faqs.org/rfcs/rfc2104.html`.

[Jou04] Antoine Joux. Multicollisions in iterated hash functions. application to cascaded constructions. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA*, volume 3152 of *LNCS*, pages 306–316. Springer, August 2004.

[JP07] 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.

[KNW08] Dmitry Khovratovich, Ivica Nikolić, and Ralf-Philipp Weinmann. Cryptanalysis of EDON-$\mathcal{R}$. Available online, 2008. url:`http://ehash.iaik.tugraz.at/uploads/7/74/Edon.pdf`.

[KS05] John Kelsey and Bruce Schneier. Second preimages on n-bit hash functions for much less than 2^n work. In *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 474–490. Springer, 2005.

[LM02] H. Lipmaa and S. Moriai. Efficient algorithms for computing differential properties of addition. In *Proceedings of FSE 2001*, volume 2355 of *LNCS*, pages 336–350. Springer-Verlag, 2002.

[LMM91] X. Lai, J. L. Massey, and S. Murphy. Markov ciphers and differential cryptanalysis. In *Advances in Cryptology, CRYPTO '91*, volume 576 of *LNCS*, pages 17–38. Springer-Verlag, 1991.

[Luc04] S. Lucks. Design principles for iterated hash functions. Cryptology ePrint Archive, Report 2004/253, 2004.

[Luc05] Stefan Lucks. A failure-friendly design principle for hash functions. In *Proceeding of ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 474–494. Springer-Verlag, 2005.

[LWD04] 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.

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

[Mer90] R. C. Merkle. One way hash functions and DES, 1990. Based on an unpublished paper from 1979 and his PhD thesis, Stanford, 1979.

[MGB99] 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.

[MR95] B.D. McKay and E. Rogoyski. Latin squares of order 10. *Electronic J. Comb.*, 2(3), 1995. `http://ejc.math.gatech.edu:8080/Journal/journalhome.html`.

[NIS07] NIST. Announcing request for candidate algorithm nominations for a new cryptographic hash algorithm (SHA-3) family. Federal Register Notice, 72(112), November 2007. `http://csrc.nist.govgroups/ST/hash/documents/FR_Notice_Nov07.pdf`.

[NIS08a] NIST. *Randomized Hashing for Digital Signatures - Draft NIST Special Publication 800-106.* Federal Information Processing Standards Publication, August, 2008. `http://csrc.nist.gov/publications/drafts/800-106/2nd-Draft_SP800-106_July2008.pdf`.

[NIS08b] NIST. *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`.

[RMG$^+$00] 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.

[Sha49] C. E. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28(4):656—715, 1949.

[Smi07] J. D. H. Smith. *An introduction to quasigroups and their representations.* Chapman & Hall/CRC, 2007.

[SV94] C.-P. Schnorr and S. Vaudenay. Black Box Cryptoanalysis of Hash Networks Based on Multipermutations. In *Proceedings of EUROCRYPT 1994*, volume 950 of *LNCS*, pages 47–57. Springer, 1994.

[Tho07] S. S. Thomsen, May 2007. Personal communication with S. S. Thomsen.

[WN94] 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.

PAPER C

On the Randomness and Regularity of Reduced Edon-R Compression Function

Rune Steinsmo Ødegård and Danilo Gligoroski

In Proceedings of the 2009 International Conference on Security & Management

Las Vegas, Nevada, USA. July 13-16, 2009.

ON THE RANDOMNESS AND REGULARITY OF REDUCED EDON-$\mathcal{R}$ COMPRESSION FUNCTION

Rune Steinsmo Ødegård
Centre for Quantifiable Quality of Service in Communication Systems.
Norwegian University of Science and Technology
Trondheim, Norway
rune.odegard@q2s.ntnu.no

Danilo Gligoroski
Department of Telematics
Faculty of Information Technology, Mathematics and Electrical Engineering
Norwegian University of Science and Technology
danilog@item.ntnu.no

Abstract EDON-$\mathcal{R}$ is one of the candidate hash functions for the ongoing NIST competition for the next cryptographic hash standard called SHA-3. Its construction is based on algebraic properties of non-commutative and non-associative quasigroups of orders 2^{256} and 2^{512}. In this paper we are giving some of our results in investigation of the randomness and regularity of reduced EDON-$\mathcal{R}$ compression functions over quasigroups of order 2^8 and 2^{16}. Our experiments show that the Bellare-Khono balance of EDON-$\mathcal{R}$ compression function is high. Actually, for the reduced Edon-R with quasigroups of order 2^8 we show that the compression function is perfectly balanced, while with quasigroups of order 2^{16} the Belare-Khono balance is $\mu(R_{16}) = 0.99985$.

Keywords: hash function, randomness, regularity, balance, SHA-3, EDON-$\mathcal{R}$

1. Introduction

Recently Gligoroski et.al submitted the hash function EDON-$\mathcal{R}$ [GØM+09b] to the NIST hash competition [NIS07]. With speeds of 5.77 cycles/byte and 3.15 cycles/byte on amd64 1401MHz Intel Core 2 Duo for EDON-$\mathcal{R}$256 and EDON-$\mathcal{R}$512 respectively, EDON-$\mathcal{R}$ is the fastest hash function submitted to the competition [Be11b]. This has generated a lot of attention in the cryptographic community and much effort has therefore been put into breaking EDON-$\mathcal{R}$. So far there have been various limited results. Klima [Kli08] showed the possibility of 2^K multicollisions requiring $K*2^{n/2}$ computations and access to $2^{n/2}$ units of memory. Khovratovich et.al [KNW08] noted the possibility of free-start collisions and used this to launch a preimage attack on EDON-$\mathcal{R}$ requiring $2^{2n/3}$ computations and access to $2^{2n/3}$ units of memory. Later Gligoroski and Ødegård [GØ09] disputed the validity of the model in which the attack of Khovratovich et. al is compared to generic attacks. The latest result on EDON-$\mathcal{R}$ by Leurent [Leu09] showed the possibility of key recovery using $2^{5n/8}$ operations when EDON-$\mathcal{R}$ is used as a special secret prefix MAC. Note that all identified weaknesses of EDON-$\mathcal{R}$ are only present in the free start collision case. In general, the problem of free start collisions can be addressed for instance by the Davies-Meier method for feed-forwarding of the chaining value.

The design of EDON-$\mathcal{R}$ is a double piped iterated compression function. As part of our cryptanalysis of the EDON-$\mathcal{R}$ hash function we present here results from tests of randomness performed on EDON-$\mathcal{R}$ compression functions reduced in size.

Using the same theory as [GØM$^+$09b], we constructed an 8-bit EDON-$\mathcal{R}$ compression function. This construction is small enough that we can test all 2^{32} possible inputs of messages and chaining values. The distribution of the output was then compared to what is expected from an ideal random function. In addition we tested to see if the function is regular.

Similarly, we also constructed a 16-bit EDON-$\mathcal{R}$ compression function. For 300 different chaining values chosen at random, we tested all 2^{32} possible message inputs. The distribution of the output was then compared to what is expected from an ideal random function. We also used the results to compute the balance of the 16-bit compression function.

This paper is organized as follows. We first give the required background in Section 2. In Section 3 and Section 4 we construct and analyze 8-bit and 16-bit EDON-$\mathcal{R}$ respectively. Finally Section 6 concludes the paper.

2. Background

In this section we give the required mathematical background for this paper. The underlying structure of EDON-$\mathcal{R}$ are quasigroups of order 2^{kw} where $kw =$ 256 and 512 for EDON-$\mathcal{R}$-256 and EDON-$\mathcal{R}$-512 respectively.

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

$$\begin{aligned} a * x &= b \\ y * a &= b \end{aligned} \tag{1}$$

has unique solutions x and y in Q.

The compression function $\mathcal{R}_{kw}$ of EDON-$\mathcal{R}$ is a series of quasigroup operations of the form

$$\mathbf{X} *_{kw} \mathbf{Y} = \pi_{1,kw}(\pi_{2,kw}(\mathbf{X}) + \pi_{3,kw}(\mathbf{Y})), \tag{2}$$

where the permutations $\pi_{i,kw}$ treat $\mathbf{X}, \mathbf{Y}$ as vectors of $k = 8$ words of size $w = 32, 64$ bits. The permutation $\pi_{1,kw}$ is a simple reordering of the variables. The permutations $\pi_{2,kw}$ and $\pi_{3,kw}$ are defined from two orthogonal Latin squares of size $k = 8$ (the same size as the vectors). For a detailed explanation of how these permutations are defined from the Latin squares we refer the reader to [GØM$^+$09b].

DEFINITION 2 *A* Latin square *of size k is an $k \times k$-matrix whose elements are the numbers* $0, \ldots, k-1$ *and each number appears exactly one time in each row and each column.*

The compression function is defined by repeated use of the quasigroup operation as shown in Figure 1. This gives the following formula for the chaining values $(\mathbf{B}_0, \mathbf{B}_1)$

$$\mathcal{R}_{kw}(\mathbf{C}_0, \mathbf{C}_1, \mathbf{A}_0, \mathbf{A}_1) = (\mathbf{B}_0, \mathbf{B}_1) \tag{3}$$

where

$$\begin{aligned} \mathbf{B}_0 &= \overline{\mathbf{A}}_0 * ((\mathbf{C}_1 * (\overline{\mathbf{A}}_1 * \mathbf{A}_0)) * \mathbf{C}_0) \\ \mathbf{B}_1 &= (\overline{\mathbf{A}}_0 * ((\mathbf{C}_1 * (\overline{\mathbf{A}}_1 * \mathbf{A}_0)) * \mathbf{C}_0)) * \\ &\quad (((\mathbf{C}_1 * (\overline{\mathbf{A}}_1 * \mathbf{A}_0)) * ((\overline{\mathbf{A}}_1 * \mathbf{A}_0) * \mathbf{A}_1)) * \\ &\quad ((\mathbf{C}_1 * (\overline{\mathbf{A}}_1 * \mathbf{A}_0)) * \mathbf{C}_0)). \end{aligned} \tag{4}$$

DEFINITION 3 *A function is said to be regular if every possible output has* 2^m *preimages for some m.*

In [BK04] Bellare and Kohno looked at the connection between the "amount of regularity" of a hash function and the general $2^{n/2}$ bound for birthday attacks on hash functions of size n. Their conclusion was that $2^{n/2}$ was the lower bound only if the hash function was regular, and that the actual lower bound can be significantly less depending on the regularity of the hash function. Bellare and Khono introduced *balance* as a measure of regularity and used its

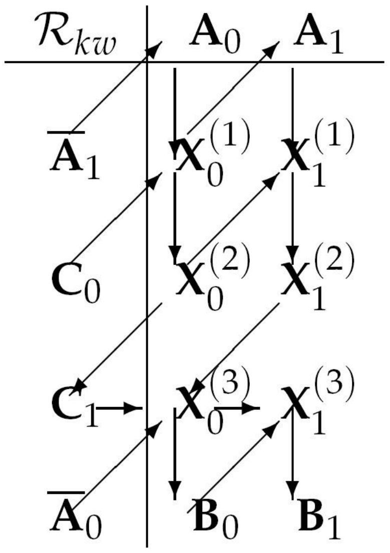

Figure 1. Schematic representation of the function $\mathcal{R}_{kw}$. The diagonal arrows can be interpreted as quasigroup operations between the source and the destination, and the vertical or the horizontal arrows as equality signs "=".

properties to prove the relation between balance and the expected number of trials in the birthday attack.

DEFINITION 4 *Let $h : R \to D$ be a function whose domain D and range $R = \{R_1, \dots, R_r\}$ have sizes $d, r \geq 2$, respectively. For $i \in \{1, \dots, r\}$ let $d_i = |h^{-1}(R_i)|$ denote the size of the pre-image of R_i under h. The balance of h, denoted $\mu(h)$, is defined as*

$$\mu(h) = \log_r \left[\frac{d^2}{d_1^2 + \cdots + d_r^2} \right] \tag{5}$$

where $\log_r(\cdot)$ denotes the logarithm in base r.

Note that Bellare and Khono showed that a Merkle-Damgrd transform does not necessarily preserve balance. This means that in addition to the balance of the compression function, the balance of the whole hash function should also be investigated [BK04].

3. Analysis of 8-bit Edon-$\mathcal{R}$

To test some of the properties of the function $\mathcal{R}_{kw}$ we constructed a small version using the same theory. Setting the size of the vectors to $w = 4$ and

$$L_1 = \left[\begin{array}{cccc} 0 & 1 & 2 & 3 \\ 1 & 0 & 3 & 2 \\ 2 & 3 & 0 & 1 \\ \hline 3 & 2 & 1 & 0 \end{array}\right] \qquad L_2 = \left[\begin{array}{cccc} 0 & 1 & 2 & 3 \\ 2 & 3 & 0 & 1 \\ 3 & 2 & 1 & 0 \\ \hline 1 & 0 & 3 & 2 \end{array}\right]$$

Table 1. Two mutually orthogonal Latin squares used to define the permutations $\pi_{2,8}$ and $\pi_{3,8}$

Quasigroup operation of order 2^8
Input: $\mathbf{X} = (X_0, X_1, X_2, X_3)$ and $\mathbf{Y} = (Y_0, Y_1, Y_2, Y_3)$ where X_i and Y_i are 2–bit variables. **Output:** $\mathbf{Z} = (Z_0, Z_1, Z_2, Z_3)$ where Z_i are 2–bit variables. **Temporary 2–bit variables:** $T_0, \ldots, T_7$.

```
1   $T_4 \leftarrow ROTL^0($0x1$ + X_0 + X_1 + X_2);$
    $T_5 \leftarrow ROTL^1( X_0 + X_1 + X_3);$
    $T_6 \leftarrow ROTL^0( X_0 + X_2 + X_3);$
    $T_7 \leftarrow ROTL^1( X_1 + X_2 + X_3);$

2   $T_0 \leftarrow ROTL^0($0x2$ + Y_0 + Y_2 + Y_3);$
    $T_1 \leftarrow ROTL^1( Y_1 + Y_2 + Y_3);$
    $T_2 \leftarrow ROTL^0( Y_0 + Y_1 + Y_2);$
    $T_3 \leftarrow ROTL^1( Y_0 + Y_1 + Y_3);$

3   $Z_3 \leftarrow T_7 + T_1;$
    $Z_2 \leftarrow T_6 + T_0;$
    $Z_0 \leftarrow T_5 + T_2;$
    $Z_1 \leftarrow T_4 + T_3;$
```

Table 2. An algorithmic description of the quasigroup operation of order 2^8.

the size of the words to $k = 2$, an 8-bit version was constructed using the two orthogonal Latin squares in Table 1. Alternating between left rotation of 0 and 1 we arrived at the quasigroup operation in Table 2

One notable difference between this quasigroup operation and the ones defined in [GØM+09b] is the missing XOR parts. The reason for this difference is that here there is only one row below the line in the Latin squares used to define the permutations. So instead of XORing the variables, they are permuted according to the rows $(3, 2, 1, 0)$ and $(1, 0, 2, 3)$.

3.1 Experiments and results for 8-bit Edon-$\mathcal{R}$

We have now constructed a reduced EDON-$\mathcal{R}$ compression function $R_8 : 2^{32} \rightarrow 2^{16}$ which is small enough to exhaustively go trough all possible input values. To test some of the properties of this function we performed the following two experiments.

EXPERIMENT 1 *The first test we performed was on the number of collisions of the compression function. Holding the chaining values* ($\mathbf{C}_0, \mathbf{C}_1$) *constant, we used all possible pairs* ($\mathbf{M}_0, \mathbf{M}_1$) *of messages as input to the compression function.*

$$R_8(\mathbf{C}_0, \mathbf{C}_1, \mathbf{M}_0, \mathbf{M}_1) = (\mathbf{B}_0, \mathbf{B}_1) \tag{6}$$

The output ($\mathbf{B}_0, \mathbf{B}_1$) *represented as a number between* 0 *and* $2^{16} - 1$ *was then tallied with respect to collisions. That is, we counted the number of elements mapped to* 0 *times, the number of elements mapped to* 1 *times and so on. This was done for all possible pairs of chaining values. The result was then compared to what is expected for an ideal random function (IRF).*

If $\mathcal{R}_8$ *is an IRF each element in the image should be mapped to by the probability* $p = 2^{-16}$*. This means that the an element will be mapped to* l *times out of* n *with probability*

$$P(l) = \binom{n}{l} p^k (1 - p)^{n-l}, \tag{7}$$

where $n = 2^{16}$*. This means that expected number of elements mapped to* l *times is* $nP(l)$*.*

RESULT 1 *The result from this experiment is listed in Table 3 and shown in Figure 2. From the table we see that even the very reduced* $\mathcal{R}_8$ *function has a distribution similar to an IRF. For* $l = 0, 1, 2, 3$ *the results are good with at most 2% difference from an IRF. However the difference rapidly grows for bigger* l*. For all* l *the 95% confidence interval for the mean was outside what is expected for an IRF. This means that the difference between* $\mathcal{R}_8$ *and an IRF is statistically significant. We also noticed some other non-random behavior. Sorting the result according to the first chaining value* $\mathbf{C}_0$ *we see that all occurrences of* 9 *and* 10 *collisions are centered around certain values of* $\mathbf{C}_0$*. The same is not true if we sort with respect to* $\mathbf{C}_1$*. This means the first chaining value has bigger influence on the final output then the second. Looking at the construction of the compression function in Figure 1 this result is of course expected since the first chaining value is introduced earlier in the computation.*

EXPERIMENT 2 *The second test we performed was on the number of pre-images each element has under the compression function. We exhaustively*

l	Min	Max	Mean	IRF	Difference %
0	23285	24108	23694,53	24109.16	1.72%
1	24007	25102	24538.69	24109.53	-1.78 %
2	11902	12624	12251.78	12054.77	-1.63 %
3	3704	4142	3936.85	4018.19	2.02 %
4	792	1046	919.00	1004.52	8.51 %
5	117	224	167.37	200.89	16.69 %
6	9	47	24,46	33.48	26.94 %
7	0	12	2.99	4.78	37.51 %
8	0	4	0.30	0.60	50.17 %
9	0	3	0.027	0.066	59.56 %
10	0	1	0.0015	0.0066	77.94 %
≥ 11	0	0	0	0.00066	100.00 %

Table 3. The distribution of the image of $\mathcal{R}_8$ for all possible pairs of chaining values. The second last columns show what is expected for an ideal random function (IRF), while the first 3 columns show the result for $\mathcal{R}_8$. The last column show the difference in percent between the IRF and the mean of $\mathcal{R}_8$.

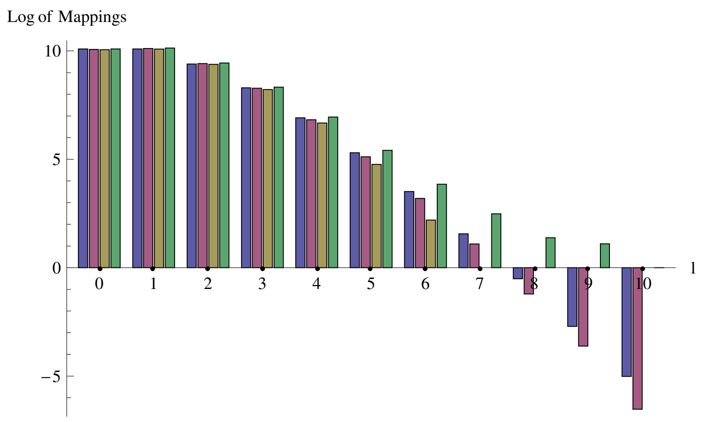

Figure 2. A bar chart comparing the binomial distribution of an IRF with $\mathcal{R}_8$. The x-axis is how many times, l, an element is mapped to, while the y-axis is the logarithm of how many elements is mapped to l times. The first bar is the IRF, the second bar is the mean of $\mathcal{R}_8$, while the third and fourth bar are the minimum and maximum of $\mathcal{R}_8$ respectively.

went trough all 2^{32} possible input values and looked at how many times each element in the image was mapped to.

RESULT 2 *The result from this experiment was that each element was mapped to exactly 2^{16} times. This means that the function $\mathcal{R}_8$ is regular according to Definition 3.*

4. Analysis of 16-bit Edon-$\mathcal{R}$

We also constructed a 16-bit version of $\mathcal{R}_{kw}$ by setting the size of the vectors to $k = 8$ and the wordsize to $w = 2$. For this construction we used the same Latin squares as in the construction of EDON-$\mathcal{R}$256 and 512. This means that the algorithmic description of the quasigroup operation of order 2^{16} are very similar to the ones found in [GØM$^+$09b]. The only difference is that we alternated between rotation of 0 and 1 instead of the rotations described for EDON-$\mathcal{R}$256 and 512.

4.1 Experiment and results for 16-bit Edon-$\mathcal{R}$

The compression function $\mathcal{R}_{16} : 2^{64} \to 2^{32}$ is to big to go trough all possible input values. However we can still perform an experiment on the distribution of the output similar to Experiment 1 in Section 3.

EXPERIMENT 3 *Holding the chaining values ($\mathbf{C}_0, \mathbf{C}_1$) constant we used all possible pairs $(\mathbf{M}_0, \mathbf{M}_1)$ of messages as input to the compression function.*

$$R_{16}(\mathbf{C}_0, \mathbf{C}_1, \mathbf{M}_0, \mathbf{M}_1) = (\mathbf{B}_0, \mathbf{B}_1) \tag{8}$$

The output $(\mathbf{B}_0, \mathbf{B}_1)$ represented as a number between 0 and $2^{32} - 1$ was then tallied with respect to collisions. This test was performed for 300 different pairs of chaining values chosen at random.

If $\mathcal{R}_{16}$ is an ideal random function, each element in the image should be mapped to by the probability $p = 2^{-32}$. This means that the an element will be mapped to l times out of n with probability

$$P(l) = \binom{n}{l} p^k (1-p)^{n-l}, \tag{9}$$

where $n = 2^{32}$. Which means that expected number of elements mapped to l times is $nP(l)$.

RESULT 3 *The result from this experiment is listed in Table 4. Note that, because of some inaccuracy*[1] *in the computation of the distribution of the ideal*

[1]The number $(1 - 2^{-32})^{2^{32}-l}$ could only be approximated using Mathematica 6.0.

l	Min	Max	Mean	IRF	Diff %
0	1579952990	1580105601	1580014613	1580030157	0.00098 %
1	1579915274	1580140186	1580047057	1580030350	-0.0011 %
2	789949319	790105082	790021531	790016352	-0.00066 %
3	263295979	263371739	263335106	263337413	0.00088%
4	65813150	65860809	65831618.77	65834545.53	0.0044%
5	13154784	13174678	13165622.68	13166934.63	0.010%
6	2190432	2198973	2194306.50	2194480.82	0.0079%
7	312150	314757	313445.75	313498.08	0.017%
8	38656	39803	39169.01	39187.43	0.047%
9	4139	4563	4349.83	4354.11	0.098%
10	361	491	433.45	435.41	0.45%
11	23	57	39.09	39.58	1.24%
12	0	10	3.46	3.30	-4.99%
13	0	3	0.26	0.25	-3.78%
14	0	1	0.017	0.018	8.041%
≥ 15	0	0	0	NA	NA

Table 4. The distribution of the image of $\mathcal{R}_{16}$ for all possible pairs of chaining values. The second last columns show what is expected for an ideal random function (IRF), while the first 3 columns show the result for $\mathcal{R}_{16}$. The last column show the difference in percent between the IRF and the mean of $\mathcal{R}_{16}$.

random function, the sum of the expected results for 0 to 14 is slightly more then 2^{32}*. The expected number of collisions larger then 15 is therefor listed as not available. Other approximations show this number to be in the order of* 10^{-3}.

From the table we see that the compression function of 16-bit EDON-$\mathcal{R}$ *has an output distribution very similar to an IRF. For most values of l the difference between* R_{16} *and an IRF is much less then* 1%*. For* $l = 11, 12, 13$ *and 14 the difference is larger. Some of the reason for this difference is that the probability for 11 or more mappings colliding is very small and the variance for such collisions is therefor higher. This is also reflected in the 95% confidence intervals for the mean. For* $l = 1, \ldots, 6$ *the computed confidence intervals is outside what is expected for an IRF, while for* $l = 7, \ldots, 14$ *the confidence intervals for the mean contains what is expected for an IRF. This means that altough the output distribution of* $\mathcal{R}_{16}$ *is very similar to an IRF they are still significantly different.*

EXPERIMENT 4 *Because of the size of* $\mathcal{R}_{16}$ *we were not able to test wether the function is regular. We did however tally how many times of the* $300 * 2^{32}$

different input values each element was mapped to. The tally was used to compute the Bellare-Khono balance of $\mathcal{R}_{16}$ as defined in Definition 4.

Result 4 *Computing the sum of the square of the number of preimages for each of the 2^{32} possible output values we got the number 387835522366350. This gives the following result for the balance.*

$$\mu(\mathcal{R}_{16}) = \log_{2^{32}} \left[\frac{(300 * 2^{32})^2}{387835522366350} \right] = 0.99985 \tag{10}$$

We will also quickly mention some other possibly interesting numbers from this experiment. The least amount of times a number was mapped to was 194, while 413 was the most amount of times a number was mapped to (300 of course being the average). The variance was $\sigma^2 = 299.9943$.

The results for the regularity of $\mathcal{R}_8$ and the balance of $\mathcal{R}_{16}$ together with the analysis for delta deviations in our documentation of Edon-$\mathcal{R}$ [GØM+09b] are strong evidence that the balance is very high for $\mathcal{R}_{kw}$ in general. An open and interesting question is for what k and w the function $\mathcal{R}_{kw}$ is completely regular.

5. Conclusion

We have shown that the reduced compression function $\mathcal{R}_8$ is regular and that its output distribution is similar to that of an ideal random function. We have also shown that distribution of the output of $\mathcal{R}_{16}$ is very similar to an ideal random function, and that the balance of $\mathcal{R}_{16}$ is high.

Comparing Table 3 and Table 4 we see that the amount of randomness drastically increases as we increase the size of the range and the domain of the compression function. Intuitively we expect this trend to follow as we increase the size of the range and the domain of the compression function even more.

Based on the results of this paper it is difficult to give a general prediction for the correlation between the size of the compression function and its balance. It is possible that also $\mathcal{R}_{16}$ is completely regular, but unfortunately we do not have the computer power to test this. Doing some tests on the balance of $\mathcal{R}_{256}$ and $\mathcal{R}_{512}$ would be quite interesting in this regard.

Additionally, it is clear that our methodology of analyzing Edon-$\mathcal{R}$ by reducing the size of the variables and then investigating the properties of such severely reduced function can be applied to all hash functions.

References

[Be11] Daniel J. Bernstein and Tanja Lange (editors). eBASH:ECRYPT Benchmarking of All Submitted Hashes. measurements of hash functions, 2011. Accessed

May 2011.

[BK04] Mihir Bellare and Tadayoshi Kohno. Hash function balance and its impact on birthday attacks. In *Advances in Cryptology - EUROCRYPT 04*, volume 3027 of *LNCS*, pages 401–418. Springer-Verlag, 2004.

[GØ09] Danilo Gligoroski and Rune Steinsmo Ødegård. On the complexity of Khovratovich et al.'s preimage attack on EDON-$\mathcal{R}$. Available online, 2009. url:`http://eprint.iacr.org/2009/120.pdf`.

[GØM$^+$09] Danilo Gligoroski, Rune Steinsmo Ødegård, Marija Mihova, Svein Johan Knapskog, Ljupco Kocarev, Aleš Drápal, and Vlastimil Klima. Cryptographic hash function EDON-$\mathcal{R}$. Submission to NIST, 2009.

[Kli08] Vlastimil Klima. Multicollisions of EDON-$\mathcal{R}$ hash function and other observations. Available online, 2008. url:`http://cryptography.hyperlink.cz/BMW/EDONR_analysis_vk.pdf`.

[KNW08] Dmitry Khovratovich, Ivica Nikolić, and Ralf-Philipp Weinmann. Cryptanalysis of EDON-$\mathcal{R}$. Available online, 2008. url:`http://ehash.iaik.tugraz.at/uploads/7/74/Edon.pdf`.

[Leu09] Gaëtan Leurent. Key recovery attack against secret-prefix EDON-$\mathcal{R}$. Cryptology ePrint Archive, Report 2009/135, 2009.

[NIS07] NIST. Announcing request for candidate algorithm nominations for a new cryptographic hash algorithm (SHA-3) family. Federal Register Notice, 72(112), November 2007. `http://csrc.nist.govgroups/ST/hash/documents/FR_Notice_Nov07.pdf`.

PAPER D

Distinguishers for the Compression Function and Output Transformation of Hamsi-256

Jean-Philippe Aumasson, Emilia Käsper, Lars Ramkilde Knudsen, Krystian Matusiewicz, Rune Steinsmo Ødegård, Thomas Peyrin and Martin Schläffer.

In Proceedings of the 15th Australian Conference on Information Security and Privacy

Sydney, Australia. July 5-7, 2010.

DISTINGUISHERS FOR THE COMPRESSION FUNCTION AND OUTPUT TRANSFORMATION OF HAMSI-256

Jean-Philippe Aumasson
Nagravision SA
Cheseaux, Switzerland
jeanphilippe.aumasson@gmail.com

Emilia Käsper
K.U.
Leuven, Belgium
Emilia.Kasper@esat.kuleuven.be

Lars Ramkilde Knudsen, Krystian Matusiewicz
Department of Mathematics,
Technical University of Denmark
{Lars.R.Knudsen,K.Matusiewicz}@mat.dtu.dk

Rune Steinsmo Ødegård
Centre for Quantifiable Quality of Service in Communication Systems.
Norwegian University of Science and Technology
Trondheim, Norway
rune.odegard@q2s.ntnu.no

Thomas Peyrin
Ingenico,
France
thomas.peyrin@gmail.com

Martin Schläffer
IAIK, TU
Graz, Austria
martin.schlaeffer@iaik.tugraz.at

Abstract Hamsi is one of 14 remaining candidates in NIST's Hash Competition for the future hash standard SHA-3. Until now, little analysis has been published on its resistance to differential cryptanalysis, the main technique used to attack hash functions. We present a study of Hamsi's resistance to differential and higher-order differential cryptanalysis, with focus on the 256-bit version of Hamsi. Our main results are efficient distinguishers and near-collisions for its full (3-round) compression function, and distinguishers for its full (6-round) finalization function, indicating that Hamsi's building blocks do not behave ideally.

Keywords: hash functions, differential cryptanalysis, SHA-3

1. Introduction

Hash functions are one of the most ubiquitous primitives in cryptography, with digital signatures and integrity checks as their main applications. Collision attacks on the deployed standards MD5 and SHA-1 [WLF^{+}05, WY05, WYY05b, WYY05a] have weakened the confidence in the MD family of hash functions. Hence, the US Institute of Standards and Technology (NIST) launched a public competition to develop a future SHA-3 standard [NIS07].

The hash function Hamsi [KÖ9a] is one of 64 designs submitted to NIST in fall 2008. Hamsi is also one of the 14 submissions selected for the second round of the competition in July 2009 as one of the few submissions with no major weaknesses detected thus far. While Hamsi reuses the round components of the Serpent block cipher [BAK98], its larger block size and different round structure make existing cryptanalytic results on Serpent hardly useful in its security analysis.

So far, little research has been published on the resistance of Hamsi to common cryptanalytic attacks: in a work independent from ours, Çalık and Turan studied differential properties of Hamsi-256, and presented message-recovery and pseudo-second-preimage attacks. Near collisions were studied by Nikolić [Nik09] and Wang et al. [WWJW09], as discussed in Section 4.3.

We study the resistance of Hamsi to differential and higher-order differential cryptanalysis, with focus on the 256-bit version Hamsi-256. In Section 3, we show by higher-order analysis that the 3-round compression function of Hamsi-256 does not achieve maximal degree. This is demonstrated by showing that the output of certain related chaining values (with fixed message word) or related message words (with fixed chaining value) sums to zero with a high probability.

In Sections 4 and 5, we focus on differential cryptanalysis and construct high-probability differential paths for the 3-round compression function as well as the full 6-round output transformation. The former gives near-collisions on $(256 - 25)$ bits of the compression function output, with only six differences in the input chaining value. Section 4 describes a technique for building low-weight, high-probability differential paths for Hamsi. Finally, Section 5 presents differential paths for six rounds of Hamsi-256 that show that the output transformation of Hamsi-256 does not behave ideally.

2. Description of Hamsi-256

This section describes the hash function Hamsi-256, henceforth just called Hamsi. We refer to [K0̈9a] for a complete specification.

2.1 High-level structure

Like most hash functions, Hamsi builds on a finite-domain *compression function*, which is used to process arbitrary-length messages through the use of a *domain extender* (or operation mode). The compression function of Hamsi can be divided into four operations:

Message expansion	$E : \{0,1\}^{32} \to \{0,1\}^{256}$
Concatenation	$C : \{0,1\}^{256} \times \{0,1\}^{256} \to \{0,1\}^{512}$
Non-linear permutations	$P, P_f : \{0,1\}^{512} \to \{0,1\}^{512}$
Truncation	$T : \{0,1\}^{512} \to \{0,1\}^{256}$

The message M to hash is appropriately padded and split into ℓ blocks of 32 bits: $M_1, \ldots, M_\ell$. Each block is iteratively processed by the compression function, which operates on a 512-bit internal state viewed as a 4×4 matrix of 32-bit words.

Figure 1 depicts an iteration of the compression function H (or H_f). Starting from the predefined initial value (IV) h_0, Hamsi iteratively computes the digest h of M as follows:

$$\begin{aligned} h_i &= H(h_{i-1}, M_i) = (T \circ P \circ C(E(M_i), h_{i-1})) \oplus h_{i-1} \quad \text{for} \quad 0 < i < \ell \\ h &= H_f(h_{\ell-1}, M_\ell) = (T \circ P_f \circ C(E(M_\ell), h_{\ell-1})) \oplus h_{\ell-1} \end{aligned}$$

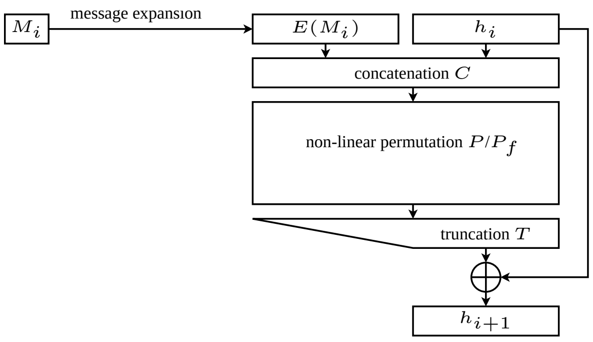

Figure 1. Domain extension algorithm of Hamsi.

2.2 Internals of the compression function of Hamsi

2.2.1 Message expansion.

The message expansion of Hamsi uses a linear code to expand a 32-bit word into eight words (that is, 256 bits). We write an expanded M_i as $(m_0, \ldots, m_7)$. Thus, the m_j's are defined as the product of a multiplication with the generator matrix of the code:

$$E(M_i) = (m_0, \ldots, m_7) = (M_i \times G) \ ,$$

where G can be found in [KÖ9a].

2.2.2 Concatenation.

The concatenation function C forms a 512-bit internal state from the 256-bit expanded message $(m_0, \ldots, m_7)$ and the 256-bit incoming chaining value $h_i = (c_0, \ldots, c_7)$ (Figure 2):

$$\begin{aligned} &C(m_0, m_1, \ldots, m_7, c_0, c_1, \ldots, c_7) \\ &\qquad = (m_0, m_1, c_0, c_1, c_2, c_3, m_2, m_3, m_4, m_5, c_4, c_5, c_6, c_7, m_6, m_7) \ . \end{aligned}$$

2.2.3 Truncation.

The truncation function T selects eight 32-bit words among the 16 from the internal state to form the new chaining value after feedforward (Figure 3):

$$T(s_0, s_1, s_2, \ldots, s_{14}, s_{15}) = (s_0, s_1, s_2, s_3, s_8, s_9, s_{10}, s_{11}) \ .$$

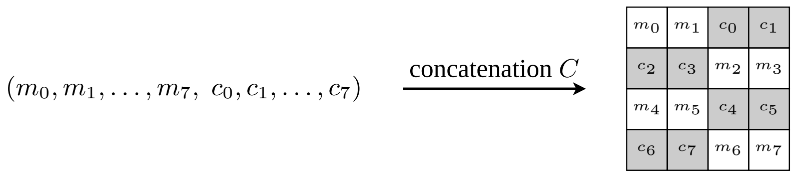

Figure 2. Concatenation of expanded message words $m_0, \ldots, m_7$ and chaining value words $c_0, \ldots, c_7$ in Hamsi.

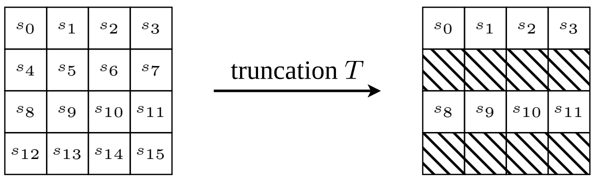
s_0 s_1 s_2 s_3
s_4 s_5 s_6 s_7
s_8 s_9 s_{10} s_{11}
s_{12} s_{13} s_{14} s_{15}
truncation T
s_0 s_1 s_2 s_3
s_8 s_9 s_{10} s_{11}

Figure 3. Truncation selects eight out of 16 words of the internal state.

2.2.4 Permutations.

Finally, we describe the permutations P and P_f. They only differ in the number of rounds (three for P and six for P_f)[1] and in the round constants. The round function is composed of three layers. First, constants and a counter are XORed to the whole internal state. Then there is a substitution layer, followed by a linear layer.

The substitution layer uses one 4-bit Sbox of the block cipher Serpent [BAK98], in a bitsliced way. That is, four bits, one from each of the four 32-words of the same column in the 4×4 internal state matrix are first extracted and then replaced after application of the Sbox. We denote s_i^j the j-th bit of the internal state word s_i. The substitution layer can be described as follows, for $0 \leq j \leq 31$ and $0 \leq i \leq 3$:

$$(s_i^j, s_{i+4}^j, s_{i+8}^j, s_{i+12}^j) := S(s_i^j, s_{i+4}^j, s_{i+8}^j, s_{i+12}^j)\ ,$$

where S is the 4×4 Sbox given in Table A.1 (Appendix 7).

[1]While 6 rounds remains the official parameter, the designer has suggested 8 rounds as a conservative alternative. Our results indicate that moving to 8 rounds may be a necessary precaution.

The linear diffusion layer applies the Serpent linear transform $L : \{0,1\}^{128} \to \{0,1\}^{128}$ to each of the four diagonals of the state, as follows:

$$\begin{aligned}(s_0, s_5, s_{10}, s_{15}) &:= L(s_0, s_5, s_{10}, s_{15})\\ (s_1, s_6, s_{11}, s_{12}) &:= L(s_1, s_6, s_{11}, s_{12})\\ (s_2, s_7, s_8, s_{13}) &:= L(s_2, s_7, s_8, s_{13})\\ (s_3, s_4, s_9, s_{14}) &:= L(s_3, s_4, s_9, s_{14})\ .\end{aligned}$$

The algorithm below (read from top-left to bottom-right) describes the linear transform L on input (a, b, c, d), with $x \lll k$ denoting the left bit rotation of k positions on the word x and $x \ll k$ denoting the left bit shift of k positions on the word x.

```
$a := a \lll 13$                      |  $d := d \lll 7$
$c := c \lll 3$                       |  $a := a \oplus b \oplus d$
$b := a \oplus b \oplus c$            |  $c := (b \ll 7) \oplus c \oplus d$
$d := (a \ll 3) \oplus c \oplus d$    |  $a := a \lll 5$
$b := b \lll 1$                       |  $c := c \lll 22$
```

3. Higher-order differential analysis

This section reports on properties of Hamsi related to higher-order derivatives. After some definitions, we present upper bounds on the algebraic degree of Hamsi's compression function and show how to exploit them to find "k-sums" and "zero-sums". This illustrates the fact that, due to its low algebraic degree, the compression function of Hamsi does not behave ideally.

3.1 Definitions

3.1.1 Higher-order derivatives.

Higher-order differential analysis [Knu94, Lai92] of cryptographic algorithms generalizes the notion of differential cryptanalysis by considering derivatives of order two or more. It is based on the basic observation that for a function f with algebraic degree $s \geq 1$, the degree of a dth-order derivative of f is at most $(s - d)$, where $s \geq d$. Consequently, an sth-order derivative of f is a constant and an $(s + 1)$st-order derivative of f is zero, which directly gives a 2^{s+1}-sum for f.

In the following we consider derivatives of functions with domain $\{0,1\}^n$, $n \geq 1$ and range $\{0,1\}$. Note that a (certain type of) d-th order derivative is then the XOR of 2^d values of the function for the 2^d choices of d input bits.

3.1.2 k-sums.

The k-sum problem is, given k lists of random n-bit values (for example, k distinct instances of a compression function $f_1, \ldots, f_k$) , to find one value from each list such that the sum of the k values is zero. The case $k = 2$ is essentially the collision problem.

The k-sum problem can be solved in polynomial time (using the XHASH attack [BM97]) when $k \geq n$. However, the problem is believed to be hard for small k. The standard method for the k-sum problem with small k is Wagner's "generalized birthday" method, which requires time and space $O(k2^{n/(1+\log k)})$ [Wag02] (see also [Ber07]).

Henceforth, we consider the problem of finding k values whose images by a same function f sum to zero. Note that if f has degree $s < (n-1)$, then a 2^{s+1}-sum can be found by returning the values corresponding to a $(s+1)$st order derivative.

An example of application of k-sums is to forge message authentication codes (MACs). Let H he a hash function and consider the "prefix-MAC" construction defined as $\mathrm{MAC}_K(m) = \mathrm{trunc}\,(H(K\|m))$, where trunc is a function removing some bits of the hash output to combat length extension attacks. Assume we know messages $m_1, \ldots, m_k$ such that the probability

$$\Pr_K \left[\bigoplus_{i=1}^{k} H(K\|m_i) = 0 \right] = p$$

is nonzero. Then by querying MAC_K with $m_1, \ldots, m_{k-1}$ we can determine $\mathrm{MAC}_K(m_k)$ with probability p and thus break the existential forgery of MAC.

This can be generalized to messages whose MAC tags sum to any fixed value, to other MAC constructions, etc. For example, one may fix a message and forge the MAC $H_K(m)$ where K is the IV of H by making related-key queries.

3.1.3 Zero-sums.

We define the zero-sum problem as a particular case of the k-sum problem: given a function f, find distinct values that sum to zero such that their images by f also sum to zero.

Both the XHASH attack [BM97] and Wagner's generalized birthday [Wag02] can be adapted to find zero-sums. These methods are generic, and are probabilistic algorithms whose failure probability can be made exponentially small.

3.2 On the degree of the compression function

3.2.1 Simple bounds.

The only nonlinear component of Hamsi's compression function is the layer of 4×4 Sboxes. One round thus has degree three (see [SAB09] for explicit expressions of the Sboxes used), so N rounds have degree at most 3^N, with respect to any choice of variables.

If variables are chosen in $c_0, \dots, c_3$ only, or in $c_4, \dots, c_7$ only, then they are all in distinct slices and thus go into distinct Sboxes in the first round. Hence, the first round is linear and after N rounds, the degree is at most 3^{N-1}. This means that the degree is at most 81 after five rounds, and that at least six rounds are necessary to reach maximal degree. In particular, the 3-round compression function has degree at most 9 with respect to choices of 128 variables in distinct slices, which distinguishes it from a randomly chosen function (whose degree would be below 9 with negligible probability).

3.2.2 Case of four variables.

If four variables are chosen in the LSB's of $c_0, \dots, c_3$, after the first application of the Sbox, all the LSB's of a given word depend on the bit varied in the corresponding column. Since only one bit is varied per column, the degree of equations corresponding to LSB's are of degree 1. Then, the linear function $L(a, b, c, d)$ is applied to each column, and we can determine, for a given bit of the state, whether it depends on the single variable of its diagonal. Based on this, we can determine whether a given 4-bit slice depends on 1, 2, 3, or 4 of the variables.

A simple computer-assisted analysis revealed that each slice depends on only one variable. Therefore, the (3-round) compression function of Hamsi always has degree 3 with respect to four variables in the first four LSB's, for any values of the other bits. Ideally, the function should have degree 4 with probability 1/2, over the choice of the other input bits.

3.3 Finding k-sums for the compression function

For randomly chosen 256-bit values, finding 4-sums for the compression function of Hamsi requires an effort of complexity approximately $4 \cdot 2^{256/3} \approx 2^{87}$, using the generalized birthday method. Below we show efficient methods to find 16-, 8-, and 4-sums.

3.3.1 16-sums.

Recall the above observation that three rounds have degree at most 3 with respect to a certain choice of four variables. This observation can directly be used to find 16-sums, without any computation. Based on empirical observations, we discovered that we can do better, as presented below.

3.3.2 8-sums.

Choose a random value of one 256-bit chaining value, then select seven other chaining values, which are different from the first one only in the LSB's of the first three 32-bit words. Denote these chaining values by $h^0, \ldots, h^7$. Choose a random 32-bit message block M, then compute $\sum_{i=0}^{7} H(h^i, M)$, In each of 1 000 000 such tests, the above sum was zero in 1458 cases (whereas for a random mapping, the probability to obtain zero is negligible). This indicates that there are 3rd-order derivatives with the value zero (or 8-sums) of a high probability for the compression function of Hamsi. It is very likely that one can identify other 3rd-order derivatives of higher probabilities (our search was limited).

3.3.3 4-sums.

We found 2nd-order derivatives with value zero, that is, 4-sums. One example is when one chaining value is the IV of Hamsi specified in [KÖ9b], and where the three others differ only in two LSB's of the second words; the XOR of the four outputs is the all-zero string (note that the four inputs also sum to zero, thus this is also a zero-sum).

Via an exhaustive search over all 2^{32} message words, we identified 70 messages for which the above four chaining values lead to a 4-sum. We also found 4-sums for the IV given in [KÖ9a], for 86 values of the 32-bit message block. Although complete analytical justification of these observations remains to be found, the results of these observations strongly differ from what one obtains for a random mapping (for which a 2nd-order derivative is zero with negligible probability).

3.3.4 *k*-sums for fixed chaining value.

Here we report on the case where the chaining value is fixed and where only the message block is varied. The outputs of the compression function in this case has a much higher algebraic degree.

Consider h_0, the IV specified in [KÖ9b], and 2^{19} values of the 32-bit message block obtained by varying the first and second bytes, and the three least significant bits of the third byte. The remaining bits can be fixed to arbitrary values. Denoting these message words by $m_0, \ldots, m_{2^{19}-1}$, we have:

$$\bigoplus_{i=0}^{2^{19}-1} H(h_0, m_i) = 0 \ .$$

This observation holds for any initial chaining variable. Here we obtain zero because we perform a 19th-order derivative of a function of degree 18 only. Indeed, in the first round at most two bit variables enter a same Sbox, hence the degree of the first round is 2. Since the two subsequent rounds have degree 3 each, the three rounds have degree $2 \times 3 \times 3 = 18$.

Note that if P_f is replaced by P in Hamsi's domain extender, then the above observation can be used to forge MAC's (cf. Section 3.1), which shows that the extended 6-round output transformation is necessary, and cannot be removed without compromising the security of Hamsi.

3.4 Finding zero-sums for the output permutation

We describe a dedicated method to find large zero-sums for the 6-round permutation of the finalization function of Hamsi(we stress that it only applies to the internal permutation and not to the finalization as a whole, for it puts no restriction on the initial state). Contrary to Wagner's and the XHASH methods, it is deterministic rather than probabilistic, and needs to evaluate (and to know) only half the function.

In the spirit of [Wag99, §9], we present an "inside-out" technique that exploits the fact that two *halves* of Hamsi's permutation have low algebraic degree. This differs from our method for finding k-sums which exploited the low degree of the full permutation. The attack works as follows:

1 Choose an arbitrary value for the state of Hamsi's permutation after three rounds.

2 Choose 28 distinct bits of the state.

3 Compute the 2^{28} initial states obtained by varying these bits and inverting the first three rounds of the permutation.

We obtain 2^{28} values that sum to zero, since their sum is the 28th-order derivative with respect to three inverse rounds. Their images also sum to zero, since they are the 28th-order derivative with respect to three forwards rounds (although the images are unknown, and need not be computed).

The method works whenever a function can be written as the composition of two low-degree functions. As explained in [BDPA10], the proposed technique is slightly more efficient than previous methods, for finding (here) zero-sums of 2^{28} elements.

4. First order differential analysis

In this section, we analyze the differential properties of the Hamsi round transformations and show how to find high-probability differential paths for up to six rounds. Since we use XOR differences in our analysis, the differential propagation is deterministic in the message expansion and in the linear layer based on the L transform. However, the propagation of differences through the Sbox layer is probabilistic and depends on the actual values of the input. To maximize the differential probability of a differential path, we try to minimize the number of active Sboxes during the path search.

4.1 Differential properties of the Sbox

The differential distribution table (DDT) of the 4-bit Hamsi Sbox S is given in Table A.2 (Appendix 7). Note that about half the differential transitions are impossible. The probabilities of the non-zero differentials are either 2^{-2} or 2^{-3}. In our approach, besides minimizing the number of active Sboxes, we thus try to minimize the number of probability-2^{-3} differentials.

4.2 Differential properties of the linear transform L

The linear transform L has on average good diffusion properties, that is, a few differences in the input lead to many differences in the output. Additionally, each bit of L contributes to one of the 128 Sboxes in each round. To minimize the number of active Sboxes, we thus need to minimize number of differences in L. The Hamming weight (HW) of a difference is a good heuristic to measure the quality of a differential path. In the following, we first analyze the difference propagation through the linear layer for differences with HW one.

If we introduce a single input difference at bit position i in one input word, the HW of the output differences depends on the position and word of the input difference. In Table 1 and Table 2 give the HW of the output difference for each of the 128 single bit input differences.

We observe that for some specific words and bit positions, the resulting HW can be quite small. This happens if one or more differences are removed by the shift operation. More specifically, the branch number of L is only 3, so certain 1-bit input differences lead to only a 2-bit output difference, and vice versa. Table 1 and Table 2 show the worst case of diffusion, that is, the output HW for a multiple-bit input difference can be upper bounded by summing the corresponding table entries. However, when inserting many differences in several input words, some bit differences might erase each other, thus lowering the overall HW.

Table 1. Hamming weight of output differences if a single difference is introduced at one input word of the 128-bit linear transformation $(a', b', c', d') = L(a, b, c, d)$ of Hamsi in *forward* direction. The total and word-wise Hamming weight of the output difference is given depending on the bit position i and input word of the input difference.

Difference in input word	Position i of input difference	Total HW of output diff.	HW of output diff. in				Conditions (mod 32)
			a'	b'	c'	d'	
a	16,17	3	2	1	-	-	$i+13>28, i+14>24$
	18	4	2	1	1	-	$i+13>28, i+14\leq 24$
	11...15	6	3	1	1	1	$i+13\leq 28, i+14>24$
	else	7	3	1	2	1	$i+13\leq 28, i+14\leq 24$
b	24...30	2	1	1	-	-	$i+1>24$
	else	3	1	1	1	-	$i+1\leq 24$
c	21...27	6	2	1	2	1	$i+4>24$
	else	7	2	1	3	1	$i+4\leq 24$
d		3	1	-	1	1	

Table 2. Hamming weight of input differences if a single difference is introduced at one output word of the 128-bit linear transformation $(a', b', c', d') = L(a, b, c, d)$ of Hamsi in *backward* direction. The total and word-wise Hamming weight of the input difference is given depending on the bit position i and output word of the output difference.

Difference in output word	Position i of output difference	Total HW of input diff.	HW of input diff. in				Conditions (mod 32)
			a	b	c	d	
a'	2...4	2	1	1	-	-	$i+27>28$
	else	3	1	1	-	1	$i+27\leq 28$
b'	28...31	3	1	2	-	-	$i>28, i>24$
	25...28	4	1	2	-	1	$i\leq 28, i>24$
	never	6	1	3	1	1	$i>28, i\leq 24$
	else	7	1	3	1	2	$i\leq 28, i\leq 24$
c'		3	-	1	1	1	
d'	29...31	4	1	-	1	2	$i>28$
	else	5	1	-	1	3	$i\leq 28$

4.3 Near-collisions for the compression function

Using our observations on the differential properties of Hamsi's Sbox and linear transform, we first searched manually for high-probability paths leading to near-collisions for the compression function, given some difference in the chaining value.

Previous work by Nikolic reported near collisions [Nik09] on $(256-25)$ bits with 14 differences in the chaining value; work by Wang et al. reported [WWJW09] near collisions on $(256-23)$ bits with 16 differences. Below we present near collisions on $(256-25)$ bits with only six differences in the chaining value, using the differential path in Table 3.

The differential path in Table 3 is followed with probability 2^{-26} under standard uniformity and independence assumptions. However, for the IV defined in [KÖ9b] the path is followed with probability 2^{-23}. This is because of the

Table 3. Differential path for three rounds of Hamsi with probability 2^{-26}.

It.	Sbox input				Sbox output				Prob.
1	00000000	00000000	00020000	00000002	00000000	00000000	00000000	00000002	8
	00004000	00000000	00000000	00000000	00004000	00000000	00000000	00000000	
	00000000	00000000	00020000	00000002	00000000	00000000	00000000	00000000	
	00004000	00000000	00000000	00000000	00000000	00000000	00020000	00000000	
2	00000000	00000000	00000000	00080000	00000000	00000000	00000000	00000000	3
	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00080000	
	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	
	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00080000	
3	80000000	00000000	02000000	00000000	00000000	00000000	00000000	04100000	15
	00000000	00000000	00000000	00100000	80020000	00000000	02010000	04100000	
	00020000	00000000	00010000	00000000	00020000	00000000	00000000	00000000	
	00000000	00000000	00000000	04000000	80000000	00000000	02010000	00000000	
End	00000000	80400800	00000000	10C130C0					
	00040105	00000000	04020000	08000000					
	00020400	A040A0A2	00000000	10004000					
	00000040	08000000	00820801	00000000					

condition put by the two fixed bits in each Sbox. These probabilities were verified experimentally.

Finally, note that the near collisions also result in other 4-sums: for example, for the IV h_0 specified in [KÖ9b], the IV h_1 obtained by applying the weight-6 initial difference in Table 3, and the message $M_1 =$ `C33BE456` and $M_2 =$ `C8D1B855`, we have:

1 A near collision between $H(h_0, M_1)$ and $H(h_1, M_1)$.

2 A near collision between $H(h_0, M_2)$ and $H(h_1, M_2)$.

3 A 4-sum $H(h_0, M_1) \oplus H(h_1, M_1) \oplus H(h_0, M_2) \oplus H(h_1, M_2) = 0$.

For random inputs of an "ideal" function, the latter equality is unlikely to hold with probability 2^{-23}, but rather with probability close to 2^{-256}.

In the following, we automate our search for high-probability differential paths. Our heuristic algorithm, described in the next section, produced good differential paths for up to six rounds of Hamsi.

4.4 Automated differential path search

As before, we search for differential paths with some difference in the input and output chaining value, and no difference in the input message. The resulting 6-round paths allow us to distinguish the output transform from random, as shown in Sect. 5.4.

Our primary heuristic is to minimise the HW of the differences in each round. To achieve that goal, we start with a very low HW (1 or 2 bit) difference in the middle of the path (at the start of round 3 for a 6-round search) and let the difference spread in both forward and backward directions. Additionally, we try to maximise the transition probabilities and randomize the search.

More precisely, our automated differential path starts from the input of the Sbox layer in round 3, forcing a 1-bit or 2-bit input difference on only one Sbox position i (among the 128 possible bit positions). We then choose one of the best differential transitions through the forward application of the Sbox and apply the linear layer on this new internal state. By best Sbox transitions, we mean the transitions that lead to a low HW after the application of the linear layer. To keep the search complexity feasible, we apply the L-layer to each active Sbox separately and use the sum of the HWs as an estimate of the total output HW at the end of each round. Since the path is sparse, the sum of HWs proves to be a good heuristic. We continue picking the best differential transitions for all the active Sbox positions until the end of the fifth round of the output function of Hamsi. As the final output HW of the difference does not influence the path complexity, we optimise for transition probabilities in the last round, and pick the most probable differential Sbox transitions (not the ones minimizing the HW). Finally, we apply the very last linear layer to obtain the full path.

The backward computation is done analogously in the middle rounds, applying the linear layer backward and picking the best backward differential transitions for all active Sboxes. In the first round (the last round when computing backward) we impose additional restrictions in order to fulfill constraints on the message expansion.

As we force no difference in the message input of the compression function, we expect the 256-bit expanded message word to contain no difference at all. Hence, in the first round we only allow Sbox transitions where the difference in the expanded message bits is zero. Note that the probabilities of the first-round transitions do not affect the complexity of the path, as long as they are different from 0. Indeed, in the first round we can use the freedom of the chaining input to fulfill the conditions on the Sboxes and we expect the complexity cost of this first round to be negligible.

In order to increase our chances to obtain a good trail, we randomized the search with several parameters. First, we randomized the first 1-bit or 2-bit perturbation introduction in the output of round 3, as well as its position i among the 128 Sbox locations. Furthermore, we are also randomizing the Sbox transitions when several candidates are equally good. Finally, another improvement has been incorporated in our implementation: after having found a potentially interesting 6-round candidate, we recompute the forward search by allowing more differential transitions through the Sbox. Said in other words, after having placed ourselves in an interesting differential paths subspace, we look in the neighborhood if better ones exist.

Our heuristic search revealed that after three rounds in both backward and forward directions, the diffusion of Hamsi is not sufficient to avoid high-probability differential paths and we can find a differential path with a rather low total HW and good probability. We were able to construct a 6-round differential path with a relatively high probability, which is used to distinguish the the whole Hamsi output transformation in the following section.

5. Non-randomness of the ouput transformation

5.1 The differential path

The best 6-round path produced by our randomized search program is depicted in Table 4. We can find an input pair (chaining values and messages) conforming to this path with a probability of 2^{-206}. Note that in the first round we have a probability of 2^{-58} for a random message and a random chaining value. However, we can fix a suitable message (see below), and choose a valid chaining value bit-by-bit such that the desired output difference is guaranteed. This means that we can find a conforming input pair to the differential path with a complexity of about 2^{148}.

5.2 First round and message expansion

In the first iteration, active Sboxes impose conditions on the expanded message: for a given non-zero Sbox differential, only one or two pairs of values of the corresponding two expanded message bits are possible. Since we have only 32 degrees of freedom in the message, we need to keep the number of active Sboxes in the first round low. To improve the probability of finding a suitable message candidate, we can vary the differences in the chaining values, whenever several input differences lead to the same output difference of the first Sbox layer. These relaxable differential Sbox transitions are listed in Table 5. In our path, five of the 23 active Sboxes of the first iteration are relaxable. In total, we have only nine Sboxes with two constraints on the message bits; 12 Sboxes with one constraint on the message; and two S-Boxes with a "half" constraint on the message (three of four bit pairs are possible). Therefore, we expect to find $2^{32-2\times9-12} \cdot \left(\frac{3}{4}\right)^2 \approx 2$ messages satisfying the relaxed first round differential. In practice, we found one such message using the constants of permutation P and three messages using the constants of the output permutation P_f. Table B.1 in Appendix A.2 lists these message words together with example chaining values that satisfy the differential path for up to four rounds. Note that finding conforming message words can be done in

Table 4. Differential path for six rounds of Hamsi with probability 2^{-148}.

It.	Sbox input				Sbox output				Prob.
start					00000000	00000000	84004880	4081C400	
					2C020018	000045C0	00000000	00000000	
					00000000	00000000	84024880	4081C400	
					28020018	000045C0	00000000	00000000	
1	00000000	00000000	84004880	4081C400	04000000	00000000	04000000	40818000	(58)
	2C020018	000045C0	00000000	00000000	28020018	000040C0	04020000	00000000	
	00000000	00000000	84024880	4081C400	00000018	00004100	00000800	00804000	
	28020018	000045C0	00000000	00000000	04020000	000004C0	80024880	00004400	
2	00000000	00000000	00000000	00010000	00000000	00000000	00000000	00010000	17
	30000010	00000080	00000000	00000080	30000000	00000000	00000000	00000080	
	30000010	00000080	00000000	00010080	00000010	00000000	00000000	00000000	
	00000000	00000000	00000000	00000000	00000000	00000080	00000000	00000000	
3	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	3
	20000000	00000000	00000000	00000000	20000000	00000000	00000000	00000000	
	20000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	
	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	
4	00000000	00000000	00000000	00000008	40000000	00000000	00000000	00000000	5
	40000000	00000000	00000000	00000000	40000000	00000000	00000000	00000008	
	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	
	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000008	
5	04038000	00000000	00000200	00000010	80000000	00001000	00000000	00200410	33
	80000000	00001000	00000000	00000010	04038002	00001000	00000801	00000000	
	00000002	00000000	00000a01	00000000	00000000	00000000	00000000	00000000	
	00000000	00000000	00000000	00200400	84038002	00000000	00000a01	00200400	
6	08420002	F8022900	00000000	30821140	08830144	A0022100	0C051080	10C01000	90
	0903000C	00000000	04001002	00000000	0181014C	58A04845	0C051082	22406340	
	00000000	A0A26145	00041080	12807200	01800148	58A04845	08011002	22406340	
	01C0014A	00000000	08051082	10420000	00400002	58000800	00040080	20020140	
End	CD9F7546	362513EA	56FE147F	85F6B1E1					
	8D0682FD	F100928A	B44C3D06	18A0D101					
	B8871BEA	70315A82	4819C14B	26257026					
	A1DD0199	40072022	8329356A	A744E830					

2^{32} by exhaustive search. The complexity to find chaining values such that the first four rounds of the path are satisfied is about 2^{25}, since we can fulfill the conditions in the first round deterministically.

5.3 Last round and truncation

In order to improve the probability of the differential path, we consider truncated differentials in the last application of the Sbox. Namely, we relax

Table 5. Relaxable differential transitions for the first round of the Hamsi Sbox. The first table shows the possible input differences that give the same output if 1, 4 and 5 are the only possible Sbox input differences. The second table shows the same possibilities if 2, 8, and 10 are the only possible Sbox input differences. For each underlined transition two message pairs are possible, while for the other transitions only one message pair is possible.

Desired output	1 a	2 b	3 ab	4 c	5 ac	6 bc	7 abc	8 d	9 ad	10 bd	11 abd	12 cd	13 acd	14 bcd	15 abcd
Possible input					1 5	$\underline{4}$ 5				1 $\underline{4}$	1 $\underline{4}$	1 5			

Desired output	1 a	2 b	3 ab	4 c	5 ac	6 bc	7 abc	8 d	9 ad	10 bd	11 abd	12 cd	13 acd	14 bcd	15 abcd
Possible input			$\underline{2}$ 8		$\underline{2}$ 8				$\underline{2}$ 8						$\underline{2}$ 8

the Sbox transitions by fixing some bits in the output difference, while letting the remaining bits vary. Since the "a"-bits and "c" bits diffuse faster through the linear layer (see Table 1), we chose to fix these bits in the output of each Sbox. Amongst four different truncated output differences (?0?0, ?0?1, ?1?0 and ?1?1), we chose, for each input, the output difference with the highest probability. Table 6 lists the relaxed input-output transitions for the Sbox.

In Appendix B.1, Table C.1 gives the truncated 6th round differential. Relaxing the Sbox transitions increases the probability of the last round to $2^{-61.8}$, giving a total path complexity $2^{-120.8}$. At the same time, since the "wild card" bits are chosen to have low diffusion, the difference is still fixed in 180 bits of the chaining value. Thus, we obtain a distinguisher by observing the difference in these output bits.

Table 6. Relaxed differential transitions for the last round of the Hamsi Sbox. The table shows the chosen set of output differences for each given input difference. Underlined transitions have probability 2^{-2}, while the other transitions have probability 2^{-3}.

input	1 a	2 b	3 ab	4 c	5 ac	6 bc	7 abc	8 d	9 ad	10 bd	11 abd	12 cd	13 acd	14 bcd	15 abcd
output	12 $\underline{14}$	$\underline{3}$ $\underline{9}$	$\underline{1}$ 9	$\underline{10}$	$\underline{1}$ 3	2 8	$\underline{4}$ 12	5 $\underline{7}$ $\underline{13}$ 15	$\underline{8}$ 10	2 8 $\underline{10}$	$\underline{1}$ 9	$\underline{11}$	$\underline{1}$ 3	$\underline{7}$ $\underline{13}$	$\underline{2}$ 10
mask	11?0	?0?1	?001	1010	00?1	?0?0	?100	?1?1	10?0	?0?0	?001	1011	00?1	?1?1	?010

5.4 Distinguishing the output transformation

To distinguish the output transformation of Hamsi we use the concept of differential q-multicollision introduced by Biryukov et al. in the cryptanalysis of AES-256 [KBN09] and applied to the SHA-3 candidate SIMD in [MN09]. Originally, differential q-multicollision have been applied to a block cipher but can be easily adapted to a random function. A differential q-multicollision for a random (compression) function $f(H, M)$ is a set of two differences ΔH, ΔM and q pairs $(H_1, M_1), (H_2, M_2), \ldots, (H_q, M_q)$ such that:

$$\begin{aligned}
&f(H_1, M_1) \oplus f(H_1 \oplus \Delta H, M_1 \oplus \Delta M) = \\
&f(H_2, M_2) \oplus f(H_2 \oplus \Delta H, M_2 \oplus \Delta M) = \\
&\ldots \\
&f(H_q, M_q) \oplus f(H_q \oplus \Delta H, M_q \oplus \Delta M)
\end{aligned}$$

The generic complexity to find differential q-multicollision for a random function f with output size n is at least $q \cdot 2^{\frac{q-2}{q+2} \cdot n}$ evaluations of f.

In the case of Hamsi-256, the function f is the output transformation, the message difference ΔM is zero and the output size is $n = 256$. The generic complexity to find differential q-multicollision should be $q \cdot 2^{\frac{q-2}{q+2} \cdot 256}$ and we get for $q = 8$ a generic complexity of $2^{156.1}$. Using our differential path of Section 5.1, we get for $q = 8$ a complexity of $8 \cdot 2^{148} = 2^{151}$. Hence, for $q \geq 8$ we can distinguish the output transfomation of Hamsi from a random function, since we expect to find a q-multicollision approximately 32 times faster than for an ideal transform.

Due to the relaxed conditions, we only fix a truncated difference in 180 output bits and hence, we get $n = 180$. In this case, the generic complexity for $q = 11$ is $q \cdot 2^{\frac{q-2}{q+2} \cdot 180} = 2^{128.1}$. Using the relaxed differential path, we get $q \cdot 2^{120.8} = 2^{124.3}$ and hence, can distinguish the output transfomation of Hamsi from a random function for $q \geq 11$.

6. Conclusion

We investigated the resistance of the 256-bit version of the second round SHA-3 candidate Hamsi against differential and higher-order differential attacks.

Using higher-order analysis, we showed that the 3-round compression function of Hamsi has suboptimal algebraic degree. Using this observation, we provided sets of four related IV's such that the outputs of the compression function obtained with a given fixed message sum to zero. We also presented a set of 2^{19} message words such that the output chaining values, using any fixed IV, sum to zero. The latter result indicates that the compression function of Hamsi, when seen as a function of message words, does not reach the expected maximal degree 27. As an application, we note that the low degree makes the standalone compression function existentially forgeable in the message authentication setting.

Further, we constructed high-probability differential paths for the 3-round compression function to demonstrate a near-collision on $(256 - 25)$ bits with only six differences in the input chaining value. We have also developed a technique for building low-weight, high-probability differential paths for more rounds of Hamsi. Our best differential path for six rounds has probability 2^{-148}, much higher than expected for a random function. Additionally, we gave a truncated differential on 180 output bits with probability $2^{-120.8}$. These are the first results on six rounds of Hamsi, allowing us to distinguish the full output transformation from a random function using differential q-multicollisions.

Although none of our findings directly leads to an attack on the hash algorithm, they indicate that the buildings blocks of Hamsi exhibit nonrandom behavior. We expect our work to serve as a starting point for future analysis of Hamsi.

In order to prevent more serious attacks, we recommend increasing the number of rounds in the output transformation as a precaution. While the current specification does not include performance figures for the 8-round alternative, this change is only expected to noticeably affect the speed of hashing short messages.

7. Acknowledgements

Emilia Käsper thanks the Computer Laboratory of the University of Cambridge for hosting her.

This work was supported in part by the European Commission through the ICT Programme under Contract ICT-2007-216646 ECRYPT II. Emilia Käsper was also supported by the IAP–Belgian State–Belgian Science Policy BCRYPT and the IBBT (Interdisciplinary institute for BroadBand Technology) of the Flemish Government.

References

[BAK98] Eli Biham, Ross J. Anderson, and Lars R. Knudsen. Serpent: A new block cipher proposal. In Serge Vaudenay, editor, *FSE*, volume 1372 of *LNCS*, pages 222–238. Springer, 1998.

[BDPA10] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. Note on zero-sum distinguishers of Keccak-f. NIST mailing list, 2010.

[Ber07] Daniel J. Bernstein. Better price-performance ratios for generalized birthday attacks. In *SHARCS*, 2007. `http://cr.yp.to/papers.html#genbday`.

[BM97] Mihir Bellare and Daniele Micciancio. A new paradigm for collision-free hashing: Incrementality at reduced cost. In Walter Fumy, editor, *EUROCRYPT*, volume 1233 of *LNCS*, pages 163–192. Springer, 1997.

[KÖ9a] Özgül Kücük. The hash function Hamsi. Submission to NIST (Round 2), January 2009.

[KÖ9b] Özgül Kücük. Reference implementation of Hamsi (round 2). Submission to NIST, January 2009.

[KBN09] Dmitry Khovratovich, Alex Biryukov, and Ivica Nikolić. Distinguisher and related-key attack on the full AES-256. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 231–249. Springer, 2009.

[Knu94] Lars R. Knudsen. Truncated and higher order differentials. In Bart Preneel, editor, *FSE*, volume 1008 of *LNCS*, pages 196–211. Springer, 1994.

[Lai92] Xuejia Lai. Higher order derivatives and differential cryptanalysis. In Richard Blahut, Daniel Costello Jr., Ueli Maurer, and Thomas Mittelholzer, editors, *Communications and Cryptography*, pages 227–233. Kluwer, 1992.

[MN09] Florian Mendel and Tomislav Nad. A Distinguisher for the Compression Function of SIMD-512. In Bimal K. Roy and Nicolas Sendrier, editors, *INDOCRYPT*, volume 5922 of *LNCS*, pages 219–232. Springer, 2009.

[Nik09] Ivica Nikolić. Near collisions for the compression function of Hamsi-256. CRYPTO rump session, 2009.

[NIS07] NIST. Announcing request for candidate algorithm nominations for a new cryptographic hash algorithm (SHA-3) family. Federal Register Notice, 72(112), November 2007. `http://csrc.nist.govgroups/ST/hash/documents/FR_Notice_Nov07.pdf`.

[SAB09] Bhupendra Singh, Lexy Alexander, and Sanjay Burman. On algebraic relations of Serpent S-boxes. Cryptology ePrint Archive, Report 2009/038, 2009.

[Wag99] David Wagner. The boomerang attack. In Lars R. Knudsen, editor, *FSE*, volume 1636 of *LNCS*, pages 156–170. Springer, 1999.

[Wag02] David Wagner. A generalized birthday problem. In *Proceedings of Advances in Cryptology - CRYPTO 2002*, volume 2442 of *LNCS*, pages 288–303. Springer, 2002.

[WLF$^+$05] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the hash functions MD4 and RIPEMD. In Ronald Cramer, editor, *EUROCRYPT'05*, volume 3494 of *LNCS*, pages 1–18. Springer, 2005.

[WWJW09] Meiqin Wang, Xiaoyun Wang, Keting Jia, and Wei Wang. New pseudo-near-collision attack on reduced-round of Hamsi-256. Cryptology ePrint Archive, Report 2009/484, 2009.

[WY05] Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In Ronald Cramer, editor, *EUROCRYPT'05*, volume 3494 of *LNCS*, pages 19–35. Springer, 2005.

[WYY05a] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In Victor Shoup, editor, *Proceedings of CRYPTO 2005*, volume 3621 of *LNCS*, pages 17–36. Springer, 2005.

[WYY05b] Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin. Efficient collision search attacks on SHA-0. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 1–16. Springer, 2005.

Appendix: The Sbox of Hamsi

Table A.1. The Hamsi Sbox in decimal basis.

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$S[x]$	8	6	7	9	3	12	10	15	13	1	14	4	0	11	5	2

Table A.2. The differential distribution table (DDT) of the Hamsi Sbox in decimal basis.

In \ Out	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	2	0	2	0	0	2	2	2	0	4	2
2	0	0	0	4	0	4	0	0	0	4	0	0	0	0	0	4
3	0	4	2	0	0	0	2	0	0	2	0	0	2	0	2	2
4	0	0	0	0	0	0	4	0	0	0	4	4	0	4	0	0
5	0	4	0	2	2	2	2	0	2	0	0	0	2	0	0	0
6	0	0	2	2	2	2	0	0	2	2	0	0	0	0	2	2
7	0	0	0	0	4	2	0	2	0	0	2	2	2	0	0	2
8	0	0	0	2	0	2	0	4	0	2	0	0	0	4	0	2
9	0	0	0	2	0	0	0	2	4	2	2	2	2	0	0	0
10	0	0	2	0	2	0	4	0	2	0	4	0	0	0	2	0
11	0	4	0	0	2	0	2	0	2	2	0	0	2	0	0	2
12	0	0	2	0	2	0	0	0	2	0	0	4	0	4	2	0
13	0	4	2	2	0	2	2	0	0	0	0	0	2	0	2	0
14	0	0	2	0	2	0	0	4	2	0	0	0	0	4	2	0
15	0	0	4	2	0	0	0	2	0	2	2	2	2	0	0	0

Appendix: Conforming Message Pairs

Table B.1. The messages we found satisfying the relaxed first round differential for permutation P and P_f together with the corresponding difference we put on the chaining value before the application of the first Sbox. These differences vary over the five relaxable Sbox transition. We also give example chaining values satisfying the differential path up to four rounds. The chaining value and input difference should be read from top-left to top-right, then bottom-left to bottom-right.

Permutation	Message				Difference on chaining value				Chaining value			
P	FD	1A	35	83	84024880	4081C400	28020018	000045C0	0FE53B63	12DED071	D5A7C265	F886F53E
					84004080	40818400	2C020018	000045C0	7F00EA2A	F0F14CCC	0F6A6528	8E235B01
P_f	53	1C	BD	E2	84004880	4081C400	28020018	000045C0	B443BB07	149683BE	DD71AD95	931F6D84
					84024880	4081C400	2C020018	000045C0	4AFBF940	631CCFF0	576A371A	76618746
P_f	68	FF	2B	71	84004880	4081C400	2C020018	000045C0	1C03A81D	7155CABB	BBF7EFC8	EE22F7CD
					84024080	4001C400	28020018	000045C0	CAFBF940	231D8FF0	34457281	81A20735
P_f	A6	ED	03	6C	84004880	4081C400	2C020018	000045C0	BB63500E	B6E6863F	FB3F6527	512A60DA
					84024880	40818400	28020018	000045C0	4EF9B140	631C8BF0	323C56EC	012D9A36

Appendix: Truncated differential path

Table C.1. Truncated differential for the last round of Hamsi with probability $2^{-61.8}$. The state is printed column by column; "?" marks an unknown difference.

It.	Sbox input	Sbox output	Prob.
6 (col 0)	00001000010000100000000000000010 00001001000000110000000000001100 00000000000000000000000000000000 00000001110000000000000101001010	00001000100000110000000101000100 0000000???00000?0000000?0?00???0 00000000100000000000000101000000 0000?00??10000??0000000?0?00??10	61.8
6 (col 1)	11111000000000100010100100000000 00000000000000000000000000000000 10100000101000100110000101000101 00000000000000000000000000000000	10100000000000100010000100000000 ?????000101000?001?0?00?01000101 01011000000000000000100000000000 01011000101000000100100001000101	
6 (col 2)	00000000000000000000000000000000 00000100000000000001000000000010 00000000000001000001000010000000 00001000000000101000100001000010	00001100000000101000100001000000 0000??000000010?000?0000100000?0 00001000000000001000100000000000 0000??000000010?000?0000100000?0	
6 (col 3)	00110000100000100001000101000000 00000000000000000000000000000000 00010010100000000111001000000000 00010000010000100000000000000000	00010000110000000001000000000000 00??0010??0000?0011?001?0?000000 00100000010000000000000101000000 001000100?0000100110001101000000	
7 (col 0)	11001011?011?1?00?11000101?????0 0000100110??0001?0010?111?1?1010		
7 (col 1)	00??01111?1?0?01??010010?11??1?? ??11101??0??00001??110?01?0?11??		
7 (col 2)	01??0010?1111?100?0?01001111??11 01?011000101?001?100?1?10?001101		
7 (col 3)	1???010?0?0101001???10????110000 ?010010?00??00011??10?1011?0?01?		

PAPER E

Analysis of the MQQ Public Key Cryptosystem

Jean-Charles Faugère, Rune Steinsmo Ødegård, Ludovic Perret and Danilo Gligoroski

In Proceedings of the 9th International Conference on Cryptology and Network Security

Kuala Lumpur, Malaysia. December 12-14, 2010.

ANALYSIS OF THE MQQ PUBLIC KEY CRYPTOSYSTEM

Jean-Charles Faugère
SALSA *Project - INRIA (Centre Paris-Rocquencourt)*
UPMC, Univ Paris 06 - CNRS, UMR 7606, LIP6
104, avenue du Président Kennedy 75016 Paris, France
jean-charles.faugere@inria.fr

Rune Steinsmo Ødegård
Centre for Quantifiable Quality of Service in Communication Systems.
Norwegian University of Science and Technology
Trondheim, Norway
rune.odegard@q2s.ntnu.no

Ludovic Perret
SALSA *Project - INRIA (Centre Paris-Rocquencourt)*
UPMC, Univ Paris 06 - CNRS, UMR 7606, LIP6
104, avenue du Président Kennedy 75016 Paris, France
ludovic.perret@lip6.fr

Danilo Gligoroski
Department of Telematics at the Norwegian University of Science and Technology in Trondheim, Norway
danilog@item.ntnu.no

Abstract MQQ is a multivariate public key cryptosystem (MPKC) based on multivariate quadratic quasigroups and a special transform called "*Dobbertin transformation*" [GMK08b]. The security of MQQ, as well as any MPKC, reduces to the difficulty of solving a non-linear system of equations easily derived from the public key. In [MDBW09], it has been observed that that the algebraic systems obtained are much easier to solve that random non-linear systems of the same size. In this paper we go one step further in the analysis of MQQ. We explain why systems arising in MQQ are so easy to solve in practice. To do so, we consider the so-called the degree of regularity; which is the exponent in the complexity of a Gröbner basis computation. For MQQ systems, we show that this degree is bounded from above by a small constant. This is due to the fact that the complexity of solving the MQQ system is the minimum complexity of solving just one quasigroup block or solving the Dobbertin transformation. Furthermore, we show that the degree of regularity of the Dobbertin transformation is bounded from above by the same constant as the bound observed on MQQ system. We then investigate the strength of a tweaked MQQ system where the input of the Dobbertin transformation is replaced with random linear equations. It appears that the degree of regularity of this tweaked system varies both with the size of the quasigroups and the number of variables. We conclude that if a suitable replacement for the Dobbertin transformation is found, MQQ can possibly be made strong enough to resist pure Gröbner attacks for adequate choices of quasigroup size and number of variables.

Keywords: multivariate cryptography, Gröbner bases , public-key, multivariate quadratic quasigroups, algebraic cryptanalysis

1. Introduction

The use of polynomial systems in cryptography dates back to the mid eighties with the design of Matsumoto and Imai [MI88], later followed by numerous other proposals. Two excellent surveys on the current state of proposals for multivariate asymmetric cryptosystems have been written by Wolf and Preneel [WP05] as well as Billet and Ding [BD09]. Basically the current proposals can be classified into four main categories, some of which combine features from several categories: Matsumoto-Imai like schemes [Pat96, PGC98], Oil and Vinegar like schemes [Pat97, KHPG99], Stepwise Triangular Schemes [Sha93, GC00] and Polly Cracker Schemes [dVMPT09]. In addition Gligoroski et al. have proposed a fifth class of trapdoor functions based on multivariate quadratic quasigroups [GMK08b].

As pointed out in [BD09], it appears that most multivariate public-key cryptosystems (MPKC) suffer from obvious to less obvious weaknesses. Some attacks are specific and focus on one particular variation and breaks it due to specific properties. One example is the attack of Kipnis and Shamir against the Oil and Vinegar scheme [KS98]. However, most attacks use general purpose algorithms that solve multivariate system of equations. Generic algorithms to solve this problem are exponential in the worst case, and solving random system of algebraic equations is also known to be difficult (i.e. exponential) in the average case. However, in the case of multivariate public-key schemes the designer has to embed some kind of trapdoor function to enable efficient decryption and signing. To achieve this, the public-key equations are constructed from a highly structured system of equations. Although the structure is hidden, it can be exploited for instance via differential or Gröbner basis based techniques.

Using Gröbner basis [Buc65] is a well established and general method for solving polynomial systems of equations. The complexity of a Gröbner basis computation is exponential in the degree of regularity, which is the maximum degree of polynomials occurring during the computation [BFS04]. The first published attack on multivariate public-key cryptosystems using Gröbner basis is the attack by Patarin on the Matsumoto-Imai scheme [Pat95]. In this paper Patarin explains exactly why one is able to solve the system by using Gröbner bases. The key aspect is that there exists bilinear equations relating the input and output of the system [BD09]. This low degree relation between the input and the output means that only polynomials of a low degree will appear during the computation of the Gröbner basis. Consequently, the complexity of solving the system is bounded by this low degree.

Another multivariate cryptosystem which has been broken by Gröbner bases cryptanalysis is the MQQ public key block cipher [GMK08b]. The cipher was broken both by Gröbner bases and MutantXL independently in [MDBW09]. Given a ciphertext encrypted using the public key, the authors of [MDBW09] were able to compute the corresponding plaintext. However, the paper did not theoretically explain why the algebraic systems of MQQ are easy to solve in practice. In this paper we explain exactly why the MQQ cryptosystem is susceptible to algebraic cryptanalysis. This is of course interesting from a cryptanalysis perspective, but also from a design perspective. If we want to construct strong multivariate cryptographic schemes we must understand why the weak schemes have been broken.

1.1 Organisation of the paper

This paper is organized as follows. In Section 2 we give an introduction to multivariate quadratic quasigroups. After that we describe the MQQ public key cryptosystem. In Section 3 we give a short introduction to the theory of Gröbner bases and reiterate the generic complexity of computing such bases. In Section 4 we show that the degree of regularity of MQQ systems is bounded from above by a small constant. We then explain this characteristic by looking at the shape of the inner system. In Section 5 we further elaborate on the weaknesses of MQQ, and investigate if some tweaks can make the system stronger. Finally, Section 6 concludes the paper.

2. Description of the MQQ public key cryptosystem

In this section we give a description of the multivariate quadratic quasigroup public key cryptosystem [GMK08b]. The system is based on previous work by Gligoroski and Markovski who introduced the use of quasigroup string processing in cryptography [Mar03, MGB99].

2.1 Multivariate quadratic quasigroups

We first introduce the key building block of the MQQ PKC, namely multivariate quadratic quasigroups. For a detailed introduction to quasigroups in general, we refer the interested reader to [Smi07].

DEFINITION 1 *A quasigroup is a set Q together with a binary operation $*$ such that for all $a, b \in Q$ the equations $\ell * a = b$ and $a * r = b$ have unique solutions ℓ and r in Q respectively. A quasigroup is said to be of order n if there are n elements in the set Q.*

Let $(Q, *)$ be a quasigroup of order 2^d, and β be a bijection from the quasigroup to the set of binary strings of length d, i.e

$$\begin{array}{rccc} \beta : & Q & \to & GF(2^d) \\ & a & \mapsto & (x_1, \dots, x_d) \end{array} \tag{1}$$

Given such a bijection, we can naturally define a vector valued Boolean function

$$\begin{array}{rccc} *_{vv} : & GF(2^d) \times GF(2^d) & \to & GF(2^d) \\ & (\beta(a), \beta(b)) & \mapsto & \beta(a * b) \end{array} \tag{2}$$

Now let $\beta(a*b) = (x_1, \dots, x_d) *_{vv} (x_{d+1}, \dots, x_{2d}) = (z_1, \dots, z_d)$. Note that each z_i can be regarded as a $2d$-ary Boolean function $z_i = f_i(x_1, \dots, x_{2d})$, where each $f_i : GF(2^d) \to GF(2)$ is determined by $*$. This gives us the following lemma [GMK08b].

LEMMA 2 *For every quasigroup $(Q, *)$ of order 2^d and for each bijection $\beta : Q \to GF(2^d)$ there is a unique vector valued Boolean function $*_{vv}$ and d uniquely determined $2d$-ary Boolean functions $f_1, f_2, \dots, f_d$ such that for each $a, b, c \in Q$:*

$$\begin{array}{c} a * b = c \\ \Updownarrow \\ (x_1, \dots, x_d) *_{vv} (x_{d+1}, \dots, x_{2d}) = (f_1(x_1, \dots, x_{2d}), \dots, f_d(x_1, \dots, x_{2d})). \end{array} \tag{3}$$

This leads to the following definition for multivariate quadratic quasigroups.

DEFINITION 3 *([GMK08b]) Let $(Q, *)$ be a quasigroup of order 2^d, and let $f_1, \dots, f_d$ be the uniquely determined Boolean functions under some bijection β. We say that the quasigroup is a multivariate quadratic quasigroup (MQQ)* *of type $Quad_{d-k}Lin_k$ (under β) if exactly $d-k$ of the corresponding polynomials f_i are of degree 2 and k of them are of degree 1, where $0 \leq k \leq d$.*

Gligoroski et al. [GMK08b] mention that quadratic terms might cancel each other. By this we mean that some linear transformation of $(f_i)_{1\leq i\leq n}$ might result in polynomials where the number of linear polynomials is larger than k, while the number of quadratic polynomials is less than $d-k$. Later Chen et al. [CKG10] have shown that this is more common than previously expected. In their paper they generalize the definition of MQQ above to a family which is invariant by linear transformations, namely:

Definition 4 *Let $(Q,*)$ be a quasigroup of order 2^d, and let $f_1,\dots,f_d$ be the unique Boolean functions under some bijection β. We say that the quasigroup is a multivariate quadratic quasigroup (MQQ) of strict type $Quad_{d-k}Lin_k$ (under β), denoted by $Quad^s_{d-k}Lin^s_k$, if there at most $d-k$ quadratic polynomials in $(f_i)_{1\leq i\leq d}$ whose linear combination do not result in a linear form.*

Chen et al. also improved Theorem 2 from [GMK08b] which gives a sufficient condition for a quasigroup to be MQQ. We restate this result below.

Theorem 5 *Let $\mathbf{A}_1 = [f_{ij}]_{d\times d}$ and $\mathbf{A}_2 = [g_{ij}]_{d\times d}$ be two $d\times d$ matrices of linear Boolean expressions with respect to $x_1,\dots,x_d$ and $x_{d+1},\dots,x_{2d}$ respectively. Let $\mathbf{c}$ be a binary column vector of d elements. If $det(\mathbf{A}_1) = det(\mathbf{A}_2) = 1$ and*

$$\mathbf{A}_1\cdot(x_{d+1},\dots,x_{2d})^T + (x_1,\dots,x_d)^T = \mathbf{A}_2\cdot(x_1,\dots,x_d)^T + (x_{d+1},\dots,x_{2s})^T, \tag{4}$$

then the vector valued Boolean operation $(x_1,\dots,x_d) *_{vv} (x_{d+1},\dots,x_{2d}) =$

$$\mathbf{B}_1\mathbf{A}_1\cdot(x_{d+1},\dots,x_{2d})^T + \mathbf{B}_2\cdot(x_1,\dots,x_d)^T + \mathbf{c} \tag{5}$$

defines a quasigroup $(Q,*)$ of order 2^d which is MQQ for any two non-singular Boolean matrices $\mathbf{B}_1$ and $\mathbf{B}_2$

In addition Chen et al. [CKG10] proved that no MQQ as in Theorem 5 can be of strict type $\text{Quad}^s_d\text{Lin}^s_0$. This result uncovered a possible weakness in [GMK08b] as the proposed scheme used 6 quasigroups of type $\text{Quad}_5\text{Lin}_0$.

Notice that the vector valued Boolean function defining the MQQ in Theorem 5 have no terms of the form x_ix_j with $i,j\leq d$ or $i,j>d$. This means that if we set the first or the last half of the variables to a constant, we end up with only linear terms in the MQQ. It is still an open question if there exists MQQ that are not as in Theorem 5.

The MQQs used in this paper have been produced using the algorithm provided in Appendix 6. The algorithm is based on the paper [CKG10], and produces MQQs that are more suitable for encryption since they are guaranteed to be of strict type $\text{Quad}^s_{d-k}\text{Lin}^s_k$ for $0<k\leq d$.

2.2 The Dobbertin bijection

In addition to MQQs, [GMK08b] also uses a bijection introduced by Dobbertin in [Dob98]. Dobbertin proved that the following function, in addition to being multivariate quadratic over $GF(2)$, is a bijection in $GF(2^{2r+1})$:

$$\begin{array}{rccc} D_r: & GF(2^{2r+1}) & \to & GF(2^{2r+1}) \\ & x & \mapsto & x^{2^{r+1}+1} + x^3 + x \end{array} \tag{6}$$

2.3 A Public Key Cryptosystem Based on MQQ

We are now ready to describe the public key cryptosystem presented by Gligoroski et al. in [GMK08b]. Let $N = nd$ be the desired number of variables $(x_1, \dots, x_N)$, and let $\{*^1_{vv}, \dots, *^k_{vv}\}$ be a collection of MQQs of size 2^d represented as $2d$-ary vector valued Boolean functions. The public key is constructed as follows.

Algorithm *MQQ public key construction.*

1. Set $\mathbf{X} = [x_1, \dots, x_N]^T$. Randomly generate an $N \times N$ non-singular Boolean matrix $\mathbf{S}$, and compute $\mathbf{X} \leftarrow \mathbf{S} \cdot \mathbf{X}$.
2. Randomly choose a n-tuple $I = \{i_1, \dots, i_n\}$, where $i_j \in \{1, \dots, k\}$. The tuple I will decide which MQQ, $*^{i_j}_{vv}$, to use at each point of the quasigroup transformation.
3. Represent $\mathbf{X}$ as a collection of vectors of length d, $\mathbf{X} = [X_1, \dots, X_n]^T$. Compute $\mathbf{Y} = [Y_1, \dots, Y_n]^T$ where $Y_1 = X_1, Y_2 = X_1 *^{i_1}_{vv} X_2$, and $Y_{j+1} = X_j *^{i_j}_{vv} X_{j+1}$ for $j = 1, \dots, n-1$.
4. Set $\mathbf{Z}$ to be the vector of all the linear terms of $Y_1, \dots, Y_n$. Here Y_1 will be all linear terms, while each Y_j has between 1 and k linear terms depending on the type $\mathrm{Quad}^s_{d-k}\mathrm{Lin}^s_k$ of MQQ used. Transform $\mathbf{Z}$ with one or more Dobbertin bijections of appropriate size. For example if $\mathbf{Z}$ is of size 27 we can use one Dobbertin bijection of dimension 27, three of dimension 9, or any other combination summing up to 27. Finally, set $\mathbf{W} \leftarrow \mathrm{Dob}(\mathbf{Z})$.
5. Replace the linear terms of $\mathbf{Y} = [Y_1, \dots, Y_n]^T$ with the terms in $\mathbf{W}$. Randomly generate an $N \times N$ non-singular Boolean matrix $\mathbf{T}$, and compute $\mathbf{Y} \leftarrow \mathbf{T} \cdot \mathbf{Y}$
6. **return** the public key $\mathbf{Y}$. The private key is $\mathbf{S}, \mathbf{T}, \{*^1_{vv}, \dots, *^k_{vv}\}$ and I.

3. Gröbner bases

This section introduces the concept of Gröbner bases as well as a complexity bound to compute such bases. We refer to (for instance) [CLO05] for basic definitions, and a more detailed description of the concepts.

Let $\mathbb{K}$ be a field and $\mathbb{K}[x_1, \dots, x_N]$ be the polynomial ring over $\mathbb{K}$ in the variables $x_1, \dots, x_N$. Recall that a *monomial* in a collection of variables is a product $x^\alpha = x_1^{\alpha_1} \cdots x_N^{\alpha_N}$ where $\alpha_i \geq 0$. Let $>$ be an admissible *monomial order* on $\mathbb{K}[x_1, \dots, x_n]$. The most common example of such ordering is the *lexicographical order* where $x^\alpha > x^\beta$ if in the difference $\alpha - \beta \in \mathbb{Z}^N$, the leftmost nonzero entry is positive. Another frequently encountered order is the *graded reverse lexicographical* order where $x^\alpha > x^\beta$ iff $\sum_i \alpha_i > \sum_i \beta_i$ or $\sum_i \alpha_i = \sum_i \beta_i$ and in the difference $\alpha - \beta \in \mathbb{Z}^N$ the rightmost nonzero entry is negative. For different monomial orderings Gröbner bases hold specific theoretical properties and show different practical behaviors. Given a monomial order $>$, the *leading term* of a polynomial $f = \sum_\alpha c_\alpha x^\alpha$, denoted $LT_>(f)$, is the product $c_\alpha x^\alpha$ where x^α is the largest monomial appearing in f in the ordering $>$.

Definition 6 *([CLO05]) Fix a monomial order $>$ on $\mathbb{K}[x_1, \dots, x_N]$, and let $I \subset \mathbb{K}[x_1, \dots, x_N]$ be an ideal. A Gröbner basis for I (with respect to $>$) is a finite collection of polynomials $G = \{g_1, \dots, g_t\} \subset I$ with the property that for every nonzero $f \in I$, $LT_>(f)$ is divisible by $LT_>(g_i)$ for some i.*

Let

$$f_1(x_1, \dots, x_N) = \cdots = f_m(x_1, \dots, x_N) = 0 \tag{7}$$

by a system of m polynomials in N unknowns over the field $\mathbb{K}$. The set of solutions in $\mathbb{K}$, which is the *algebraic variety*, is defined as

$$V = \{(z_1, \dots, z_N) \in k | f_i(z_1, \dots, z_N) = 0 \forall 1 \leq i \leq m\} \tag{8}$$

In our case we are interested in the solutions of the MQQ system, which are defined over $GF(2)$.

Proposition 7 *([FJ03]) Let G be a Gröbner basis of $\langle f_1, \dots, f_m, x_1^2 - x_1, \dots, x_n^2 - x_n \rangle \subset GF(2)[x_1, \dots, x_n]$. Then the following holds:*

1 $V = \emptyset$ (no solution) iff $G = \{1\}$.

2 V has exactly one solution iff $G = \{x_1 - a_1, \dots, x_n - a_n\}$ where $a_i \in GF(2)$. Then $(a_1, \dots, a_n)$ is the solution in $GF(2)$ of the algebraic system $f_1 = \cdots = f_m = 0$.

It is clear that as we are solving systems over $GF(2)$ we have to add the field equations $x_i^2 = x_i$ for $i = 1, \dots, N$. This means that we have to compute Gröbner bases of $m + N$ polynomials and N variables. This is quite helpful, since the more equations you have, the more able you are to compute Gröbner bases [FJ03].

3.1 Complexity of Computing Gröbner Bases

Historically, the concept of Gröbner bases, together with an algorithm for computing them, was introduced by Bruno Buchberger in his PhD-thesis [Buc65]. Buchberger's algorithm is implemented in many computer algebra systems. However, in the last decade, more efficient algorithms for computing Gröbner bases have been proposed. Most notable are Jean-Charles Faugère's F_4[Fau99] and F_5 [Fau02] algorithms. In this paper we have used the Magma [MAG] 2.16-1 implementation of the F_4 algorithm on a 4 core Intel Xeon 2.93GHz computer with 128GB of memory.

The complexity of computing a Gröbner basis of an ideal I depends on the maximum degree of the polynomials appearing during the computation. This degree, called *degree of regularity*, is the key parameter for understanding the complexity of a Gröbner basis computation [BFS04]. Indeed, the complexity of the computation is exponential in the degree of regularity D_{reg}, more precisely the complexity is:

$$\mathcal{O}(N^{\omega D_{\text{reg}}}), \tag{9}$$

which basically correspond to the complexity of reducing a matrix of size $\approx N^{D_{\text{reg}}}$. Here $2 < \omega \leq 3$ is the "linear algebra constant", and N the number of variables of the system. Note that D_{reg} is also a function of N, where the relation between D_{reg} and N depends on the specific system of equations. This relation is well understood for regular (and semi-regular) systems of equations [Bar04, BFS04, BFS02, BFSY05]. However, as soon as the system has some kind of structure, this degree is much more difficult to predict. In some particular cases, it is actually possible to bound the degree of regularity (see the works done on HFE [FJ03, GJS06]). But this is a hard task in general.

As already pointed out, the degree of regularity is abnormally small for algebraic systems occuring in MQQ. This fact explains the weakness observed in [MDBW09]. In this paper, we go one step further in the security analysis by explaining why the degree of regularity is so small for MQQ.

Note that the degree of regularity is related to the ideal $I = \langle f_1, \ldots, f_m \rangle$ and not the equations $f_1, \ldots, f_m$ themselves. In particular, for any non-singular matrix T, the degree of regularity of $[f'_1, \ldots, f'_m]^t = T \cdot [f_1, \ldots, f_m]^t$ is similar to the degree of regularity of $[f_1, \ldots, f_m]$. More generally, we can assume that this degree is generically (i.e. with high probability) invariant for a random (invertible) linear change of variables, and an (invertible) combination of the polynomials. These are exactly the transformations performed to mask the MQQ structure. Note that such a hypothesis has already been used for instance in [GJS06].

Table 1. Results for MQQ-(30,60,120,180). Computed with Magma 2.16-1's implementation of the F_4 algorithm on a Intel Xeon 2.93GHz quad core computer with 128GB of memory.

Variables	D_{reg}	Solving Time (s)	Memory (b)
30	3	0,06	15,50
60	3	1,69	156,47
120	3	379,27	4662,00
180	3	4136,31	28630,00

4. Why MQQ is Susceptible to Algebraic Cryptanalysis

In [MDBW09], MQQ systems with up to 160 variables was broken using MutantXL (the same result has also been obtained independently with F_4). The most important point made by [MDBW09] is that the degree of regularity is bounded from above by 3. This is much lower than a random system of quadratic equations where the degree of regularity increases linearly with the number of variables N. Indeed, for a random system it holds that D_{reg} is asymptotically equivalent to $\frac{N}{11.114}$ [BFS02]. The authors of [MDBW09] observed that this low degree is due to the occurrence of many new low degree relations during the computation of a Gröbner basis. In Section 4.2, we will explain in detail how the very structure of the MQQ system results in the apparance of the low degree relations. First, however, we will show that same upper bound on the degree of regularity is obtained using the improved quasigroups described in Section 2.1.

4.1 Experimental Results on MQQ

To test how the complexity of Gröbner bases computation of MQQ systems is related to the number of variables, we constructed MQQ systems in $30, 60, 120$ and 180 variables following the procedure described in Section 2.3. In this construction we used 17 MQQs of strict type $\text{Quad}_8^s\text{Lin}_2^s$ and Dobbertin bijections over different extension fields of dimension 7 and 9 respectively. We then tried to compute the plaintext given a ciphertext encrypted with the public key. The results of this test are presented in Table 1. From the table we see that the degree of regularity does not increase with the number of variables, but remains constant at 3. This means breaking the MQQ system is only polynomial in the number of variables. Once again, this is not the behaviour of a random system of equations, for which the degree of regularity increases

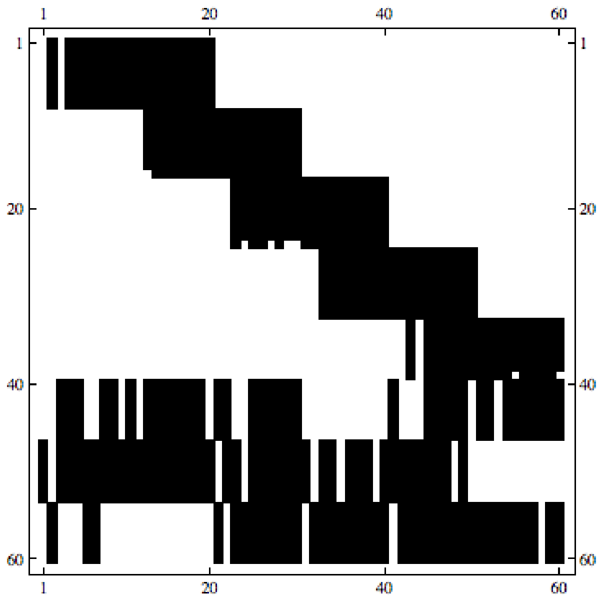

Figure 1. Shape of 60 variable MQQ public key system without the use of S and T transformations. The black color means that the corresponding variables is used in the equation. The system was constructed using 4 MQQs of type $\mathrm{Quad}_8^s\mathrm{Lin}_2^s$, one MQQ of type $\mathrm{Quad}_7^s\mathrm{Lin}_3^s$, and 3 Dobbertin bijections defined over 3 different extension fields of dimension 7.

linearly with the number of variables, and the solving time therefore increases exponentially. We explain the reason of such difference in the next section.

4.2 Shape of the MQQ system

The non-random behavior described above can be explained by considering the shape of the "unmasked" MQQ system. By unmasked we mean the MQQ system without the linear transformations S and T. As already explained in Section 3.1, the maximum degree of the polynomials occurring in the computation of a Gröbner basis is invariant under the linear transformation S and T.

In Figure 1 we show which variables appear in each equation for an unmasked MQQ system of 60 variables. The staircase shape comes from the cascading use of quasigroups, while the three blocks of equations at the bottom are from the Dobbertin bijection of size 7. Obviously, a random multivariate system would use all 60 variables in all equations. For this instance of MQQ, only $\frac{1}{3}$ of the variables are used in each quasigroup and about $\frac{2}{3}$ is used in each block of the Dobbertin transformation.

Now assume that the Gröbner basis algorithm somewhere during the calculation has found the solution for one of the quasigroup blocks $Y_j = X_j *_{vv}^{i_j} X_{j+1}$. Due to the cascading structure of the MQQ system, the variables of X_j are used in the block $Y_{j-1} = X_{j-1} *_{vv}^{i_{j-1}} X_j$ and the variables of X_{j+1} are used in the block $Y_{j+1} = X_{j+1} *_{vv}^{i_{j+1}} X_{j+2}$. In Section 2.1 we showed that if we set the first or the last half of the variables of an MQQ to constant, all equations become linear. This means that if we have solved the block Y_j, the equations of the blocks Y_{j-1} and Y_{j+1} becomes linear. The blocks Y_{j-1} and Y_{j+1} can then be solved easily. This gives a solution for the variables X_{j-1} and X_{j+2}, which again makes the equations in the blocks Y_{j-2} and Y_{j+2} linear. Continuing like this we have rapidly solved the whole system.

Similarly, assume the Gröbner basis has solved the Dobbertin blocks at some step. This gives us the solution to all the variables in X_1 which makes the first quasigroup block $Y_1 = X_1 *_{vv}^{i_1} X_2$ linear. Solving this gives us the first half of the equations of the block Y_2 and so on. As a conclusion, solving a MQQ system is reduced to either solving just one block of quasigroup equations, or solving the Dobbertin transformation. The security of solving an MQQ system is therefore the minimum complexity between solving the Dobbertin transformation or one MQQ block.

5. Weaknesses of MQQ

The goal of this part is to determine the weakest part of the system; the Dobbertin transformation or the quasigroup transformation. We first look closer at the Dobbertin block of equations. Since these equations constitutes a square system of equations, we expect them to be easier to solve then the quasigroup block of equations, which is an undetermined system of equations.

5.1 The Dobbertin transformation

Recall that the Dobbertin transformation is a bijection over $GF(2^{2r+1})$ defined by the function $D_r(x) = x^{2^{r+1}+1} + x^3 + x$. For any r, we can view this function as $2r+1$ Boolean equations in $2r+1$ variables. Using Magma 2.16-1's implementation of the F_4 algorithm[1], we experimentally computed the degree of regularity for solving this system of equations for $r = 2, \ldots, 22$. We observed that the degree of regularity was 3 for all computed instances.

[1]The computer used was 4 processor Intel Xeon 2.93GHz computer with 128GB of memory

Therefore the Dobbertin transformation can be easily solved by a Gröbner basis computation. In addition we learn that tweaking the MQQ system by increasing the size of the extension field, over which the transformation is defined, will have no effect on strengthening the system.

Proving mathematically (if true) that the degree of regularity of $D_r(x)$ is constant at 3 for all r is difficult. We can, however, explain why the degree of regularity is low for all practical r. Let $\mathbb{K} = \mathbb{F}_q$ be a field of q elements, and let $\mathbb{L}$ be an extension of degree n over $\mathbb{K}$. Recall that an HFE polynomial f is a low-degree polynomial over $\mathbb{L}$ with the following shape:

$$f(x) = \sum_{\substack{0\le i,j\le n \\ q^i+q^j\le d}} a_{i,j}x^{q^i+q^j} + \sum_{\substack{0\le k\le n \\ q^k\le d}} b_k x^{q^k} + c, \qquad (10)$$

where $a_{i,j}, b_k$ and c all lie in $\mathbb{L}$. The maximum degree d of the polynomial has to be chosen such that factorization over $\mathbb{L}$ is efficient [BFJT09]. Setting $q = 2$ and $n = 2r + 1$ we notice that the Dobbertin transformation is actually an HFE polynomial, $D_r(x) = x^{2^{r+1}+2^0} + x^{2^1+2^0} + x^{2^0}$. This is very helpful since a lot of work has been done on the degree of regularity for Gröbner basis compuation of HFE polynomials [FJ03, BFJT09]. Indeed, it has been proved that the degree of regularity for HFE polynomial of degree d is bounded from above by $\log_2(d)$ [FJ03, FMRS08]. For Dobbertin's transformation this means the degree of regularity is bounded from above by $r + 1$ at least.

However, since the coefficients of the Dobbertin transformation all lie in $GF(2)$, we can give an even tighter bound on the degree of regularity. Similarly to the weak-key polynomials in [BFJT09], the Dobbertin transformation commutes with the Frobenius automorphism and its iterates $F_i(x) : x \mapsto x^{2^i}$ for $0 \le i \le n$, namely

$$D_r \circ F_i(x) = F_i \circ D_r(x). \qquad (11)$$

Thus $D_r(x) = 0$ implies that $F_i \circ D_r(x) = 0$. This means for each i we can add the $2r+1$ equations over $GF(2)$ corresponding to the equation $D_r \circ F_i(x) = 0$ over $GF(2^{2r+1})$ to the ideal. However, many of these equations are similar. Actually, we have that F_i and F_j are similar if and only if $gcd(i, 2r + 1) = gcd(j, 2r + 1)$ [BFJT09]. Worst case scenario is when $2r + 1$ is prime. The Frobenius automorphism then gives us (only) $2(2r + 1)$ equations in $2r + 1$ variables. From [BFS03] we have the following formula for the degree of regularity for a random system of multivariate equations over $GF(2)$ when the number of equations m is a multiple of the number of variables N. For $m = N(k + o(1))$ with $k > 1/4$ the degree of regularity is

$$\frac{D_{\text{reg}}}{N} = \frac{1}{2} - k + \frac{1}{2}\sqrt{2k^2 - 10k - 1 + 2(k+2)\sqrt{k(k+2)}} + o(1). \qquad (12)$$

Setting $k = 2$ we get $D_{\text{reg}} = -\frac{3}{2} + \frac{1}{2}\sqrt{-13 + 16\sqrt{2}} \cdot (2r + 1) \approx 0.051404 \cdot (2r + 1) = 0.102808 \cdot r + 0.051404$. Note that the degree of regularity cannot be smaller then 3. This means we have $max(3, 0.102808 \cdot r + 0.051404)$ as an upper bound for a *random* multivariate system with the same number of equations and variables as the Dobbertin transformation. This provides a good indication that the degree of regularity for Dobbertin (which is not random at all) should be small, as observed in the experiments, and even smaller than a regular HFE polynomial.

5.2 The Quasigroup Transformation

To get an idea how strong the quasigroup transformation is, we performed some experiments where we replaced the input of the Dobbertin transformation by random linear equations. This means that solving a Dobbertin transformation block will no longer make all the equations in the first quasigroup transformation linear. The result of our experiment on this special MQQ system where the linear equations are perfectly masked is listed in Table 2. Note that the degree of regularity of 5 is still too small to prevent Gröbner bases attacks. What is important is how the degree of regularity increases when we increase different parameters. From the table it appears that both the quasigroup size and the number of variables have an effect on the degree of regularity. This tells us that if we replace the Dobbertin transformation with a stronger function, the MQQ system can possibly be made strong enough to resist pure Gröbner attacks for adequate choices of quasigroup size and number of variables.

6. Conclusion

We further explained the results of [MDBW09] by showing that the degree of regularity for MQQ systems are bounded from above by a small constant. Therefore even MQQ systems with large number of variables can easily be broken with Gröbner bases cryptanalysis. The main result of this paper is an explanation of the underlying reason for this abnormal degree of regularity. We demonstrated how the complexity of solving MQQ systems with Gröbner bases is equal to the minimum of the complexity of solving the Dobbertin transformation and the complexity of solving one MQQ block. Furthermore, our experimental data showed that the degree of regularity for solving the Dobbertin transformation is bounded from above by 3, the same as the bound on the MQQ system. These experimental results were also explained mathematically. A natural interpretation of the results of our investigation is that

Table 2. Effects of quasigroup size and the Dobbertin transformation on the observed degree of regularity for different MQQ. D_{reg} is the observed degree of regularity of normal MQQ systems, while D^*_{reg} is the observed degree of regularity for the same system where the input to Dobbertin has been replaced with random linear equations.

Variables	Quasigroup size	Quasigroups type	Dobbertin	D_{reg}	D^*_{reg}
30	2^5	4 $\mathrm{Quad}^s_3\mathrm{Lin}^s_2$ and 1 $\mathrm{Quad}^s_2\mathrm{Lin}^s_3$	7,9	3	3
	2^{10}	2 $\mathrm{Quad}^s_8\mathrm{Lin}^s_2$	7,7	3	4
40	2^5	5 $\mathrm{Quad}^s_3\mathrm{Lin}^s_2$ and 2 $\mathrm{Quad}^s_2\mathrm{Lin}^s_3$	7,7,7	3	4
	2^{10}	3 $\mathrm{Quad}^s_8\mathrm{Lin}^s_2$	7,9	3	4
	2^{20}	1 $\mathrm{Quad}^s_{17}\mathrm{Lin}^s_3$	7,7,9	3	4
50	2^5	9 $\mathrm{Quad}^s_3\mathrm{Lin}^s_2$	7,7,9	3	3
	2^{10}	4 $\mathrm{Quad}^s_8\mathrm{Lin}^s_2$	9,9	3	4
60	2^5	11 $\mathrm{Quad}^s_3\mathrm{Lin}^s_2$	9,9,9	3	3
	2^{10}	4 $\mathrm{Quad}^s_8\mathrm{Lin}^s_2$ and 1 $\mathrm{Quad}^s_7\mathrm{Lin}^s_3$	7,7,7	3	5
	2^{20}	1 $\mathrm{Quad}^s_{18}\mathrm{Lin}^s_2$ and 1 $\mathrm{Quad}^s_{17}\mathrm{Lin}^s_3$	7,9,9	3	5

the Dobbertin transformation employed is a serious weakness in the MQQ system.

From a design point of view, we also showed that if Dobbertin's transformation is replaced with an ideal function – which perfectly hides the linear parts of the system – the degree of regularity varies with the size of the quasigroups and the number of variables. We conclude that if a suitable replacement for Dobbertin's transformation is found, MQQ can possibly be made strong enough to resist pure Gröbner attacsk for adequate choices of quasigroup size and number of variables. This remains an interesting open problem.

References

[Bar04] Magali Bardet. *Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie.* PhD thesis, Université de Paris VI, 2004.

[BD09] Olivier Billet and Jintai Ding. Overview of cryptanalysis techniques in multivariate public key cryptography. In Massimiliano Sala, Teo Mora, Ludovic Perret, Shojiro Sakata, and Carlo Traverso, editors, *Gröbner bases, coding and cryptography*, pages 263–283. Springer Verlag, 2009.

[BFJT09] Charles Bouillaguet, Pierre-Alain Fouque, Antoine Joux, and Joana Treger. A family of weak keys in HFE (and the corresponding practical key-recovery). Cryptology ePrint Archive, Report 2009/619, 2009.

[BFS02] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. Complexity study of Gröbner basis computation. Technical report, INRIA, 2002. `http://www.`

`inria.fr/rrrt/rr-5049.html`.

[BFS03] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. Complexity of Gröbner basis computation for semi-regular overdetermined sequences over F2 with solutions in F2. Technical report, Institut national de recherche en informatique et en automatique, 2003.

[BFS04] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In *Proc. International Conference on Polynomial System Solving (ICPSS)*, pages 71–75, 2004.

[BFSY05] Magali Bardet, Jean-Charles Faugère, Bruno Salvy, and Bo-Yin Yang. Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems. In *Proc. of MEGA 2005, Eighth International Symposium on Effective Methods in Algebraic Geometry*, 2005.

[Buc65] Bruno Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal.* PhD thesis, Leopold-Franzens University, 1965.

[CKG10] Yanling Chen, Svein Johan Knapskog, and Danilo Gligoroski. Multivariate quadratic quasigroups (MQQs): Construction, bounds and complexity. In *Inscrypt*, 6th International Conference on Information Security and Cryptology. Science Press of China, October 2010.

[CLO05] David Cox, John Little, and Donal O'Shea. *Using Algebraix Geometry.* Springer, 2005.

[Dob98] Hans Dobbertin. One-to-one highly nonlinear power functions on GF(2^n). *Appl. Algebra Eng. Commun. Comput.*, 9(2):139–152, 1998.

[dVMPT09] Francoise Levy dit Vehel, Maria Grazia Marinari, Ludovic Perret, and Carlo Traverso. A survey on polly cracker system. In Massimiliano Sala, Teo Mora, Ludovic Perret, Shojiro Sakata, and Carlo Traverso, editors, *Gröbner bases, coding and cryptography*, pages 263–283. Springer Verlag, 2009.

[Fau99] Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases (F_4). *Journal of Pure and Applied Algebra*, 139(1-3):61–88, June 1999.

[Fau02] Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5). In *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*, New York, 2002. ACM.

[FJ03] Jean-Charles Faugère and Antoine Joux. Algebraic cryptanalysis of Hidden Field Equation (HFE) cryptosystems using Gröbner bases. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *LNCS*, pages 44–60. Springer, 2003.

[FMRS08] Pierre-Alain Fouque, Gilles Macario-Rat, and Jacques Stern. Key Recovery on Hidden Monomial Multivariate Schemes. In *Advances in Cryptology - EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 19–30. Springer, 2008.

[GC00] Louis Goubin and Nicolas T. Courtois. Cryptanalysis of the TTM cryptosystem. In *Advances of Cryptology, Asiacrypt 2000*, volume 1976 of *LNCS*, pages 44–57. Springer, 2000.

[GJS06] Louis Granboulan, Antoine Joux, and Jacques Stern. Inverting HFE Is Quasipolynomial. In *Advances in Cryptology - CRYPTO 2006*, volume 4117 of *LNCS*, pages 345–356. Springer, 2006.

[GMK08] Danilo Gligoroski, Smile Markovski, and Svein Johan Knapskog. Multivariate quadratic trapdoor functions based on multivariate quadratic quasigroups. In *MATH'08: Proceedings of the American Conference on Applied Mathematics*, pages 44–49, Stevens Point, Wisconsin, USA, 2008. World Scientific and Engineering Academy and Society (WSEAS).

[KHPG99] Aviad Kipnis, Hamarpe St. Har Hotzvim, Jacques Patarin, and Louis Goubin. Unbalanced Oil and Vinegar Signature Schemes. In *Advances in Cryptology - EUROCRYPT 1999*, volume 5479 of *LNCS*, pages 206–222. Springer, 1999.

[KS98] Aviad Kipnis and Adi Shamir. Cryptanalysis of the Oil & Vinegar Signature Scheme. In *CRYPTO '98: Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology*, volume 1462 of *LNCS*, pages 257–266, London, UK, 1998. Springer-Verlag.

[MAG] MAGMA. High performance software for algebra, number theory, and geometry — a large commercial software package. url:`http://magma.maths.usyd.edu.au`.

[Mar03] Smile Markovski. Quasigroup string processing and applications in cryptography. In *Proc. 1-st Inter. Conf. Mathematics and Informatics for industry MII 2003, 1416 April, Thessaloniki*, pages 278–290, 2003.

[MDBW09] Mohamed Saied Mohamed, Jintai Ding, Johannes Buchmann, and Fabian Werner. Algebraic attack on the MQQ public key cryptosystem. In *CANS '09: Proceedings of the 8th International Conference on Cryptology and Network Security*, pages 392–401, Berlin, Heidelberg, 2009. Springer-Verlag.

[MGB99] 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.

[MI88] Tsutomu Matsumoto and Hideki Imai. Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In *Advances in Cryptology – EUROCRYPT 1988*, volume 330 of *LNCS*, pages 419–453. Springer–Verlag, 1988.

[Pat95] Jacques Patarin. Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt–88. In Don Coppersmith, editor, *Advances in Cryptology – CRYPT0'95*, volume 963 of *LNCS*, pages 248–261. Springer Berlin / Heidelberg, 1995.

[Pat96] Jacques Patarin. Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms. In *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 33–48. Springer-Verlag, 1996.

[Pat97] Jacques Patarin. The Oil & Vinegar signature scheme. In *Proceedings of Dagstuhl workshop on cryptography*, 1997.

[PGC98] Jacques Patarin, Louis Goubin, and Nicolas Courtois. C^*_{-+} and HM : Variations Around Two Schemes of T. Matsumoto and H. Imai. In Kazuo Ohta and Dingyi Pei, editors, *Advances in Cryptology – ASIACRYPT'98*, volume 1514 of *LNCS*, pages 35–50. Springer Berlin / Heidelberg, 1998.

[Sha93] Adi Shamir. Efficient signature schemes based on birational permutations. In *Proceedings of CRYPTO'93*, volume 773 of *LNCS*, pages 1–12. Springer-Verlag, 1993.

[Smi07] J. D. H. Smith. *An introduction to quasigroups and their representations.* Chapman & Hall/CRC, 2007.

[WP05] Christopher Wolf and Bart Preneel. Taxonomy of Public Key Schemes based on the problem of Multivariate Quadratic equations. IACR Eprint archive, 2005. url:`http://eprint.iacr.org/2005/077`.

Appendix: Algorithm for generating random MQQ

In this section we present the pseudo-code for how the MQQs used in this paper have been generated. The code was implemented in Magma.

Algorithm *MQQ algorithm*

```
1.  $n \leftarrow$ {size of quasigroup}
2.  $L \leftarrow$ {number of linear terms}
3.  if $L \leq 2$
4.    then $Q = n$
5.    else  $Q = n - L$
6.  CorrectDeg $\leftarrow$True
7.  while CorrectDeg
8.     do $A1 \leftarrow$IdentityMatrix($n$) ($*$ The identity matrix of size $n$ $*$)
9.        $X1 \leftarrow [x_1, \ldots, x_n]^T$
10.       $X2 \leftarrow [x_{n+1}, \ldots, x_{2n}]^T$
11.       for $i \leftarrow 1$ to $Q$
12.           do for $j \leftarrow i + 1$ to $n$
13.                 do for $k \leftarrow i + 1$ to $(n)$
14.                         $r \in_R \{0, 1\}$ ($*$ random element from the set {0,1} $*$)
15.                         $A1_{(i,j)} = A1_{(i,j)} + r * X1_k$
16.       $B \leftarrow$RandomNonSingularBooleanMatrix($n$) ($*$ Random non singular Boolean
          matrix of size $n$ $*$)
17.       $C \leftarrow$RandomBooleanVector(n) ($*$ Random Boolean vector of size $n$ $*$)
18.       $A1 \leftarrow B * A1$
19.       $X1 \leftarrow B * X1 + C$
20.       $L1 \leftarrow$RandomNonSingularBooleanMatrix($n$) ($*$ Random non singular Boolean
          matrix of size $n$ $*$)
21.       $L2 \leftarrow$RandomNonSingularBooleanMatrix($n$) ($*$ Random non singular Boolean
          matrix of size $n$ $*$)
22.       $A1 \leftarrow$LinTrans($A1, L1$) ($*$ Lineary transform the indeterminates of $A1$ according
          to $L1$ $*$)
23.       $X1 \leftarrow$LinTrans($X1, L1$) ($*$ Lineary transform the indeterminates of $X1$ according
          to $L1$ $*$)
24.       $X2 \leftarrow$LinTrans($X2, L2$) ($*$ Lineary transform the indeterminates of $X2$ according
          to $L2$ $*$)
25.       MQQ $\leftarrow A1 * X2 + X1$
26.       GBMQQ $\leftarrow$Gröbner(MQQ,2) ($*$ The truncated Gröbnerbasis of degree 2 under
          graded reverse lexicographical ordering. $*$)
27.       Deg $\leftarrow$ {number of linear terms in GBMQQ}
28.       if Deg$= L$
29.          then CorrectDeg $\leftarrow$False
30. return GBMQQ
```

PAPER F

MQQ-SIG. An Ultra-fast and provably CMA Resistant Digital Signature Scheme

Danilo Gligoroski,Rune Steinsmo Ødegård, Rune Erlend Jensen, Ludovic Perret, Jean-Charles Faugère, Svein Johan Knapskog and Smile Markovski.

In Proceedings of the 3rd International Conference on Trusted Systems, INTRUST 2011.

Beijing, China. November 27-29, 2011.

MQQ-SIG AN ULTRA-FAST AND PROVABLY CMA RESISTANT DIGITAL SIGNATURE SCHEME

Danilo Gligoroski
Department of Telematics at the Norwegian University of Science and Technology in Trondheim, Norway
danilog@item.ntnu.no

Rune Steinsmo Ødegård
Centre for Quantifiable Quality of Service in Communication Systems.
Norwegian University of Science and Technology
Trondheim, Norway
rune.odegard@q2s.ntnu.no

Rune Erlend Jensen
Centre for Quantifiable Quality of Service in Communication Systems.
Norwegian University of Science and Technology
Trondheim, Norway
runeerle@idi.ntnu.no

Ludovic Perret
SALSA *Project - INRIA (Centre Paris-Rocquencourt)*
UPMC, Univ Paris 06 - CNRS, UMR 7606, LIP6
104, avenue du Président Kennedy 75016 Paris, France
ludovic.perret@lip6.fr

Jean-Charles Faugère
SALSA *Project - INRIA (Centre Paris-Rocquencourt)*
UPMC, Univ Paris 06 - CNRS, UMR 7606, LIP6
104, avenue du Président Kennedy 75016 Paris, France
jean-charles.faugere@inria.fr

Svein Johan Knapskog
Centre for Quantifiable Quality of Service in Communication Systems.
Norwegian University of Science and Technology
Trondheim, Norway
knapskog@q2s.ntnu.no

Smile Markovski
"Ss Cyril and Methodius" University, Faculty of Natural Sciences and Mathematics, Institute of Informatics, P.O.Box 162, 1000 Skopje, MACEDONIA
smile@ii.edu.mk

Abstract We present MQQ-SIG, a signature scheme based on "*Multivariate Quadratic Quasigroups*". The MQQ-SIG signature scheme has a public key consisting of $\frac{n}{2}$ quadratic polynomials in n variables where $n = 160, 192, 224$ or 256. Under the assumption that solving systems of $\frac{n}{2}$ MQQ's equations in n variables is as hard as solving systems of random quadratic equations, we prove that in the random oracle model our signature scheme is CMA (Chosen-Message Attack) resistant.

From efficiency point of view, the signing and verification processes of MQQ-SIG are three orders of magnitude faster than RSA or ECDSA. Compared with other MQ signing schemes, MQQ-SIG has both advantages and disadvantages. Advantages are that it has more than three times smaller private keys (from 401 to 593 bytes), and the signing process is an order of magnitude faster than other MQ schemes. That makes it very suitable for implementation in smart cards and other embedded systems. However, MQQ-SIG has a big public key (from 125 to 512 Kb) and it is not suitable for systems where the size of the public key has to be small.

Keywords: Public Key Cryptography, Ultra-Fast Public Key Cryptography, Multivariate Quadratic Polynomials, Quasigroup String Transformations, Multivariate Quadratic Quasigroup

1. Introduction

Multivariate quadratic schemes (MQ schemes) are an active research area since their introduction more than 26 years ago in the papers of Matsumoto and Imai [IM86, MI88]. They have a lot of performance advantages over classical public key schemes based on integer factorization (RSA) and on the discrete logarithm problem in the additive group of points defined by elliptic curves over finite fields (ECC), but they have also one additional advantage: there are no known quantum algorithms that would break MQ schemes faster than generic brute force attacks.

We can say that MQ schemes can be generally divided in five types of schemes that conceptually differ in the construction of the nonlinear quadratic part of the scheme. There is a nice (but a little bit older survey from 2005) [WP05] that covers the first four classes of multivariate quadratic public key cryptosystems: MIA [IM86], STS [Sha93, Moh99, GC00], HFE [Pat96] and UOV [KHPG99].

The fifth scheme MQQ was introduced in [GMK08c, GMK08b] in 2008. MQQ is based on the theory of quasigroups and quasigroup string transformations. Since it had interesting performance characteristics, it immediately attracted the attention of cryptographers trying to attack it. It was first successfully cryptanalysed independently by Perret [Per08] using Gröbner basis approach, and Mohamed et al. using MutantXL [MDBW09]. Later, improved cryptanalysis by Faugère et al. in [FØPG10] explained exactly why the MQQ systems are so easy to solve in practice.

In this paper we describe a digital signature variant of MQQ (called MQQ-SIG). To thwart previous successful attacks, we propose to use the *minus modifier*, i.e. to remove some equations of the public key. More specifically, we remove $\frac{1}{2}$ of the public equations of the original MQQ public key algorithm. We also present numerical (experimental) evidence that gives us arguments to believe that Gröbner bases approach (and having in mind that MutantXL approach is equivalent) is ineffective in solving the remaining known equations.

Thus, based on the assumption that solving $\frac{n}{2}$ quadratic MQQ's equations with n variables is as hard as solving systems of random quadratic equations, we show that in the random oracle model our signature scheme is provably CMA resistant.

The properties of MQQ-SIG digital signature scheme can be briefly summarized as:

- In the random oracle model it is provably CMA resistant under the assumption that solving $\frac{n}{2}$ MQQ's quadratic equations with n variables is as hard as solving systems of random equations;
- Its conjectured security level is at least $2^{\frac{n}{2}}$;
- The length of the signature is $2n$ bits where ($n = 160, 192, 224$ or 256);
- The size of the private key is between 401 and 593 bytes.
- The size of the public key is between 125 and 512 Kb.
- In software, its signing speed is in the range of 300–3,500 times faster than the most popular public key schemes, and 5 to 20 times faster than

other multivariate quadratic schemes with equivalent security parameters;

- Its verification speed is comparable to the speed of other multivariate quadratic PKCs;
- In hardware, its signing or verification speed can be more than 10,000 times faster than the most popular public key schemes;
- In 8-bit MCUs, smart cards and RFIDs, it is hundreds or thousands times faster than the most popular public key signature schemes;

2. Preliminaries - Quasigroups and Multivariate Quadratic Quasigroups

Here we give a brief overview of quasigroups and quasigroup string transformations. A more detailed explanation can be found in [Bel67, DK74, Smi07].

Definition 1 *A quasigroup $(Q,*)$ is a groupoid satisfying the law*

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

This implies the cancelation laws $x*y = x*z \implies y = z$, $y*x = z*x \implies y = z$. Note also that the equations $a*x = b$, $y*a = b$ have unique solutions x, y for each $a, b \in Q$. Given a quasigroup $(Q,*)$ five so called "*parastrophes*" (or "*conjugate operations*") can be adjoint to $*$. Here, we use only two of them – denoted by $\backslash$ and $/$, – defined by

$$x * y = z \iff y = x \setminus z \iff x = z/y \tag{2}$$

Then $(Q,\backslash)$ and $(Q,/)$ are quasigroups too and the algebra $(Q,*,\backslash,/)$ satisfies the identities

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

Conversely, if an algebra $(Q,*,\backslash,/)$ with three binary operations satisfies the identities (3), then $(Q,*)$, $(Q,\backslash)$, $(Q,/)$ are quasigroups and (2) holds.

In what follows we will work with finite quasigroups of order 2^d i.e. where $|Q| = 2^d$. To define a multivariate quadratic PKC for our purpose, we will use the following result.

Lemma 2 ([GMK08c, GMK08b]) *For every quasigroup $(Q,*)$ of order 2^d and for each bijection $Q \to \{0, 1\ldots, 2^d-1\}$ there are a uniquely determined* *vector valued Boolean functions* $*_{vv}$ *and* d *uniquely determined* $2d$*-ary Boolean functions* $f_1, f_2, \ldots, f_d$ *such that for each* $a, b, c \in Q$

$$a * b = c \Longleftrightarrow$$
$$*_{vv}(x_1, \ldots, x_d, y_1, \ldots, y_d) = \big(f_1(x_1, \ldots, x_d, y_1, \ldots, y_d), \ldots, f_d(x_1, \ldots, x_d, y_1, \ldots, y_d)\big). \tag{4}$$

Recall that each k-ary Boolean function $f(x_1, \ldots, x_k)$ can be represented in a unique way by its algebraic normal form (ANF), i.e., as a sum of products $\text{ANF}(f) = \alpha_0 + \sum_{i=1}^{k} \alpha_i x_i + \sum_{1\le i<j\le k} \alpha_{i,j} x_i x_j + \sum_{1\le i<j<s\le k} \alpha_{i,j,s} x_i x_j x_s + \ldots$, where the coefficients $\alpha_0, \alpha_i, \alpha_{i,j}, \ldots$ are in the set $\{0, 1\}$ and the addition and multiplication are in the field $GF(2)$.

The ANFs of the functions f_i defined in Lemma 2 give us information about the complexity of the quasigroup $(Q, *)$ via the degrees of the Boolean functions f_i. In general, for a randomly generated quasigroup of order 2^d, $d \geq 4$, the degrees are higher than 2. Such quasigroups are not quadratic and thus are not suitable for our construction of multivariate quadratic PKC.

Definition 3 *A quasigroup* $(Q, *)$ *of order* 2^d *is called Multivariate Quadratic Quasigroup (MQQ) of type* $Quad_{d-k}Lin_k$ *if exactly* $d - k$ *of the polynomials* f_i *are of degree 2 (i.e., are quadratic) and* k *of them are of degree 1 (i.e., are linear), where* $0 \leq k < d$.

In [GMK08c, GMK08b] the authors give sufficient conditions a quasigroup to be a MQQ as well as an algorithm for finding MQQs up to the order of 2^5. That work was later extended in [CKG10] for constructing MQQs of order 2^d for any d. The common characteristic of the MQQs produced by those two methods is that the quasigroups are bilinear. Namely, the equations (4) describing a multivariate quadratic quasigroup $(Q, *)$ can be expressed in the following form:

$$\mathbf{A_1} \cdot (y_1, \ldots, y_d)^T + \mathbf{b_1} \equiv \mathbf{A_2} \cdot (x_1, \ldots, x_d)^T + \mathbf{b_2} \tag{5}$$

where $\mathbf{A_1} = [f_{ij}]_{d\times d}$ is a $d \times d$ matrix and $\mathbf{b_1} = [u_i]_{d\times 1}$ is a $d \times 1$ vector of linear Boolean expressions of the variables $x_1, \ldots, x_d$, while $\mathbf{A_2} = [g_{ij}]_{d\times d}$ is a $d \times d$ matrix and $\mathbf{b_2} = [v_i]_{d\times 1}$ is a $d \times 1$ vector of linear Boolean expressions of the variables $y_1, \ldots, y_d$.

A Multivariate Quadratic Quasigroup (MQQ) $*$ of order 2^d used in MQQ-SIG can be described shortly by the following expression:

$$\mathbf{x} * \mathbf{y} \equiv \mathbf{B} \cdot \mathbf{U}(\mathbf{x}) \cdot \mathbf{A_2} \cdot \mathbf{y} + \mathbf{B} \cdot \mathbf{A_1} \cdot \mathbf{x} + \mathbf{c} \tag{6}$$

where $\mathbf{x} = (x_1, \ldots, x_d)$, $\mathbf{y} = (y_1, \ldots, y_d)$, the matrices $\mathbf{A_1}$, $\mathbf{A_2}$ and $\mathbf{B}$ are nonsingular of size $d \times d$ in $GF(2)$, the vector $\mathbf{c}$ is a random d-dimensional vector with elements in $GF(2)$ and all of them are generated by a uniformly random process. The matrix $\mathbf{U}(\mathbf{x})$ is an upper triangular matrix with all diagonal elements equal to 1, and the elements above the main diagonal are linear expressions of the variables of $\mathbf{x} = (x_1, \ldots, x_d)$. It is computed by the following expression:

$$\mathbf{U}(\mathbf{x}) = I + \sum_{i=1}^{d-1} \mathbf{U}_i \cdot \mathbf{A_1} \cdot \mathbf{x}, \tag{7}$$

where the matrices $\mathbf{U}_i$ have all elements 0 except the elements in the rows from $\{1, \ldots, i\}$ that are strictly above the main diagonal. Those elements can be either 0 or 1 generated by a uniformly random process.

Additionally, we require the quasigroups to satisfy the following two conditions:

$$\forall i \in \{1, \ldots, d\},\ Rank(\mathbf{B}_{f_i}) \geq 2d - 4, \tag{8a}$$

$$\exists j \in \{1, \ldots, d\},\ Rank(\mathbf{B}_{f_j}) = 2d - 2 \tag{8b}$$

where the matrices $\mathbf{B}_{f_i}$ are $2d \times 2d$ Boolean matrices defined from the expressions f_i as

$$\mathbf{B}_{f_i} = [b_{j,k}],\ b_{j,d+k} = b_{d+k,j} = 1, \text{ iff } x_j y_k \text{ is a term in } f_i. \tag{9}$$

The reasons why we need the additional conditions (8a) and (8b) will be explained in the beginning of the Section 5.

PROPOSITION 4 *For $d = 8$, a multivariate quadratic quasigroup that satisfies the conditions (6), ..., (9) can be encoded in a unique way with 81 bytes.* □

3. Description of the MQQ-SIG Digital Signature Scheme

Our scheme can be expressed as a $(\frac{1}{2})$ truncation of a typical multivariate quadratic system:

$$\mathbf{S} \circ P' \circ \mathbf{S}' : \{0,1\}^n \to \{0,1\}^n,$$

where $\mathbf{S}' = \mathbf{S} \cdot \mathbf{x} + \mathbf{v}$ (i.e. $\mathbf{S}'$ is a bijective affine transformation), $\mathbf{S}$ is a nonsingular linear transformation, and $P' : \{0,1\}^n \to \{0,1\}^n$ is a central bijective multivariate quadratic mapping defined in Table 1. It is graphically presented in Fig. 1.

The graphical presentation of the construction of the central mapping P' using the quasigroup operation $*$ is shown in Fig. 2, and its inverse P'^{-1} constructed with the parastrophe operations $\backslash$ and $/$ is shown in Fig. 3.

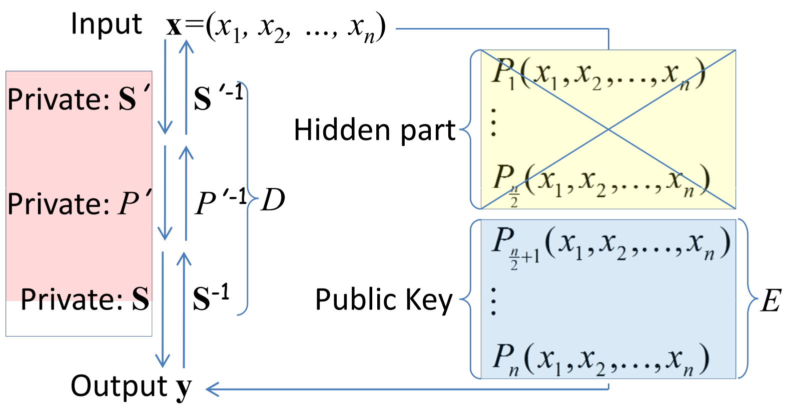
Input $\mathbf{x}=(x_1, x_2, \ldots, x_n)$
Private: $\mathbf{S}'$
$\mathbf{S}'^{-1}$
Private: P'
P'^{-1}
D
Private: $\mathbf{S}$
$\mathbf{S}^{-1}$
Output $\mathbf{y}$
Hidden part
$P_1(x_1,x_2,\ldots,x_n)$
$P_{\frac{n}{2}}(x_1,x_2,\ldots,x_n)$
Public Key
$P_{\frac{n}{2}+1}(x_1,x_2,\ldots,x_n)$
$P_n(x_1,x_2,\ldots,x_n)$
E

Figure 1. A graphical presentation of our MQ "minus" scheme.

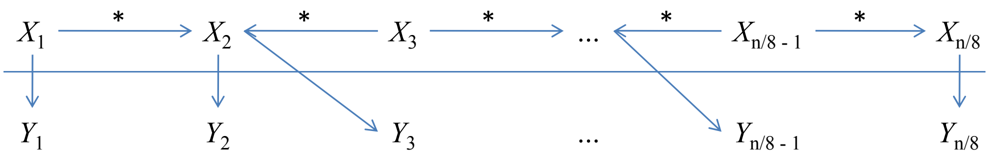

Figure 2. A graphical presentation of the construction of the central bijective multivariate quadratic mapping P'.

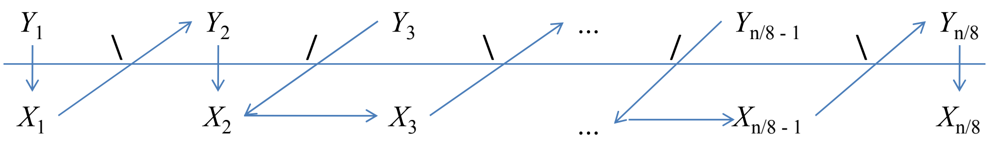

Figure 3. A graphical presentation of the construction of the inverse central mapping P'^{-1} with parastrophe operations.

The algorithm for generating the public and private key is defined in Table 2.

Let us denote by $D(\mathbf{y})$ the composition of inverse operations $\mathbf{S}^{-1}$, P'^{-1} and $\mathbf{S}'^{-1}$ on vector $\mathbf{y}$ i.e. $D(\mathbf{y}) \equiv \mathbf{S}^{-1}(P'^{-1}(\mathbf{S}'^{-1}(\mathbf{y})))$. Also, let us denote by $E(\mathbf{x})$ the mapping of a vector $\mathbf{x}$ with the public polynomials $P_i(x_1, \ldots, x_n)$ $i = 1 + \frac{n}{2}, \ldots, n$. Both signing and verification for MQQ-SIG are graphically presented on Fig. 4 while the algorithmic steps for the signing procedure are presented in details in Table 3, and the verification steps in Table 4.

The central Bijective multivariate quadratic mapping $P'(\mathbf{x})$
Input. A vector $\mathbf{x} = (f_1, \dots, f_n)$ of n linear Boolean functions of n variables. We implicitly suppose that a multivariate quadratic quasigroup $*$ is previously defined, and that $n = 32 \times k$, with $k \in \{5, 6, 7, 8\}$ already fixed.
Output. 8 linear expressions $P'_i(x_1, \dots, x_n), i = 1, \dots, 8$ and $n-$ 8 multivariate quadratic polynomials $P'_i(x_1, \dots, x_n), i = 9, \dots, n$
1. Represent a vector $\mathbf{x} = (f_1, \dots, f_n)$ of n linear Boolean functions of n variables $x_1, \dots, x_n$, as a string $\mathbf{x} = X_1 \dots X_{\frac{n}{8}}$ where X_i are vectors of dimension 8;
2. Compute $\mathbf{y} = Y_1 \dots Y_{\frac{n}{8}}$ where: $Y_1 = X_1$, $Y_{j+1} = X_j * X_{j+1}$, for even $j = 2, 4, \dots$, and $Y_{j+1} = X_{j+1} * X_j$, for odd $j = 3, 5, \dots$
3. Output: $\mathbf{y}$.

Table 1. Definition of the central bijective multivariate quadratic mapping $P' : \{0,1\}^n \to \{0,1\}^n$.

Generating Public and Private key for the MQQ-SIG scheme.
Input. Integer n, where $n = 32 \times k$ and $k \in \{5, 6, 7, 8\}$.
Output. A public key $\mathbf{P}$ given by $\frac{n}{2}$ multivariate quadratic polynomials $P_i(x_1, \dots, x_n)$, $i = 1 + \frac{n}{2}, \dots, n$, and a private key given by two permutations σ_0^0 and σ_0^1 on $\{1, \dots, n\}$, and 81 bytes for encoding a quasigroup $*$.
1. Generate an MQQ $*$ according to equations (6) ... (9).
2. Generate a nonsingular $n \times n$ Boolean matrix $\mathbf{S}$ and affine transformation $\mathbf{S}'$ according to equations (10), ..., (13).
3. Compute $\mathbf{y} = \mathbf{S}(P'(\mathbf{S}'(\mathbf{x})))$, where $\mathbf{x} = (x_1, \dots, x_n)$.
4. Output: The public key is $\mathbf{y}$ as $\frac{n}{2}$ multivariate quadratic polynomials $P_i(x_1, \dots, x_n)$ $i = 1 + \frac{n}{2}, \dots, n$, and the private key is the tuple $(\sigma_0^0, \sigma_0^1, *)$

Table 2. Generating the public and private key

4. Design Rationale

In this section we will describe the reasons behind the choices we made when designing MQQ-SIG. In the next subsection we explain why and how we chose the nonsingular Boolean matrices, and in subsection 4.2 we explain why and how we chose our quasigroups.

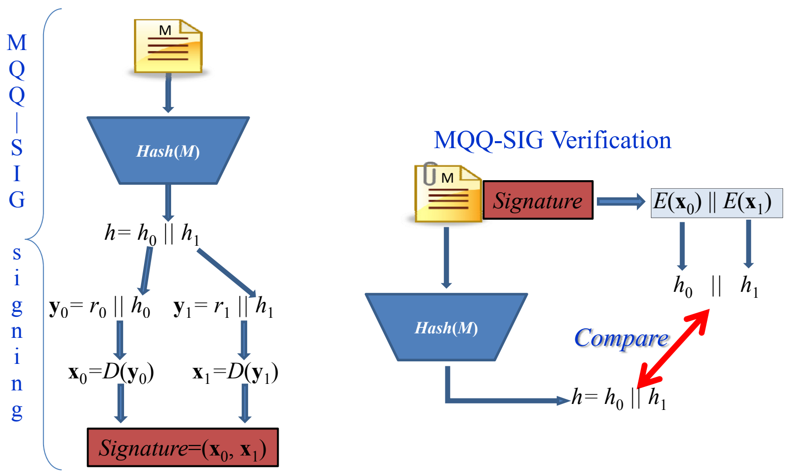
MQQ-SIG signing
M
Hash(M)
$h= h_0 \,||\, h_1$
$\mathbf{y}_0= r_0 \,||\, h_0$
$\mathbf{y}_1= r_1 \,||\, h_1$
$\mathbf{x}_0=D(\mathbf{y}_0)$
$\mathbf{x}_1=D(\mathbf{y}_1)$
Signature=$(\mathbf{x}_0, \mathbf{x}_1)$
MQQ-SIG Verification
M
Signature
$E(\mathbf{x}_0) \,||\, E(\mathbf{x}_1)$
$h_0 \,||\, h_1$
Hash(M)
Compare
$h= h_0 \,||\, h_1$

Figure 4. A graphical presentation of the signing and verification process with MQQ-SIG.

Signing with a private key $(\sigma_0^0, \sigma_0^1, *)$
Input. A document M to be signed.
Output. A signature $\mathbf{sig} = (\mathbf{x}_0, \mathbf{x}_1)$.
1. Compute the pair $h = h_0 \| h_1 \leftarrow Hash(M)$, where $Hash()$ is the standardized hash function. Here we assume that the output of the hash function is n bits, and that h_0 and h_1 are $\frac{n}{2}$ bits long.   2. Set $\mathbf{y}_0 = r_0 \| h_0$ and $\mathbf{y}_1 = r_1 \| h_1$, where the values r_0 and r_1 are $\frac{n}{2}$-bit values chosen uniformly at random.   3. Compute $\mathbf{x}_0 = D(\mathbf{y}_0)$ and $\mathbf{x}_1 = D(\mathbf{y}_1)$.   4. The MQQ-SIG digital signature of the document M is the pair $\mathbf{sig} = (\mathbf{x}_0, \mathbf{x}_1)$.

Table 3. Digital signing

4.1 Nonsingular Boolean matrices in MQQ-SIG

The nonsingular Boolean matrices that are used in MQQ-SIG are generated in a specific way. In general, we need n^2 bits to store a randomly generated nonsingular Boolean matrix of size $n \times n$. In our case we need to store $\mathbf{S}^{-1}$ because we need it in the process of signing. With our proposed sizes for $n = 160, 192, 224, 256$, storing $\mathbf{S}^{-1}$ would require between 3.125 and 8.0 Kbytes.

The idea of reducing the size of the keys in MQ schemes by using circulant matrices has been applied previously in several works [YCCC06, SBA09, PBB10]. Instead of using one circulant matrix, we use two. The rationale why

Signature verification with a public key $\mathbf{P} = \{P_i(x_1, \ldots, x_n) \mid i = 1 + \frac{n}{2}, \ldots, n\}$
Input. A document M and its signature $\mathbf{sig} = (\mathbf{x}_0, \mathbf{x}_1)$.
Output. TRUE or FALSE.
1. Compute $h = h_0 \| h_1 = Hash(M)$, where M is the signed message, and $Hash()$ is the standardized hash function.   2. Compute $\mathbf{z}_0 = E(\mathbf{x}_0)$ and $\mathbf{z}_1 = E(\mathbf{x}_1)$.   3. If $\mathbf{z}_0 = h_0$ and $\mathbf{z}_1 = h_1$ then return TRUE, else return FALSE.

Table 4. Digital verification

and how we construct the private linear (affine) transformations from them is given in what follows.

In order to compress the private information for the linear and affine transformations we define nonsingular matrices $\mathbf{S}$ by the following expression:

$$\mathbf{S}^{-1} = \bigoplus_{i=0}^{\frac{n}{16}} I_{\sigma_i^0} \oplus \bigoplus_{i=0}^{\frac{n}{16}+3} I_{\sigma_i^1}, \tag{10}$$

where $I_{\sigma_i^0}, i = \{0, 1, 2, \ldots, \frac{n}{16}\}$ and $I_{\sigma_i^1}, i = \{0, 1, 2, \ldots, \frac{n}{16}+1\}$ are permutation matrices of size n, the operation $\oplus$ is a "bitwise exclusive or" of the elements in the permutation matrices and permutations σ_i^0 and σ_i^1 are permutations on n elements. They are defined by the following expressions:

$$\begin{cases} \sigma_0^0 & - \quad \text{random permutation on } \{1, 2, \ldots n\}, \\ \sigma_i^0 & = \quad RotateLeft(\sigma_{i-1}^0, 8), \text{ for } i = 1, \ldots, \frac{n}{16}, \\ \sigma_0^1 & - \quad \text{random permutation on } \{1, 2, \ldots n\}, \\ \sigma_i^1 & = \quad RotateLeft(\sigma_{i-1}^1, 8), \text{ for } i = 1, \ldots, \frac{n}{16} + 1, \end{cases} \tag{11}$$

We chose the permutations σ_0^0 and σ_0^1 such that the expression (10) gives a non-singular matrix $\mathbf{S}^{-1}$ (and $\mathbf{S} = (\mathbf{S}^{-1})^{-1}$). From $\mathbf{S}$ we will obtain the affine transformation

$$\mathbf{S}'(\mathbf{x}) = \mathbf{S} \cdot \mathbf{x} + \mathbf{v}, \tag{12}$$

where the vector $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ is an n–dimensional Boolean vector defined from the values of the permutation $\sigma_0^1 = (s_1, s_2, \ldots, s_n)$ by the following expression:

$$v_i = \left(\left(\frac{\left(\left(s_{1+\lfloor\frac{i-1}{8}\rfloor}\right) \bmod 16\right) \times 16}{2^{(8-i) \bmod 8}}\right) + \left(\frac{s_{65+\lfloor\frac{i-1}{8}\rfloor}}{2^{(8-i) \bmod 8}}\right)\right) \bmod 2. \qquad (13)$$

In words: we construct the bits of the vector $\mathbf{v}$ by constructing two arrays. The first array is constructed by taking the four least significant bits of the values $s_1, \ldots, s_{\frac{n}{8}}$ and each of them is shifted by four positions to the left. The second array is just simple extraction of the values $s_{65}, \ldots, s_{65+\frac{n}{8}}$. Finally we XOR respectively those two arrays of values in order to produce the vector $\mathbf{v}$ of n bits. *Although the expression (13) looks complex, it is chosen specifically to be very fast in software and hardware.*

PROPOSITION 5 *The linear transformation $\mathbf{S}^{-1}$ can be encoded in a unique way with $2n$ bytes.* $\square$

The reasons why we decided to use two permutations σ_0^0 and σ_0^1 in order to define the matrix $\mathbf{S}^{-1}$ as in (10) are due to the fact that the inverse matrix of any circulant matrix is again circulant [Dav94]. Thus, if we would use a circulant matrix $\mathbf{S}^{-1}$, its inverse $\mathbf{S}$ that is used in the production of the public key would be also circulant. From a cryptographic point of view, we wanted to avoid the circular property of $\mathbf{S}$ since its strong regularity. This strong regularity might affect the randomness of the multivariate quadratic expressions in the public key. We have made a tradeoff between the totaly non-circulant matrix $\mathbf{S}$ generated completely by a uniformly distributed random process which will cost a lot in terms of space, and the regular circulant matrices, by using two circulant matrices that are combined as it is described in the expression (10). The obtained $\mathbf{S}$ from $\mathbf{S}^{-1}$ is without the circulant regularity, and still we can store it in just $2n$ bytes.

To illustrate our technique for producing non-circulant matrices $\mathbf{S}^{-1}$ and $\mathbf{S}$ we give the following baby example with $n = 16$ and where rotations to the left are performed by 2 positions.

Let $\sigma_0^0 = \left(\begin{smallmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 0 & 6 & 2 & 5 & 15 & 8 & 11 & 12 & 1 & 9 & 14 & 3 & 10 & 7 & 4 & 13 \end{smallmatrix}\right)$ and $\sigma_0^1 = \left(\begin{smallmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 12 & 5 & 14 & 2 & 6 & 7 & 9 & 0 & 10 & 11 & 8 & 4 & 1 & 15 & 13 & 3 \end{smallmatrix}\right)$. Since this is a baby example, we have to adopt the expression (10) for this smaller value of n. The adopted expression is: $\mathbf{S}^{-1} = \bigoplus_{i=0}^{2} I_{\sigma_i^0} \oplus \bigoplus_{i=0}^{3} I_{\sigma_i^1}$

and we get

$$\mathbf{S}^{-1} = \begin{pmatrix}
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0
\end{pmatrix}$$

and

$$\mathbf{S} = \begin{pmatrix}
1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0
\end{pmatrix}$$

Note that **S** is not a circulant matrix.

4.2 Choosing the order and characteristics of quasigroups

In the original MQQ proposal [GMK08c, GMK08b], the authors used several different multivariate quasigroups of order 2^5. That design decision was mainly done because the authors did not know how to construct MQ quasigroups of bigger order.

In the meantime, Chen et al., in [CKG10] and Samardjiska et al., in [SMG10] have found ways how to construct MQQs of arbitrary order 2^d. Thus, we have decided to use quasigroups of order 2^8. That decision was made in order to match the byte size of 8 bits. This enables efficient implementations of MQQ-SIG even on tiny industrial 8-bit MCUs, as well as on high end systems (PCs or workstations). The left and right parastrophes can be pre-computed each taking 64KBytes. These pre-computed parastrophes can speedup the signing phase at least 10 times, but using pre-computed parastrophes of size 2^d where $d > 8$ simply becomes too costly.

Without going into details of the different characteristics of MQQs produced by methods described in [CKG10] and [SMG10] we can say that for encoding MQQs as described in [SMG10] we need 256 bytes, while for MQQs from [CKG10] we need just 81 bytes (see Proposition 4). This is due to the fact that MQQs in [CKG10] have bi-linear nature, while MQQs constructed in [SMG10] are based on T-functions and generally are not bi-linear.

We have performed experiments with both types of MQQs and after removing $\frac{n}{2}$ MQQ's expressions from the public key, we have not observed any security consequences of using the bi-linear MQQs from [CKG10]. That fact combined with the fact that the knowledge of MQQ is a part of the private key, and that the encoding of MQQs from [CKG10] needs just 81 bytes (versus 256 bytes for MQQs from [SMG10]), was the decisive argument in favor of MQQs defined in [CKG10].

In our design we use affine transformation $\mathbf{S}'$ instead of the linear one $\mathbf{S}$, and we also use a non-zero vector $\mathbf{c}$ in the quasigroup construction. The reasons for this is that without $\mathbf{S}'$ our scheme would have the zeroth vector as a fixed point and the same is true for a quasigroup that has $\mathbf{c} = \mathbf{0}$. We consider that these properties are unnecessary and easily avoidable weaknesses.

5. Security analysis of the algorithm

In this section we will describe all the security analysis we have performed during the design of MQQ-SIG. First we want to emphasize that MQQ-SIG similarly as the original MQQ is still resistant against the well know attacks such as: Patarin's chosen plaintext attack on MIA scheme [Pat00], the attacks with differential cryptanalysis that were proposed by Fouque, Granboulan and Stern in [FGS05], solving the isomorphism of polynomials with one secret done by Perret and others in [Per05, FP06b, BFFP11] and MinRank attacks. For the resistance against MinRank attacks we want to note that the minimal rank r of the matrices $\mathbf{B}_{f_i}$ for the nonlinear part of our scheme have to fulfil the conditions (8a, 8b), thus at least one of the ranks is 14 and all of the ranks are at least 12. Additionally, it is not known how to extend the MinRank attack to our scheme, since some equations of the public-key have been removed. In [BFP11], it has been proved that the attack can be extended when 1 equation is removed in HFE. However, the attack cannot be applied in our context when $\frac{n}{2}$ equations are removed.

We suggest the reader to see [GMK08c, GMK08b] for the arguments why MQQ-SIG is resistant against these attacks.

5.1 Experiments with Gröbner bases

The public key encryption algorithm MQQ introduced in [GMK08c, GMK08b] was quickly shown to be weak against algebraic cryptanalysis. It was broken both by Perret [Per08] using Gröbner basis approach, and by Emam Mohamed et al [MDBW09] using MutantXL. Later Faugère et al [FØPG10] explained why the MQQ systems are so easy to solve in practice.

To understand their results we must first introduce to concept of degree of regularity.

As explained in [FØPG10], the complexity of computing a Gröbner basis of an ideal depends on the maximum degree of the polynomials appearing during the computation. This degree, called *degree of regularity*, is the key parameter for understanding the complexity of a Gröbner basis computation [BFS04]. Indeed, the complexity of the computation is exponential in the degree of regularity D_{reg}, more precisely the complexity is:

$$\mathcal{O}(n^{\omega D_{\text{reg}}}), \tag{14}$$

which basically correspond to the complexity of reducing a matrix of size $\approx n^{D_{\text{reg}}}$. Here $2 < \omega \leq 3$ is the "linear algebra constant", and n the number of variables of the system. Note that D_{reg} is also a function of the number of variables, n, and the number of equations m. The relation between D_{reg}, n and m depends on the specific system of equations. This relation is well understood for regular (and semi-regular) systems of equations [Bar04, BFS02, BFS04, BFSY05]. However, as soon as the system has some kind of structure, this degree is much more difficult to predict.

In [FØPG10], the authors showed that the degree of regularity of the original public key algorithm MQQ was bounded from above by a small constant. Having in mind the successful and very efficient way how Gröbner bases and XL methods are solving the full systems of MQQ equations, we want to ensure that MQQ-SIG does not have a similar small bound on the degree of regularity. A classical way to avoid this is to remove some equations of the system. Indeed, an under-defined system of equations ($n > m$) will have an exponential number of solutions. This is an issue since the complexity of Gröbner bases is also related to the number of solutions [FGLM93]. To circumvent this problem, a solution is to fix $n - m$ variables (or more [BFP09]). However, as soon as sufficiently many variables were fixed, we observed that the new system behaved as a "random" system of equations of the same size. This has been also observed and used in the hybrid approach [BFP09].

To confirm this behavior in our context, we have performed experiments on MQQ-SIG equations systems of reduced sizes. The observed degree of regularity is compared to the expected degree of regularity for a random multivariate system of the same size. The strategy for choosing S has changed during the course of our experiments. The experiments where performed with random Boolean matrices. However, from a security against Gröbner bases attack point of view, the most important feature is that we ensure that the 8 linear expressions are removed from the equations set. Below is our experimental strategy for small-scale version of MQQ-SIG equation systems in n variables:

```
1 Repeat:
2 Generate a bijective multivariate quadratic mapping $P'_i(x_1,\dots,x_n), i = 1,\dots,n$
3 Remove the 8 linear expressions $P'_i(x_1,\dots,x_n), i = 1,\dots,8$
4 Multiply with random nonsingular Boolean matrices $S_R$ and $T_R$, $\mathbf{P} = \mathbf{S_R} \circ P' \circ \mathbf{T_R}$.
5 For $j = 8$ to $j = \frac{n}{2}$ do:
    (a) Remove the last $j-8$ equations from $\mathbf{P}$.
    (b) Set a random Boolean vector $(x_{n-j+1},\dots,x_n) \in \{0,1\}^j$
    (c) Obtain a system $\mathbf{P_1} = \{P_i(x_1,\dots,x_{n-j}) \mid i = 1,\dots,n-j\}$ of $n-j$ equations with $n-j$ variables $(x_1,\dots,x_{n-j})$
    (d) Call $F_4(\mathbf{P_1})$ algorithm from Magma, to find a Gröbner basis for the system $\mathbf{P_1}$, and measure the degree of regularity.
6 Compute the average degree of regularity.
```

We have performed 100 experiments for $16, 24$,32 and 40 variables. Due to the complexity, the experiments have only been repeated 10 times for 48 variables and just once for 56 and 64 variables. For 56 and 64 variables many of the instances either required more than the 1TB RAM our system has, or did not finish after about 1 month of computation. These instances are marked with a $\cdot$ in the table. We also experienced that 72 variables with 36 equations removed did not finish after about a month of computation. The experiments were done with Magma 2.17-3's implementation [MAG] of the F_4 [Fau99] algorithm on a workstation with 32 cores based on Intel Xeon 2.27GHz, with 1TB of RAM. The results of these experiments are listed in Table 5. In the table the expected degree of regularity for a random system of equations over $GF(2)$ in $V-R$ variables are also listed in parentheses. These numbers have been calculated using the formula provided in [BFS02]. From the table we see that the bigger percentage of equations we remove from the system, the closer the measured degree of regularity is to a random system of equations. The reason for this is that we are removing crucial relations among terms, thus rendering the remaining sets of equations as random sets of multivariate equations. It is then natural to formulate the following conjecture:

Conjecture 6 *For every full set of public key equations produced by MQQ as defined in steps 1–3 in Table 2, removing $\frac{n}{2}$ of the equations, makes the remaining set of $\frac{n}{2}$ multivariate quadratic equations to act as a set of $\frac{n}{2}$ random multivariate quadratic equations with $\frac{n}{2}$ variables in $GF(2)$.*

R/V	16	24	32	40	48	56	64
8	**3,00(3)**	3,33 (5)	3,75 (6)	4,15 (6)	4,30 (7)	5 (8)	6 (9)
9		3,09 (4)	3,97 (5)	4,05 (6)	4,10 (7)	4 (8)	4 (9)
10		3,74 (4)	4,00 (5)	4,04 (6)	4,30 (7)	4 (8)	5 (9)
11		3,87 (4)	4,01 (5)	4,56 (6)	4,90 (7)	5 (8)	5 (9)
12		**3,93(4)**	4,06 (5)	5,00 (6)	5,00 (7)	5 (8)	5 (9)
13			4,33 (5)	5,00 (6)	5,00 (7)	5 (8)	· (9)
14			4,48 (5)	5,00 (6)	5,50 (7)	6 (8)	· (9)
15			4,46 (5)	5,00 (6)	5,60 (7)	6 (8)	· (8)
16			**4,21(5)**	5,00 (6)	5,60 (6)	6 (7)	· (8)
17				5,00 (5)	5,90 (6)	· (7)	· (8)
18				5,00 (5)	5,90 (6)	· (7)	· (8)
19				5,00 (5)	6,00 (6)	· (7)	· (8)
20				**5,00(5)**	6,00 (6)	· (7)	· (8)
21					6,00 (6)	· (7)	· (8)
22					6,00 (6)	· (7)	· (8)
23					6,00 (6)	· (7)	· (8)
24					**6,00(6)**	6 (6)	· (7)
25						6 (6)	· (7)
26						6 (6)	· (7)
27						6 (6)	· (7)
28						**6(6)**	· (7)
29							· (7)
30							· (7)
31							· (7)
32							**6(6)**

Table 5. The average degree of regularity for a MQQ signature system in V variables with R equations removed. In parentheses, the expected degree of regularity for a random system of size $V - R$.

5.2 The size of the pool of MQQs of order 2^8

It is very important to address the question of the size of the set of MQQs of order 2^8 that we use in our MQQ-SIG scheme. In [CKG10], Chen et al., gave a lower bound on the number of MQQs of order 2^8. That number is projected to 2^{273}. However, we are using additional conditions (8). By a heuristical measuring we have obtained that approximately one in 2^7 randomly generated MQQs of order 2^8 complies with the conditions (8). That means that the lower bound of the size of the pool of MQQs of order 2^8 is 2^{266}.

5.3 Secret Key Leakage Scenarios

Originally this attack was presented to us by an anonymous reviewer of an earlier variant of our scheme submitted to WCC 2011. We would like to express *big acknowledgement* to that anonymous reviewer.

In a previous version of our scheme instead of $\mathbf{y} = r_0||h_0$, the value $\mathbf{y} = h$ obtained as the output of the hashing procedure is n bits long, and the signature part is $\mathbf{x} = D(\mathbf{y})$. The following Chosen Message Attack could then be launched. An attacker asks for signatures of $1 + n + \binom{n}{2} + O(1)$ messages i.e. he will have the triplets $(M_i, \mathbf{x}_i, \mathbf{y}_i \equiv Hash(M_i))$. He will then attempt to recover the missing $\frac{n}{2}$ equations in the public key. Given the missing equations he can successfully launch an efficient Gröbner bases attack.

Consider the extraction of the first missing equation $y_1 = P_1(x_1, \ldots, x_n)$, which can be expressed in a general form as:

$$y_1 = d_0 + d_1 x_1 + d_2 x_2 + \ldots + d_n x_n + d_{n+1} x_1 x_2 + \ldots + d_{2n+1} x_2 x_3 + \ldots + d_{1+n+\binom{n}{2}} x_{n-1} x_n. \tag{15}$$

Since the attacker knows the values of $1 + n + \binom{n}{2} + O(1)$ triplets $(M_i, \mathbf{x}_i, \mathbf{y}_i)$, from the equation (15), with high probability, he can obtain a full rank linear system of equations with $1 + n + \binom{n}{2}$ unknown variables d_j. Additionally and most importantly he knows the corresponding values $y_1^{(i)}$ for every of the values $\mathbf{y}_i = (y_1^{(i)}, \ldots, y_n^{(i)})$. Thus, by solving the obtained linear system of equations he can recover the values of the coefficients d_j i.e. he can recover the first missing equation. The extraction of other hidden equations is similar.

This attack is easily mitigated by our strategy to construct the values $\mathbf{y}_0 = r_0||h_0$ and $\mathbf{y}_1 = r_1||h_1$ where r_0 and r_1 are strings of $\frac{n}{2}$ randomly generated bits with every signing invocation, and $h = h_0||h_1$ is the hash output that is digesting the message M.

We formulate the previous discussion about the leakage of the private key in the non-randomized MQQ-SIG and its prevention by the following two lemmas:

Lemma 7 *For any MQQ signature scheme with K expressions removed, if the signatures for the messages M are obtained as $\mathbf{x} = D(\mathbf{y})$, where $\mathbf{y} = Hash(M)$, the extraction of the removed part has complexity of $O(Kn^2)$.* □

Lemma 8 *For the MQQ-SIG signature scheme as defined in steps 1–3 in Table 2, by removing $\frac{n}{2}$ of the expressions, an attack for extraction of the removed part as in Lemma 7 has complexity of $O(2^{n^3})$.*

Proof Since the signature for a message M has two parts $\mathbf{x}_0$ and $\mathbf{x}_1$ that are computed as $\mathbf{x}_0 = D(r_0||h_0)$ and $\mathbf{x}_1 = D(r_1||h_1)$ where $h = h_0||h_1 = Hash(M)$, and the values r_0 and r_1 are $\frac{n}{2}$-bit values chosen uniformly at random for every particular procedure of signing, the extraction technique from Lemma 7 can give the correct extraction of the hidden part if and only if for all $O(n^2)$ queries, the random values r_0 and r_1 are known to the attacker. Having in mind that for every produced signature the values r_0 and r_1 are unknown, fresh, uniformly distributed random values, the probability of guessing their values is $2^{-\frac{n}{2}} \times 2^{-\frac{n}{2}} = 2^{-n}$. For all $O(n^2)$ queries this gives us a total probability of $(2^{-n})^{n^2} = 2^{-n^3}$, i.e. the complexity for extracting the hidden part is $O(2^{n^3})$. $\square$

5.4 MQQ-SIG is Provably CMA Resistant

We will use the following definition of security against chosen message attack [Kat10]:

DEFINITION 9 *Signature scheme (Gen, Sign, Vrfy) is* ***existentially unforgeable under a chosen-message attack*** *if for all probabilistic, polynomial-time adversaries A, the success probability of A in the following experiment is negligible (as a function of k):*

1 *The key-generation algorithm Gen(1^k) is run to obtain a pair of keys (pk, sk)*

2 *A is given pk and allowed to interact with a signing oracle $Sign_{sk}(\cdot)$, requesting signatures on as many messages as it likes. Let M denote the set of messages queried to the signing oracle by A.*

3 *Eventually, A outputs (m, σ)*

4 *A* ***succeeds*** *if $Vrfy_{pk}(m, \sigma) = 1$ and $m \notin M$*

It is well known that solving multivariate quadratic polynomials is an NP-complete problem (see for instance [GJ79]). This theorem is repeated below.

THEOREM 10 ([GJ79]) *Let $P_i(x_1, \ldots, x_n), 1 \leq i \leq m$ be a collection of polynomials over $GF[2]$. The problem of finding $u_1, \ldots, u_n$ such that $P_i(u_1, \ldots, u_n) = 0$ for $1 \leq i \leq m$ remains NP-complete even if none of the polynomials has a term involving more than two variables or if there is just one polynomial.*

THEOREM 11 *MQQ-SIG is CMA resistant in the random oracle model under the assumptions that solving $\frac{n}{2}$ MQQ equations with n variables is as hard as solving systems of $\frac{n}{2}$ random multivariate quadratic equations.*

In what follows we give a sketch of the proof and the ideas how to use the fact that the verification of the MQQ-SIG signatures depends on the values h_0 and h_1 that are each $\frac{n}{2}$ bits long. This fact implies that a chosen message attack on MQQ-SIG would need either at least $2^{\frac{n}{2}}$ pairs of messages in order to find a collision of the used hash function or to solve the system of $\frac{n}{2}$ random multivariate quadratic equations with n variables. A formal proof showing the strict reduction from the CMA-resistance of the scheme to the assumption that solving $\frac{n}{2}$ MQQ equations with n variables is as hard as solving systems of $\frac{n}{2}$ random multivariate quadratic equations with n variables will be given in the extended version of this paper.

Proof (sketch) The security parameter input to the generating algorithm is $k = \frac{n}{2}$, which controls the number of equations over $GF(2)$ and is directly connected with the value n: the output size of the hash function.

Given the assumption that solving $\frac{n}{2}$ MQQ equations with n variables is as hard as solving systems of $\frac{n}{2}$ random multivariate quadratic equations, there are no structural weaknesses of the MQQ equations that can be exploited to solve the system faster then solving $\frac{n}{2}$ random multivariate quadratic equations. This means the adversary has basically three strategies of breaking MQQ-SIG:

1 To find a collision in the hash digest $(h_0||h_1)$ of length $n = 2k$.

2 To solve two systems of $\frac{n}{2}$ MQ equations with n Boolean variables.

3 Some combination of the two above.

Strategy 1: Breaking with the strategy 1 means finding a collision for a random oracle with a $n = 2k$ bit output. Interacting with the signing oracle will not help the adversary for this instance, since he is only interested in the output of the random oracle. By the generic birthday attack the adversary needs $O(2^k)$ queries to the random oracle to find a collision for the whole digest. The probability for a polynomial time adversary to break the signature scheme by finding a collision in the digest is therefore negligible in k.

Strategy 2: Under the assumption that solving the $\frac{n}{2}$ MQQ equations in n variables is as hard as solving k MQ equations in k variables, we know by Theorem 10 that the probability the adversary solves either of the equations with the strategy 2 is negligible in k. However, to prove that the signature scheme is CMA, we must also show that querying the signing oracle gives the adversary no significant advantage in solving the equations. There are two ways the signing oracle might leak information.

1 Signing leaks information about the hidden equations:

In Lemma 8 we proved that extracting information about the removed part has complexity $O(2^{n^3})$. With our security parameter of $k = \frac{n}{2}$ this is out of reach for a polynomially bound adversary.

2 Signing leaks some other information that can help solve the equation system:

Consider the following game where the adversary does not have access to the random oracle. The adversary asks for a signature for a chosen message M. The signing oracle then flips a coin.

(a) If the coin land on heads the signing oracle outputs the digest $H(M) = (h_0||h_1)$, and the corresponding signature $(\mathbf{x}_0, \mathbf{x}_1)$.

(b) If the coin lands on tails the signing oracle outputs the evaluation of the encryption function in some random numbers $(E(r_0), E(r_1))$, and the corresponding random numbers (r_0, r_1).

The adversary is then asked if the coin is heads or tails.

Since by the definition of random oracles the output of $H(M)$ is independent of M, it should be clear that the adversary has no way of winning the game above. This illustrates that from the adversary point of view, there is no difference between querying the signing oracle and evaluating the known equations on random inputs. The fact that the adversary actually has access to the random oracle does not change this conclusion because the adversary has no control over the output of the random oracle.

To summarize this means that signing reveals no information about the hidden equations, and leaks no other information that can be used to solve the equations. The signature scheme is therefore CMA with respect to the strategy number 2.

Strategy 3: First note that finding a $k-l$, $1 \leq l \leq k$, bit collision in, for instance h_0, will not help computing the corresponding $\mathbf{x}_0$. The reason for this is the nature of the random MQ equations, where the solution to the system will drastically change by just flipping one output bit. Namely, each output bit depends on average on $\frac{k(k-1)}{2}$ combination of all pairs of variables. This means that the best for the adversary in strategy attack number 3 is to find a collision in either h_0 or h_1, and to solve the equation system for the part that a solution is not known. This requires "just" $O(2^{\frac{k}{2}-1})$ calls to the random oracle. However, the adversary still needs to solve a system of k equations in $2k$ variables, proven to be CMA resistant by the arguments under the attack strategy number 2. $\square$

5.5 Non-applicability of successful attacks against STS on MQQ-SIG

An anonymous reviewer for IMACC 2011 (to whom we express *big acknowledgement*) has pointed out an interesting comment that MQQ-SIG scheme looks similar as STS schemes and thus the successful attacks that have broken

STS schemes may also break MQQ-SIG. Here we explain the crucial and essential differences between STS and MQQ-SIG schemes and the non-applicability of successful attacks against STS on MQQ-SIG.

The Stepwise Triangular Scheme was introduced by Wolf et al., [WBP04] as a generalization of earlier multivariate quadratic schemes, such as [Sha93, Moh99, GC00, KS04]. The main purpose of the generalization in [WBP04] was to show how all these schemes, and the whole STS family in general, is either insecure or impractical. The general attacks presented exploit the chain of kernels introduced by the triangular structure of the hidden polynomials.

There are at least two important reasons why this attack is not applicable on MQQ-SIG. First, even tough the kernel of two adjacent sub-blocks share half of each others variables, the triangular structure of the hidden polynomials does not result in a chain of kernels. The production of the public key in MQQ-SIG is essentially parallel and chained for the whole n-dimentional space, while the production of the public key in STS is essentially sequential with increasingly larger embedded subspaces. It is this structure that the attacks on STS exploit.

The second reason is that the attacks linearly combine the public key expressions in order to get ranks within certain values. Non-applicability of these attacks against MQQ-SIG is due to the fact that half of the public key expressions are removed, and linearly combining the remaining half in order to obtain low ranks does not necessarily produce vectors from the kernel of the transformation T^{-1}.

6. Operating characteristics

In this section we discuss the sizes of the private and public key as well as the number of operations for verification and signing.

6.1 The size of the public and the private key

Since the public key consists of $\frac{n}{2}$ randomly generated multivariate quadratic equations, the size of the public key follows the rules given in [WP05]. So, for n bit blocks the size of the public key is $0.5 \times n \times (1 + \frac{n(n+1)}{2})$ bits. The private key of our scheme is the tuple $(\sigma_0^0, \sigma_0^1, *)$. The corresponding memory size needed for storage of the private key is $2n + 81$ bytes.

In Table 6, there are two columns for the size of the private and public key and as we can see for MQQ-SIG the size of the public key for $n \in \{160, 192, 224, 256\}$ is in the range from 125 up to 521 KBytes.

We want to emphasize that recently Samardjiska and Chen in [SCG11] have proposed extension of their algorithms for construction of MQQs over

Security level (power of 2)	Algorithm	KeyGen	Sign 59 bytes (CPU cycles)	Verify (CPU cycles)	Private key size (bytes)	Public key size (bytes)	Signature size (bytes)
80	RSA1024	102,869,553	2,213,112	60,084	1024	128	128
	ECDSA160	1,201,188	944,364	1,083,060	60	40	40
	MQQSIG160	799,501,482	6,534	92,232	401	137,408	40
	RainbowBinary 256181212	30,311,648	38,784	43,800	23,408	30,240	42
96	RSA1536	322,324,721	5,452,076	87,516	1536	192	192
	ECDSA192	1,799,284	1,390,560	1,662,664	72	48	48
	MQQSIG192	800,724,096	7,938	138,972	465	222,360	48
112	RSA2048	786,466,598	11,020,696	125,776	2048	256	256
	ECDSA224	2,022,896	1,555,740	1,821,348	84	56	56
	MQQSIG224	1,107,486,126	9,492	184,392	529	352,828	56
128	RSA3072	2,719,353,538	31,941,760	230,536	3072	384	384
	ECDSA256	2,296,976	1,780,524	2,085,588	96	64	64
	MQQSIG256	1,501,955,022	9,138	218,700	593	526,368	64
	TTS6440	60,827,704	84,892	76,224	16,608	57,600	43
	3ICP	15,520,100	1,641,032	60,856	12,768	35,712	36

Table 6. Comparison between RSA, ECDSA, and several MQ schemes: MQQ-SIG, Rainbow, TTS and 3ICP. Operations have been performed in 64-bit mode of operation on Intel Core i7 920X machine running at 2 GHz.

arbitrary finite fields and that by their construction it is possible to reduce the huge public key size of MQQ-SIG to be in the range 2.3 – 8.8 Kbytes.

6.2 Performance of the software implementation of the MQQ-SIG algorithm

We have implemented MQQ-SIG in C for the SUPERCOP benchmarking system [Be11a] and tested it together with the corresponding RSA [RSA78] and ECC [Mil86, Kob87] (actually ECDSA) and several other multivariate quadratic systems such as: Rainbow [DYC$^+$08], enhanced TTS [YC05] and 3ICP [DWY07]. In Table 6 we give the comparison of the mentioned signatures schemes where the measurements were performed in 64-bit mode of operation on Intel Core i7 920X machine running at 2 GHz. Although, our C code is not yet optimized for the key generation part, we expect that the performance of key generation part to be the most time consuming part of our algorithm.

From the Table 6 it is clear that in signing of 59 bytes MQQ-SIG is faster than RSA in the range from 300 up to 3500 times, and is faster than ECDSA in the range from 140 up to 200 times. If we exclude the time for hashing the messages, signing operations in MQQ-SIG in Table 6 take from 2,500 up to 5,000 cycles. MQQ-SIG is also significantly faster than other multivariate methods such as Rainbow, TTS or 3ICP and that performance advantage in the signing procedure is in the range from 5 to 20 times.

The verification speed in our code is not optimized so far. We expect the optimized verification speed of MQQ-SIG to be in the range of Rainbow, TTS and 3ICP.

7. Conclusions

We have constructed a multivariate quadratic digital signature scheme MQQ-SIG based on multivariate quadratic quasigroups.

By learning about the weaknesses of the previous attempt to design a multivariate quadratic scheme based on quasigroups - MQQ, by analyzing the successful attacks on all existing MQ schemas, and by our experimentally supported assumption that solving $\frac{n}{2}$ quadratic polynomials with n variables is as hard as solving random systems of equations, we have designed a digital signature scheme that in the random oracle model is provably CMA resistant and that we believe is strong enough to attract the attention of the cryptographic community.

The efficiency of producing digital signatures of our scheme outperforms all the existing signature schemes (RSA, ECDSA and other MQ schemes) in the range from 5 up to 3,500 times. The speed of verification of our scheme is similar to the other MQ schemes. However the MQQ-SIG scheme that was described in this paper has an unpractically big public key. The ongoing research efforts are in this direction and soon we can expect MQQ-SIG variants with significantly smaller public keys.

We believe that its superior performance will allow an employment of strong and fast authentication protocols based on the paradigm of the public key cryptography in many new areas of our modern society.

References

[Bar04] Magali Bardet. *Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie.* PhD thesis, Université de Paris VI, 2004.

[Be11] D. J. Bernstein and T. Lange (editors). eBACS: ECRYPT benchmarking of cryptographic systems, 2011. Accessed 12 January,2011.

[Bel67] V. D. Belousov. Osnovi teorii kvazigrup i lup (in russian), 1967. Nauka, Moscow.

[BFFP11] C. Bouillaguet, J.-C. Faugère, P.A. Fouque, and L. Perret. Practical cryptanalysis of the identification scheme based on the isomorphism of polynomial with one secret problem. In *Public Key Cryptography – PKC 2011*, volume 6571 of *LNCS*, pages 441–458. Springer, 2011.

[BFP09] Luk Bettale, Jean-Charles Faugère, and Ludovic Perret. Hybrid approach for solving multivariate systems over finite fields. *Journal of Mathematical Cryptology*, volume 3(issue 3):177–197, 2009.

[BFP11] Luk Bettale, Jean-Charles Faugère, and Ludovic Perret. Cryptanalysis of multivariate and odd-characteristic HFE variants. In *Public Key Cryptography – PKC 2011*, volume 6571 of *LNCS*, pages 441–458. Springer, 2011.

[BFS02] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. Complexity study of Gröbner basis computation. Technical report, INRIA, 2002. `http://www.inria.fr/rrrt/rr-5049.html`.

[BFS04] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In *Proc. International Conference on Polynomial System Solving (ICPSS)*, pages 71–75, 2004.

[BFSY05] Magali Bardet, Jean-Charles Faugère, Bruno Salvy, and Bo-Yin Yang. Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems. In *Proc. of MEGA 2005, Eighth International Symposium on Effective Methods in Algebraic Geometry*, 2005.

[CKG10] Yanling Chen, Svein Johan Knapskog, and Danilo Gligoroski. Multivariate quadratic quasigroups (MQQs): Construction, bounds and complexity. In *Inscrypt*, 6th International Conference on Information Security and Cryptology. Science Press of China, October 2010.

[Dav94] Philip J. Davis. *Circulant Matrices.* AMS Chelsea Publishing, 1994.

[DK74] J. Denes and A. D. Keedwell. *Latin squares and their applications.* Academic Press, New York :, 1974.

[DWY07] Jintai Ding, Christopher Wolf, and Bo-Yin Yang. l-Invertible Cycles for Multivariate Quadratic (MQ) Public Key Cryptography. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography*, volume 4450 of *LNCS*, pages 266–281. Springer, 2007.

[DYC$^+$08] Jintai Ding, Bo-Yin Yang, Chia-Hsin Owen Chen, Ming-Shing Chen, and Chen-Mou Cheng. New Differential-Algebraic Attacks and Reparametrization of Rainbow. In *Proceedings of Applied Cryptography and Network Security, 6th International Conference, ACNS 2008*, volume 5037 of *LNCS*, pages 242–257, 2008.

[Fau99] Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases (F_4). *Journal of Pure and Applied Algebra*, 139(1-3):61–88, June 1999.

[FGLM93] J. C. Faugère, P. Gianni, D. Lazard, and T. Mora. Efficient computation of zero-dimensional Gröbner bases by change of ordering. *J. Symb. Comput.*, 16:329–344, October 1993.

[FGS05] Pierre-Alain Fouque, Louis Granboulan, and Jacques Stern. Differential cryptanalysis for multivariate schemes. In *Eurocrypt 2005*, volume 3494 of *LNCS*, pages 341–353. Springer, 2005.

[FØPG10] Jean-Charles Faugère, Rune Steinsmo Ødegård, Ludovic Perret, and Danilo Gligoroski. Analysis of the MQQ public key cryptosystem. In Swee-Huay Heng, Rebecca N. Wright, and Bok-Min Goi, editors, *CANS*, volume 6467 of *LNCS*, pages 169–183. Springer, 2010.

[FP06] Jean-Charles Faugère and Ludovic Perret. Polynomial Equivalence Problems: Algorithmic and Theoretical Aspects. In *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 30–47. Springer, 2006.

[GC00] Louis Goubin and Nicolas T. Courtois. Cryptanalysis of the TTM cryptosystem. In *Advances of Cryptology, Asiacrypt 2000*, volume 1976 of *LNCS*, pages 44–57. Springer, 2000.

[GJ79] Michael R. Garey and Davis S. Johnson. *Computers and Intractability. A guide to the theory of NP-Completeness.* Bell Telephone Laberatories, Incorporated, 1979.

[GMK08a] Danilo Gligoroski, Smile Markovski, and Svein Johan Knapskog. Multivariate quadratic trapdoor functions based on multivariate quadratic quasigroups. In *MATH'08: Proceedings of the American Conference on Applied Mathematics*, pages 44–49, Stevens Point, Wisconsin, USA, 2008. World Scientific and Engineering Academy and Society (WSEAS).

[GMK08b] Danilo Gligoroski, Smile Markovski, and Svein Johan Knapskog. Public key block cipher based on multivariate quadratic quasigroups. In Cryptology ePrint Archive, Report 2008/320, 2008.

[IM86] Hideki Imai and Tsutomu Matsumoto. Algebraic methods for constructing asymmetric cryptosystems. In *Proceedings of the 3rd International Conference on Algebraic Algorithms and Error-Correcting Codes*, AAECC-3, pages 108–119, London, UK, 1986. Springer-Verlag.

[Kat10] Jonathan Katz. *Digital Signatures.* Springer, 2010.

[KHPG99] Aviad Kipnis, Hamarpe St. Har Hotzvim, Jacques Patarin, and Louis Goubin. Unbalanced Oil and Vinegar Signature Schemes. In *Advances in Cryptology - EUROCRYPT 1999*, volume 5479 of *LNCS*, pages 206–222. Springer, 1999.

[Kob87] Neal Koblitz. Elliptic Curve Cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.

[KS04] Masao Kasahara and Ryuichi Sakai. A construction of public key cryptosystem for realizing ciphertext of size 100 bit and digital signature scheme. *IEICE Transactions*, 87-A(1):102–109, 2004.

[MAG] MAGMA. High performance software for algebra, number theory, and geometry — a large commercial software package. url:`http://magma.maths.usyd.edu.au`.

[MDBW09] Mohamed Saied Mohamed, Jintai Ding, Johannes Buchmann, and Fabian Werner. Algebraic attack on the MQQ public key cryptosystem. In *CANS '09: Proceedings of the 8th International Conference on Cryptology and Network Security*, pages 392–401, Berlin, Heidelberg, 2009. Springer-Verlag.

[MI88] Tsutomu Matsumoto and Hideki Imai. Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In *Advances in Cryptology – EUROCRYPT 1988*, volume 330 of *LNCS*, pages 419–453. Springer–Verlag, 1988.

[Mil86] Victor S. Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology - CRYPTO '85*, volume 218 of *LNCS*, pages 417–426. Springer-Verlag, 1986.

[Moh99] T. Moh. A public key system with signature and master key functions. Communications in Algebra, 1999.

[Pat96] Jacques Patarin. Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms. In *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 33–48. Springer-Verlag, 1996.

[Pat00] Jacques Patarin. Cryptanalysis of the Matsumoto and Imai public key scheme of Eurocrypt 98. *Des. Codes Cryptography*, 20:175–209, June 2000.

[PBB10] Albrecht Petzoldt, Stanislav Bulygin, and Johannes Buchmann. CyclicRainbow - A multivariate Signature Scheme with a Partially Cyclic Public Key based on Rainbow. Cryptology ePrint Archive, Report 2010/424, 2010.

[Per05] Ludovic Perret. A fast cryptanalysis of the isomorphism of polynomials with one secret problem. In *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 354–370. Springer, 2005.

[Per08] Ludovic Perret. Personal e-mail communication with Danilo Gligoroski, 2008.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21:120–126, February 1978.

[SBA09] Rajesh P Singh, B.K.Sarma, and A.Saikia. Public key cryptography using permutation p-polynomials over finite fields. Cryptology ePrint Archive, Report 2009/208, 2009. `http://eprint.iacr.org/`.

[SCG11] Simona Samardjiska, Yanling Chen, and Danilo Gligoroski. Construction of Multivariate Quadratic Quasigroups (MQQs) in Arbitrary Galois Fields. In *Proceeding of IAS 2011, Malacca, Malaysia, Dec. 5-8, 2011*, 2011.

[Sha93] Adi Shamir. Efficient signature schemes based on birational permutations. In *Proceedings of CRYPTO'93*, volume 773 of *LNCS*, pages 1–12. Springer-Verlag, 1993.

[SMG10] S. Samardjiska, S. Markovski, and D. Gligoroski. Multivariate quasigroups defined by t-functions. In *Proceedings of SCC2010 - The 2nd International Conference on Symbolic Computation and Cryptography*, 2010.

[Smi07] J. D. H. Smith. *An introduction to quasigroups and their representations.* Chapman & Hall/CRC, 2007.

[WBP04] Christopher Wolf, An Braeken, and Bart Preneel. Efficient Cryptanalysis of RSE(2)PKC and RSSE(2)PKC. In *Conference on Security in Communication Networks – SCN 2004*, volume 3352 of *LNCS*, pages 294–309. Springer, 2004. Extended version:`http://eprint.iacr.org/2004/237`.

[WP05] Christopher Wolf and Bart Preneel. Taxonomy of Public Key Schemes based on the problem of Multivariate Quadratic equations. IACR Eprint archive, 2005. url:`http://eprint.iacr.org/2005/077`.

[YC05] Bo-Yin Yang and Jiun-Ming Chen. Building Secure Tame-like Multivariate Public-Key Cryptosystems: The New TTS. In *Information Security and Privacy, 10th Australasian Conference, ACISP 2005*, volume 3574 of *LNCS*, pages 518–531. Springer, 2005.

[YCCC06] Bo-Yin Yang, Chen-Mou Cheng, Bor-Rong Chen, and Jiun-Ming Chen. Implementing minimized multivariate PKC on low-resource embedded systems. In *Security in Pervasive Computing, SPC 2006*, volume 3934 of *LNCS*, pages 73–88. Springer, 2006.

Bibliography

[AA09] Frederik Armknecht and Gwenolé Ars. Algebraic Attacks on Stream Ciphers with Gröbner Bases. In *Gröbner Bases, Coding, and Cryptography*, pages 329–348. Springer Berlin Heidelberg, 2009.

[AACW08] Martin Albrecht, Daniel Augot, Anne Canteaut, and Ralf-Philipp Weinmann. Algebraic cryptanalysis of symmetric primitives. Technical report, ECRYPT, 2008.

[ABK98] Ross Anderson, Eli Biham, and Lars Knudsen. Serpent: A flexible block cipher with maximum assurance. In *The First Advanced Encryption Standard Candidate Conference*, 1998.

[ADMS09] Jean-Philippe Aumasson, Itai Dinur, Willi Meier, and Adi Shamir. Cube Testers and Key Recovery Attacks on Reduced-Round MD6 and Trivium. In Orr Dunkelman, editor, *Fast Software Encryption (FSE)*, volume 5665 of *LNCS*, pages 1–22. Springer, Berlin, Heidelberg, 2009.

[AF03] G. Ars and J.C Faugère. An Algebraic Cryptanalysis of Nonlinear Filter Generators using Gröbner Bases. INRIA Research Report, n 4739, 2003.

[AF05] G. Ars and J.C Faugère. Algebraic immunities of functions over finite fields. In *Boolean Function : Cryptography and Applications - BFCA 05*, pages 21–38, 2005.

[AFG$^+$08] Daniel Augot, Matthieu Finiasz, Philippe Gaborit, Stéphane Manuel, and Nicolas Sendrier. SHA-3 proposal: FSB. Submission to NIST, 2008.

[AFI$^+$04] Gwénolé Ars, Jean-Charles Faugère, Hideki Imai, Mitsuru Kawazoe, and Makoto Sugita. Comparison between XL and Gröbner Basis Algorithms. In *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 338–353. Springer-Verlag, 2004.

[AHMP08] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and Raphael C.-W. Phan. SHA-3 proposal BLAKE. Submission to NIST (Round 1/2), 2008.

[AM09] Jean-Philippe Aumasson and Willi Meier. Zero-sum distinguishers for reduced Keccak-f and for the core functions of Luffa and Hamsi. NIST mailing list, 2009.

[AMZ08] Rafael Alvarez, Gary McGuire, and Antonio Zamora. The Tangle Hash Function. Submission to NIST, 2008.

[AS09] Kazumaro Aoki and Yu Sasaki. Meet-in-the-Middle Preimage Attacks Against Reduced SHA-0 and SHA-1. In *Advances in Cryptology - CRYPTO 2009*, volume 5677 of *LNCS*, pages 70–89. Springer-Verlag, 2009.

[Ass00] American Bankers Association. *Keyed Hash Message Authentication Code.* ANSI X9.71, Washington, D.C., 2000.

[Aur09] Valerie Aurora. Lifetimes of cryptographic hash functions, 2009. `http://valerieaurora.org/hash.html`.

[BAK98] Eli Biham, Ross J. Anderson, and Lars R. Knudsen. Serpent: A new block cipher proposal. In Serge Vaudenay, editor, *FSE*, volume 1372 of *LNCS*, pages 222–238. Springer, 1998.

[Bar04] Magali Bardet. *Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie.* PhD thesis, Université de Paris VI, 2004.

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer Berlin / Heidelberg, 2004.

[BBBV97] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM J. Comput.*, 26:1510–1523, October 1997.

[BBG$^+$08] Ryad Benadjila, Olivier Billet, Henri Gilbert, Gilles Macario-Rat, Thomas Peyrin, Matt Robshaw, and Yannick Seurin. SHA-3 Proposal: ECHO. Submission to NIST, 2008.

[BCC10] Christina Boura, Anne Canteaut, and Christophe De Canniere. Higher-order differential properties of Keccak and Luffa. Cryptology ePrint Archive, Report 2010/589, 2010.

[BCCM$^+$08] Emmanuel Bresson, Anne Canteaut, Benoît Chevallier-Mames, Christophe Clavier, Thomas Fuhr, Aline Gouget, Thomas Icart, Jean-François Misarsky, Marìa Naya-Plasencia, Pascal Paillier, Thomas Pornin, Jean-René Reinhard, Céline Thuillet, and Marion Videau. Shabal, a Submission to NIST's Cryptographic Hash Algorithm Competition. Submission to NIST, 2008.

[BCK96] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In *Advances in Cryptology - CRYPTO '96*, volume 1109 of *LNCS*, pages 1–15. Springer-Verlag, 1996.

[BD06] Eli Biham and Orr Dunkelman. A framework for iterative hash functions: Haifa. In *In Proceedings of Second NIST Cryptographic Hash Workshop*, 2006.

[BD08] Eli Biham and Orr Dunkelman. The SHAvite-3 Hash Function. Submission to NIST (Round 1), 2008.

[BD09] Olivier Billet and Jintai Ding. Overview of cryptanalysis techniques in multivariate public key cryptography. In Massimiliano Sala, Teo Mora, Ludovic Perret, Shojiro Sakata, and Carlo Traverso, editors, *Gröbner bases, coding and cryptography*, pages 263–283. Springer Verlag, 2009.

[BDPA07] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. Sponge functions. Ecrypt Hash Workshop, 2007.

[BDPA08] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. Keccak sponge function family main document. Submission to NIST (Round 1), 2008.

[BDPA10] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. Note on zero-sum distinguishers of Keccak-f. NIST mailing list, 2010.

[Be11a] D. J. Bernstein and T. Lange (editors). eBACS: ECRYPT benchmarking of cryptographic systems, 2011. Accessed 12 January,2011.

[Be11b] Daniel J. Bernstein and Tanja Lange (editors). eBASH:ECRYPT Benchmarking of All Submitted Hashes. measurements of hash functions, 2011. Accessed May 2011.

[Bel67] V. D. Belousov. Osnovi teorii kvazigrup i lup (in russian), 1967. Nauka, Moscow.

[Ber92] T. A. Berson. Differential Cryptanalysis Mod 2^{32} with Applications to MD5. In *Advances in Cryptology - EUROCRYPT '92*, volume 658 of *LNCS*, pages 71–80, 1992.

[Ber07] Daniel J. Bernstein. Better price-performance ratios for generalized birthday attacks. In *SHARCS*, 2007. `http://cr.yp.to/papers.html#genbday`.

[Ber08a] 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.

[Ber08b] Daniel J. Bernstein. Cubehash specification (2.b.1). Submission to NIST (Round 1), 2008.

[BERW08] Andrey Bogdanov, Thomas Eisenbarth, Andy Rupp, and Christopher Wolf. Time-Area Optimized Public-Key Engines: MQ -Cryptosystems as Replacement for Elliptic Curves? In *Cryptographic Hardware and Embedded Systems (CHES)*, volume 5154, pages 145–61. LNCS, 2008.

[BFFP11] C. Bouillaguet, J.-C. Faugère, P.A. Fouque, and L. Perret. Practical cryptanalysis of the identification scheme based on the isomorphism of polynomial with one secret problem. In *Public Key Cryptography – PKC 2011*, volume 6571 of *LNCS*, pages 441–458. Springer, 2011.

[BFJT09] Charles Bouillaguet, Pierre-Alain Fouque, Antoine Joux, and Joana Treger. A family of weak keys in HFE (and the corresponding practical key-recovery). Cryptology ePrint Archive, Report 2009/619, 2009.

[BFP09] Luk Bettale, Jean-Charles Faugère, and Ludovic Perret. Hybrid approach for solving multivariate systems over finite fields. *Journal of Mathematical Cryptology*, volume 3(issue 3):177–197, 2009.

[BFP11] Luk Bettale, Jean-Charles Faugère, and Ludovic Perret. Cryptanalysis of multivariate and odd-characteristic HFE variants. In *Public Key Cryptography – PKC 2011*, volume 6571 of *LNCS*, pages 441–458. Springer, 2011.

[BFS02] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. Complexity study of Gröbner basis computation. Technical report, INRIA, 2002. `http://www.inria.fr/rrrt/rr-5049.html`.

[BFS03] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. Complexity of Gröbner basis computation for semi-regular overdetermined sequences over F2 with solutions in F2. Technical report, Institut national de recherche en informatique et en automatique, 2003.

[BFS04] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In *Proc. International Conference on Polynomial System Solving (ICPSS)*, pages 71–75, 2004.

[BFSY05] Magali Bardet, Jean-Charles Faugère, Bruno Salvy, and Bo-Yin Yang. Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems. In *Proc. of MEGA 2005, Eighth International Symposium on Effective Methods in Algebraic Geometry*, 2005.

[BG81] Charles H. Bennett and John Gill. Relative to a random oracle a, $p^a \neq np^a \neq co - np^a$ with probability 1. *Siam Journal on Computing*, 10:96–113, 1981.

[BJN94] P.B. Bhattacharya, S.K. Jain, and S.R. Nagpaul. *Basic abstract algebra.* Cambridge University Press, 1994.

[BJV04] Thomas Baignres, Pascal Junod, and Serge Vaudenay. How far can we go beyond linear cryptanalysis? In Pil Lee, editor, *Advances in Cryptology - ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 113–128. Springer Berlin / Heidelberg, 2004.

[BK04] Mihir Bellare and Tadayoshi Kohno. Hash function balance and its impact on birthday attacks. In *Advances in Cryptology - EUROCRYPT 04*, volume 3027 of *LNCS*, pages 401–418. Springer-Verlag, 2004.

[BLSS04] Dan Boneh, Ben Lynn, Hovav Shacham, and Hovav Shacham. Short signatures from the weil pairing. *Journal of Cryptology*, pages 297–319, 2004.

[BM97] Mihir Bellare and Daniele Micciancio. A new paradigm for collision-free hashing: Incrementality at reduced cost. In Walter Fumy, editor, *EUROCRYPT*, volume 1233 of *LNCS*, pages 163–192. Springer, 1997.

[BP95] A. Bosselaers and B. Preneel. Integrity Primitives for secure information systems: Final report of RACE integrity primitives evaluation. RIPE-RACE, 1995.

[BPW06a] Johannes Buchmann, Andrei Pyshkin, and Ralf-Philipp Weinmann. A Zero-Dimensional Gröbner Basis for AES-128. In *Fast Software Encryption, 13th International Workshop, FSE 2006*, volume 4047 of *LNCS*, pages 78–88, 2006.

[BPW06b] Johannes Buchmann, Andrei Pyshkin, and Ralf-Philipp Weinmann. Block Ciphers Sensitive to Gröbner Basis Attacks. In David Pointcheval, editor, *CT-RSA*, volume 3860 of *LNCS*, pages 313–331. Springer, 2006.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security*, CCS '93, pages 62–73, New York, NY, USA, 1993. ACM.

[Bra08] Colin Bradbury. Blender: A proposed new family of cryptographic hash algorithms. Submission to NIST, 2008.

[BRS02] John Black, Phillip Rogaway, and Thomas Shrimpton. Black-box analysis of the block-cipher-based hash function constructions from PGV. In *Proceedings of CRYPTO 2002*, volume 2442 of *LNCS*, pages 320–335, 2002.

[BS91] Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In *Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '90, pages 2–21, London, UK, UK, 1991. Springer-Verlag.

[Buc65] Bruno Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal.* PhD thesis, Leopold-Franzens University, 1965.

[Buc79] B. Buchberger. A criterion for Detecting Unnecessary Reductions in the Construction of Gröbner Bases. In *Proceedings of the EUROSAM 79 Symposium on Symbolic and Algebraic Manipulation, Marseille, June 26-28*, volume 72, pages 3–21, London, UK, 1979. Johannes Kepler University Linz, Springer, Berlin - Heidelberg - New York.

[Buc85] B. Buchberger. *Gröbner-Bases: An Algorithmic Method in Polynomial Ideal Theory.* Reidel Publishing Company, Dodrecht - Boston - Lancaster, 1985.

[BWP05] An Braeken, Christopher Wolf, and Bart Preneel. A Study of the Security of Unbalanced Oil and Vinegar Signature Schemes. In Alfred Menezes, editor, *Topics in Cryptology CT-RSA 2005*, volume 3376 of *LNCS*, pages 29–43. Springer Berlin / Heidelberg, 2005.

[CB10] Anne Canteau Christina Boura. Zero-Sum Distinguishers for Iterated Permutations and Application to Keccak-f and Hamsi-256. In *Selected Areas in Cryptography - SAC'10*, volume 6544 of *LNCS*. Springer, 2010.

[CCC$^+$09] Anna Inn-Tung Chen, Ming-Shing Chen, Tien-Ren Chen, Chen-Mou Cheng, Jintai Ding, Eric Li-Hsiang Kuo, Frost Yu-Shuang Lee, and Bo-Yin Yang. SSE Implementation of Multivariate PKCs on Modern x86 CPUs. In *Cryptographic Hardware and Embedded Systems, CHES 2009*, volume 5747 of *LNCS*, pages 33–48. Springer, 2009.

[CD96] Ronald Cramer and Ivan Damgård. New Generation of Secure and Practical RSA-Based Signatures. In *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology - CRYPTO '96*, volume 1109 of *LNCS*, pages 173–185. Springer-Verlag, 1996.

[CD03] Nicolas T. Courtois and Magnus Daum. On the security of HFE, HFEv- and Quartz. In *Proceedings of PKC 2003, volume 2567 of LNCS*, pages 337–350. SpringerVerlag, 2003.

[CDMP05] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård Revisited: How to Construct a Hash Function. In *Advances in Cryptology: CRYPTO'05*, volume 3621 of *LNCS*, pages 430–448. Springer Berlin / Heidelberg, 2005.

[CDN95] G. Carter, E. Dawson, and L. Nielsen. A latin square variation of DES. In *Proc. Workshop of Selected Areas in Cryptography*, Ottawa, Canada, 1995.

[CDS94] J. Cooper, D. Donovan, and J. Seberry. Secret sharing schemes arising from latin squares. *Bulletin of the Institute of Combinatorics and its Applications*, 12(4):33–43, 1994.

[Che09] Lily Chen. *Recommendation for Key Derivation Using Pseudorandom Function.* National Institute of Standards and Technology, 2009.

[CHK$^+$08] Donghoon Chang, Seokhie Hong, Changheon Kang, Jinkeon Kang, Jongsung Kim, Changhoon Lee, Jesang Lee, Jongtae Lee, Sangjin Lee, Yuseop Lee, Jongin Lim, and Jaechul Sung. Arirang. Submission to NIST, 2008.

[CKG10] Yanling Chen, Svein Johan Knapskog, and Danilo Gligoroski. Multivariate quadratic quasigroups (MQQs): Construction, bounds and complexity. In *Inscrypt*, 6th International Conference on Information Security and Cryptology. Science Press of China, October 2010.

[CKPS00] Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *Theory and Application of Cryptographic Techniques*, pages 392–407, 2000.

[CLO05] David Cox, John Little, and Donal O'Shea. *Using Algebraix Geometry.* Springer, 2005.

[CLS06] Scott Contini, Arjen K. Lenstra, and Ron Steinfeld. VSH, an Efficient and Provable Collision-Resistant Hash Function. In *EUROCRYPT'06*, volume 4004 of *LNCS*, pages 165–182. Springer, 2006.

[CMR05] C. Cid, S. Murphy, and M. J. B. Robshaw. Small Scale Variants of the AES. In *Proceedings of FSE 2005*, volume 3557 of *LNCS*, pages 145–162. Springer-Verlag, 2005.

[CMR07] Christophe De Cannière, Florian Mendel, and Christian Rechberger. Collisions for 70-Step SHA-1: On the Full Cost of Collision Search. In Carlisle M. Adams, Ali Miri, and Michael J. Wiener, editors, *Selected Areas in Cryptography*, volume 4876 of *LNCS*, pages 56–73. Springer, 2007.

[CP02] Nicolas Courtois and Josef Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. Cryptology ePrint Archive, Report 2002/044, 2002. `http://eprint.iacr.org/`.

[CS00] Ronald Cramer and Victor Shoup. Signature schemes based on the strong RSA assumption. *ACM Trans. Inf. Syst. Secur.*, 3:161–185, August 2000.

[CSW08] Christophe De Canniere, Hisayoshi Sato, and Dai Watanabe. Hash Function Luffa: Supporting Document. Submission to NIST (Round 1), 2008.

[ÇT10] Çağdaş Çalık and Meltem Sonmez Turan. Message recovery and pseudo-preimage attacks on the compression function of Hamsi-256. Cryptology ePrint Archive, Report 2010/057, 2010.

[CW09] Carlos Cid and Ralf-Philipp Weinmann. Block Ciphers: Algebraic Cryptanalysis and Gröbner Bases. In *Gröbner Bases, Coding, and Cryptography*, pages 307–327. Springer Berlin Heidelberg, 2009.

[Dam87] I. B. Damgård. Collision free hash functions and public key signature schemes. In *Advances in Cryptology–EUROCRYPT '87*, volume 304 of *LNCS*, pages 203–216. Springer-Verlag, 1987.

[Dam89] Ivan Damgård. A design principle for hash functions. In Gilles Brassard, editor, *CRYPTO*, volume 435 of *LNCS*, pages 416–427. Springer, 1989.

[Dav94] Philip J. Davis. *Circulant Matrices.* AMS Chelsea Publishing, 1994.

[dBB93] Bert den Boer and Antoon Bosselaers. Collisions for the Compression Function of MD5. In *EUROCRYPT'93*, volume 765 of *LNCS*, pages 293–304, 1993.

[DBP96] H Dobbertin, A Bosselaers, and B Preneel. RIPEMD-160: A strengthened version. In *Fast Software Encryption (FSE)*, volume 1039 of *LNCS*. Springer-Verlag, 1996.

[DFSS07] Vivien Dubois, Pierre-Alain Fouque, Adi Shamir, and Jacques Stern. Practical cryptanalysis of SFLASH. In *Advances in Cryptology - CRYPTO 2007*, volume 4622 of *LNCS*. Springer-Verlag, 2007.

[DH76a] Whitfield Diffie and Martin E. Hellman. Multiuser cryptographic techniques. In *AFIPS National Computer Conference*, pages 109–112, 1976.

[DH76b] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[DH77] W. Diffie and M. E. Hellman. Special Feature Exhaustive Cryptanalysis of the NBS Data Encryption Standard. *Computer*, 10:74–84, June 1977.

[DK74] J. Denes and A. D. Keedwell. *Latin squares and their applications.* Academic Press, New York :, 1974.

[DK92] J. Dénes and A. D. Keedwell. A new authentication scheme based on latin squares. *Discrete Math.*, 106-107:157–161, 1992.

[DN94] Cynthia Dwork and Moni Naor. An efficient existentially unforgeable signature scheme and its applications. In *Journal of Cryptology*, pages 234–246. Springer-Verlag, 1994.

[Dob98] Hans Dobbertin. One-to-one highly nonlinear power functions on GF(2^{n}). *Appl. Algebra Eng. Commun. Comput.*, 9(2):139–152, 1998.

[DPP98] Ivan Damgård, Torben P. Pedersen, and Birgit Pfitzmann. Statistical secrecy and multibit commitments. *IEEE Transactions on Information Theory*, 44(3):1143–1151, 1998.

[DR98] Joan Daemen and Vincent Rijmen. AES Proposal: Rijndael. In *The First Advanced Encryption Standard Candidate Conference*, 1998.

[DS09] Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. In *Theory and Application of Cryptographic Techniques*, pages 278–299, 2009.

[DS10] Itai Dinur and Adi Shamir. An Improved Algebraic Attack on Hamsi-256. Cryptology ePrint Archive, Report 2010/602, 2010.

[dVMPT09] Francoise Levy dit Vehel, Maria Grazia Marinari, Ludovic Perret, and Carlo Traverso. A survey on polly cracker system. In Massimiliano Sala, Teo Mora, Ludovic Perret, Shojiro Sakata, and Carlo Traverso, editors, *Gröbner bases, coding and cryptography*, pages 263–283. Springer Verlag, 2009.

[DWY07] Jintai Ding, Christopher Wolf, and Bo-Yin Yang. l-Invertible Cycles for Multivariate Quadratic (MQ) Public Key Cryptography. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography*, volume 4450 of *LNCS*, pages 266–281. Springer, 2007.

[DYC^{+}08] Jintai Ding, Bo-Yin Yang, Chia-Hsin Owen Chen, Ming-Shing Chen, and Chen-Mou Cheng. New Differential-Algebraic Attacks and Reparametrization of Rainbow. In *Proceedings of Applied Cryptography and Network Security, 6th International Conference, ACNS 2008*, volume 5037 of *LNCS*, pages 242–257, 2008.

[EG84] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, volume 196 of *LNCS*, pages 10–18. Springer-Verlag New York, Inc., 1984.

[Fau99] Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases (F_4). *Journal of Pure and Applied Algebra*, 139(1-3):61–88, June 1999.

[Fau02] Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5). In *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*, New York, 2002. ACM.

[Fau09] Jean-Charles Faugère. Interactions between computer algebra (Gröbner bases) and cryptology, 2009.

[Fay08] Björn Fay. Meshhash. Submission to NIST, 2008.

[FdVP08] Jean-Charles Faugère, Françoise Levy dit Vehel, and Ludovic Perret. Cryptanalysis of MinRank. In *Advances in Cryptology - CRYPTO 2008*, volume 5157 of *LNCS*, pages 280–296. Springer, 2008.

[FFG08] Ewan Fleischmann, Christian Forler, and Michael Gorski. Classification of the SHA-3 Candidates. Cryptology ePrint Archive, Report 2008/511, 2008.

[FGLM93] J. C. Faugère, P. Gianni, D. Lazard, and T. Mora. Efficient computation of zero-dimensional Gröbner bases by change of ordering. *J. Symb. Comput.*, 16:329–344, October 1993.

[FGS05] Pierre-Alain Fouque, Louis Granboulan, and Jacques Stern. Differential cryptanalysis for multivariate schemes. In *Eurocrypt 2005*, volume 3494 of *LNCS*, pages 341–353. Springer, 2005.

[Fil02] Eric Filiol. A new statistical testing for symmetric ciphers and hash functions. In Robert Deng, Feng Bao, Jianying Zhou, and Sihan Qing, editors, *Information and Communications Security*, volume 2513 of *LNCS*, pages 342–353. Springer Berlin / Heidelberg, 2002.

[Fis03] Marc Fischlin. The Cramer-Shoup Strong-RSA Signature Scheme Revisited. In *Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography: Public Key Cryptography*, PKC '03, pages 116–129, London, UK, 2003. Springer-Verlag.

[FJ03] Jean-Charles Faugère and Antoine Joux. Algebraic cryptanalysis of Hidden Field Equation (HFE) cryptosystems using Gröbner bases. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *LNCS*, pages 44–60. Springer, 2003.

[FLS$^+$08] Niels Ferguson, Stefan Lucks, Bruce Schneier, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jon Callas, and Jesse Walker. The Skein Hash Function Family. Submission to NIST (Round 1), 2008.

[FMRS08] Pierre-Alain Fouque, Gilles Macario-Rat, and Jacques Stern. Key Recovery on Hidden Monomial Multivariate Schemes. In *Advances in Cryptology - EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 19–30. Springer, 2008.

[FØPG10] Jean-Charles Faugère, Rune Steinsmo Ødegård, Ludovic Perret, and Danilo Gligoroski. Analysis of the MQQ public key cryptosystem. In Swee-Huay Heng, Rebecca N. Wright, and Bok-Min Goi, editors, *CANS*, volume 6467 of *LNCS*, pages 169–183. Springer, 2010.

[FOPT10] J.-C. Faugére, A. Otmani, L. Perret, and J.-P. Tillich. Algebraic Cryptanalysis of McEliece Variants with Compact Keys. In *Proceedings of Eurocrypt 2010*, volume 6110 of *LNCS*, pages 279–298. Springer Verlag, 2010.

[FP06a] Jean-Charles Faugére and Ludovic Perret. Cryptanalysis of $2R^-$ Schemes. In *Proceedings of Advances in Cryptology - CRYPTO 2006*, volume 4117 of *LNCS*, pages 357–372. Springer, 2006.

[FP06b] Jean-Charles Faugère and Ludovic Perret. Polynomial Equivalence Problems: Algorithmic and Theoretical Aspects. In *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 30–47. Springer, 2006.

[FP09] Jean-Charles Faugère and Ludovic Perret. On the Security of UOV. Cryptology ePrint Archive, Report 2009/483, September 2009.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, Proceedings*, volume 263 of *LNCS*, pages 186–194. Springer, 1986.

[Fuh10] Thomas Fuhr. Finding Second Preimages of Short Messages for Hamsi-256. In *In Advances in Cryptology - ASIACRYPT 2010, Proceedings*, volume 6477 of *LNCS*, pages 20–37. Springer, 2010.

[GC00] Louis Goubin and Nicolas T. Courtois. Cryptanalysis of the TTM cryptosystem. In *Advances of Cryptology, Asiacrypt 2000*, volume 1976 of *LNCS*, pages 44–57. Springer, 2000.

[GHR99] Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure hash-and-sign signatures without the random oracle. In *Proceedings of the 17th international conference on Theory and application of cryptographic techniques*, EUROCRYPT'99, pages 123–139, Berlin, Heidelberg, 1999. Springer-Verlag.

[GJ79] Michael R. Garey and Davis S. Johnson. *Computers and Intractability. A guide to the theory of NP-Completeness.* Bell Telephone Laberatories, Incorporated, 1979.

[GJS06] Louis Granboulan, Antoine Joux, and Jacques Stern. Inverting HFE Is Quasipolynomial. In *Advances in Cryptology - CRYPTO 2006*, volume 4117 of *LNCS*, pages 345–356. Springer, 2006.

[GK08] 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.

[GKAJ11] Danilo Gligoroski, Svein Johan Knapskog, Jørn Amundsen, and Rune Erlend Jensen. Internationally standardized efficient cryptographic hash function. In Javier Lopez and Pierangela Samarati, editors, *SECRYPT*, pages 426–433. SciTePress, 2011.

[GKK$^+$08] Danilo Gligoroski, Vlastimil Klima, Svein Johan Knapskog, Mohamed El-Hadedy, Jørn Amundsen, and Stig Frode Mjølsnes. Cryptographic Hash Function Blue Midnight Wish. Submission to NIST (Round 1), 2008.

[GKM$^+$08] Praveen Gauravaram, Lars R. Knudsen, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. Grøstl – a SHA-3 candidate. Submission to NIST (Round 1/2), 2008.

[GKS11] Danilo Gligoroski, Svein Johan Knapskog, and Simona Samardjiska. The need for parallel ultra fast cryptographic designs for emerging technologies. In *Workshop on Cryptography for Emerging Technologies and Applications*, November 2011.

[Gla] B. Gladman. SHA1, SHA2, HMAC and Key Derivation in C. url:`http://fp.gladman.plus.com/cryptography_technology/sha/index.htm`.

[Gli05] D. Gligoroski. Candidate one-way functions and one-way permutations based on quasigroup string transformations. Cryptology ePrint Archive, Report 2005/352, 2005.

[Gli09] D. Gligoroski. On a family of minimal candidate one-way functions and one-way permutations. *International Journal of Network Security*, in print 2009.

[Gli10] Danilo Gligoroski. Narrow-pipe SHA-3 candidates differ significantly from ideal random functions defined over big domains. NIST mailing list, 2010. url:`http://people.item.ntnu.no/~danilog/Hash/Non-random-behaviour-narrow-pipe-designs-03.pdf`.

[GLRW10] Jian Guo, San Ling, Christian Rechberger, and Huaxiong Wang. Advanced Meet-in-the-Middle Preimage Attacks: First Results on Full Tiger, and Improved Results on MD4 and SHA-2. In *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 56–75, 2010.

[GM86] R. Gebauer and H. Möller. Buchberger's algorithm and staggered linear bases. In *Symposium on Symbolic and Algebraic Manipulation*, pages 218–221, New York, USA, 1986. ACM Press.

[GMK06] D. Gligoroski, S. Markovski, and L. Kocarev. Edon–$\mathcal{R}$, an infinite family of cryptographic hash functions. In *Second NIST Cryptographic Hash Workshop*, August 2006.

[GMK08a] 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.

[GMK08b] Danilo Gligoroski, Smile Markovski, and Svein Johan Knapskog. Multivariate quadratic trapdoor functions based on multivariate quadratic quasigroups. In *MATH'08: Proceedings of the American Conference on Applied Mathematics*, pages 44–49, Stevens Point, Wisconsin, USA, 2008. World Scientific and Engineering Academy and Society (WSEAS).

[GMK08c] Danilo Gligoroski, Smile Markovski, and Svein Johan Knapskog. Public key block cipher based on multivariate quadratic quasigroups. In Cryptology ePrint Archive, Report 2008/320, 2008.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[GØ09] Danilo Gligoroski and Rune Steinsmo Ødegård. On the complexity of Khovratovich et al.'s preimage attack on EDON-$\mathcal{R}$. Available online, 2009. url:`http://eprint.iacr.org/2009/120.pdf`.

[GØM+09a] Danilo Gligoroski, Rune Steinsmo Ødegård, Marija Mihova, Svein Johan Knapskog, Aleš Drápal, Vlastimil Klima, Jørn Amundsen, and Mohamed El-Hadedy. Cryptographic hash function Edon-$\mathcal{R}'$. In *Proceedings of the 1st International Workshop on Security and Communication Networks*. IEEE, 2009.

[GØM+09b] Danilo Gligoroski, Rune Steinsmo Ødegård, Marija Mihova, Svein Johan Knapskog, Ljupco Kocarev, Aleš Drápal, and Vlastimil Klima. Cryptographic hash function EDON-$\mathcal{R}$. Submission to NIST, 2009.

[GQ88] Louis C. Guillou and Jean-Jacques Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In *Advances in Cryptology - CRYPTO '88*, volume 403 of *LNCS*, pages 216–231. Springer, 1988.

[HHJ08] Shai Halevi, William E. Hall, and Charanjit S. Jutla. The Hash Function Fugue. Submission to NIST, 2008.

[HK06] S. Halevi and H. Krawczyk. Strengthening digital signatures via randomized hashing. In *Proceedings of CRYPTO 2006*, volume 4117 of *LNCS*, pages 41–59, 2006.

[HKC97] M. Bellare H. Krawczyk and R. Canetti. *RFC2104 - HMAC: Keyed-Hashing for Message Authentication.* Internet Engineering Task Force, 1997. url:`http://www.faqs.org/rfcs/rfc2104.html`.

[HW09] Susan Hohenberger and Brent Waters. Short and Stateless Signatures from the RSA Assumption. In *Proceedings of the 29th Annual International Cryptology Conference on Advances in Cryptology - CRYPTO 2009*, volume 5677 of *LNCS*, pages 654–670. Springer-Verlag, 2009.

[IM86] Hideki Imai and Tsutomu Matsumoto. Algebraic methods for constructing asymmetric cryptosystems. In *Proceedings of the 3rd International Conference on Algebraic Algorithms and Error-Correcting Codes*, AAECC-3, pages 108–119, London, UK, 1986. Springer-Verlag.

[Jou04] Antoine Joux. Multicollisions in iterated hash functions. application to cascaded constructions. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA*, volume 3152 of *LNCS*, pages 306–316. Springer, August 2004.

[JP07] 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.

[K̈08] Özgül Kücük. The Hash Function Hamsi. Submission to NIST (Round 1), 2008.

[K̈09a] Özgül Kücük. The hash function Hamsi. Submission to NIST (Round 2), January 2009.

[K̈09b] Özgül Kücük. Reference implementation of Hamsi (round 2). Submission to NIST, January 2009.

[Kal92] B. Kaliski. The MD2 Message-Digest Algorithm. RFC 1319 (Informational), April 1992.

[Kat10] Jonathan Katz. *Digital Signatures*. Springer, 2010.

[KBN09] Dmitry Khovratovich, Alex Biryukov, and Ivica Nikolić. Distinguisher and related-key attack on the full AES-256. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 231–249. Springer, 2009.

[KHPG99] Aviad Kipnis, Hamarpe St. Har Hotzvim, Jacques Patarin, and Louis Goubin. Unbalanced Oil and Vinegar Signature Schemes. In *Advances in Cryptology - EUROCRYPT 1999*, volume 5479 of *LNCS*, pages 206–222. Springer, 1999.

[KK06] John Kelsey and Tadayoshi Kohno. Herding hash functions and the Nostradamus attack. In *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 183–200. Springer, 2006.

[Kli08] Vlastimil Klima. Multicollisions of EDON-$\mathcal{R}$ hash function and other observations. Available online, 2008. url:`http://cryptography.hyperlink.cz/BMW/EDONR_analysis_vk.pdf`.

[KNR10] Dmitry Khovratovich, Ivica Nikolić, and Christian Rechberger. Rotational Rebound Attacks on Reduced Skein. In *ASIACRYPT'10*, volume 6477 of *LNCS*, pages 1–19. Springer, 2010.

[Knu94] Lars R. Knudsen. Truncated and higher order differentials. In Bart Preneel, editor, *FSE*, volume 1008 of *LNCS*, pages 196–211. Springer, 1994.

[Knu95] Lars Knudsen. Truncated and higher order differentials. In Bart Preneel, editor, *Fast Software Encryption*, volume 1008 of *LNCS*, pages 196–211. Springer Berlin / Heidelberg, 1995.

[KNW08] Dmitry Khovratovich, Ivica Nikolić, and Ralf-Philipp Weinmann. Cryptanalysis of EDON-$\mathcal{R}$. Available online, 2008. url:`http://ehash.iaik.tugraz.at/uploads/7/74/Edon.pdf`.

[Kob87] Neal Koblitz. Elliptic Curve Cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.

[KR00] Hugo Krawczyk and Tal Rabin. Chameleon signatures. In *Proceedings of the Network and Distributed System Security Symposium.* The Internet Society, 2000.

[KRT07] Lars R. Knudsen, Christian Rechberger, and Søren Steffen Thomsen. The Grindahl hash functions. In Alex Biryukov, editor, *Fast Software Encryption (FSE) 2007*, volume 4593 of *LNCS*, pages 39 – 57. Springer, 2007.

[KS98] Aviad Kipnis and Adi Shamir. Cryptanalysis of the Oil & Vinegar Signature Scheme. In *CRYPTO '98: Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology*, volume 1462 of *LNCS*, pages 257–266, London, UK, 1998. Springer-Verlag.

[KS99] Aviad Kipnis and Adi Shamir. Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization. In *Advances in Cryptology - CRYPTO '99*, volume 1666 of *LNCS*, pages 19–30. Springer-Verlag, 1999.

[KS04] Masao Kasahara and Ryuichi Sakai. A construction of public key cryptosystem for realizing ciphertext of size 100 bit and digital signature scheme. *IEICE Transactions*, 87-A(1):102–109, 2004.

[KS05] John Kelsey and Bruce Schneier. Second preimages on n-bit hash functions for much less than 2^n work. In *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 474–490. Springer, 2005.

[KSHW98] John Kelsey, Bruce Schneier, Chris Hall, and David Wagner. Secure applications of low-entropy keys. In *Proceedings of the First International Workshop on Information Security*, ISW '97, pages 121–134, London, UK, 1998. Springer-Verlag.

[KW03] Jonathan Katz and Nan Wang. Efficiency improvements for signature schemes with tight security reductions. In *ACM Conference on Computer and Communications Security*, pages 155–164, 2003.

[Lai92] Xuejia Lai. Higher order derivatives and differential cryptanalysis. In Richard Blahut, Daniel Costello Jr., Ueli Maurer, and Thomas Mittelholzer, editors, *Communications and Cryptography*, pages 227–233. Kluwer, 1992.

[LBF08] Gaëtan Leurent, Charles Bouillaguet, and Pierre-Alain Fouque. SIMD Is a Message Digest. Submission to NIST (Round 1), 2008.

[Leu09] Gaëtan Leurent. Key recovery attack against secret-prefix EDON-$\mathcal{R}$. Cryptology ePrint Archive, Report 2009/135, 2009.

[LM02] H. Lipmaa and S. Moriai. Efficient algorithms for computing differential properties of addition. In *Proceedings of FSE 2001*, volume 2355 of *LNCS*, pages 336–350. Springer-Verlag, 2002.

[LM11] Mario Lamberger and Florian Mendel. Higher-Order Differential Attack on Reduced SHA-256. Cryptology ePrint Archive, Report 2011/037, 2011.

[LMM91] X. Lai, J. L. Massey, and S. Murphy. Markov ciphers and differential cryptanalysis. In *Advances in Cryptology, CRYPTO '91*, volume 576 of *LNCS*, pages 17–38. Springer-Verlag, 1991.

[Luc04] S. Lucks. Design principles for iterated hash functions. Cryptology ePrint Archive, Report 2004/253, 2004.

[Luc05] Stefan Lucks. A failure-friendly design principle for hash functions. In *Proceeding of ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 474–494. Springer-Verlag, 2005.

[LWD04] 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.

[MAG] MAGMA. High performance software for algebra, number theory, and geometry — a large commercial software package. url:`http://magma.maths.usyd.edu.au`.

[Man11] Stéphane Manuel. Classification and generation of disturbance vectors for collision attacks against SHA-1. *Des. Codes Cryptography*, 59:247–263, April 2011.

[Mar03] Smile Markovski. Quasigroup string processing and applications in cryptography. In *Proc. 1-st Inter. Conf. Mathematics and Informatics for industry MII 2003, 1416 April, Thessaloniki*, pages 278–290, 2003.

[Max08] Peter Maxwell. The Sgáil Cryptographic Hash Function. Submission to NIST, 2008.

[McE78] Robert J. McEliece. A public-key cryptosystem based on algebraic coding theory. Technical report, Jet Propulsion Lab Deep Space Network Progress report, 1978.

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

[MDBW09] Mohamed Saied Mohamed, Jintai Ding, Johannes Buchmann, and Fabian Werner. Algebraic attack on the MQQ public key cryptosystem. In *CANS '09: Proceedings of the 8th International Conference on Cryptology and Network Security*, pages 392–401, Berlin, Heidelberg, 2009. Springer-Verlag.

[Mer89] Ralph C. Merkle. A certified digital signature. In *Proceedings on Advances in cryptology - CRYPTO '89*, volume 435 of *LNCS*, pages 218–238. Springer-Verlag New York, Inc., 1989.

[Mer90a] R. C. Merkle. One way hash functions and DES, 1990. Based on an unpublished paper from 1979 and his PhD thesis, Stanford, 1979.

[Mer90b] Ralph C. Merkle. A fast software one-way hash function. *J. Cryptology*, 3(1):43–58, 1990.

[MGB99] 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.

[MI88] Tsutomu Matsumoto and Hideki Imai. Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In *Advances in Cryptology – EUROCRYPT 1988*, volume 330 of *LNCS*, pages 419–453. Springer–Verlag, 1988.

[Mil86] Victor S. Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology - CRYPTO '85*, volume 218 of *LNCS*, pages 417–426. Springer-Verlag, 1986.

[MM08] Smile Markovski and Aleksandra Mileva. NaSHA - algorithm specification. Submission to NIST, 2008.

[MMO85] S Matyas, C Meyer, and J Oseas. Generating strong one-way functions with cryptographic algorithm. *IBM Technical Disclosure Bulletin*, 27(10A), 1985.

[MMT92] H. Michael Möller, Teo Mora, and Carlo Traverso. Gröbner bases computation using syzygies. In *Papers from the international symposium on Symbolic and algebraic computation*, ISSAC '92, pages 320–328. ACM, 1992.

[MN09] Florian Mendel and Tomislav Nad. A Distinguisher for the Compression Function of SIMD-512. In Bimal K. Roy and Nicolas Sendrier, editors, *INDOCRYPT*, volume 5922 of *LNCS*, pages 219–232. Springer, 2009.

[MNPN$^+$09] Krystian Matusiewicz, María Naya-Plasencia, Ivica Nikolic, Yu Sasaki, and Martin Schläffer. Rebound Attack on the Full LANE Compression Function. Cryptology ePrint Archive, Report 2009/443, 2009.

[Moh99] T. Moh. A public key system with signature and master key functions. Communications in Algebra, 1999.

[MR95] B.D. McKay and E. Rogoyski. Latin squares of order 10. *Electronic J. Comb.*, 2(3), 1995. `http://ejc.math.gatech.edu:8080/Journal/journalhome.html`.

[MRST09] Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grötl. In Orr Dunkelman, editor, *FSE*, volume 5665 of *LNCS*, pages 260–276. Springer, 2009.

[MS10] Pawel Morawiecki and Marian Srebrny. A SAT-based preimage analysis of reduced Keccak hash functions. Cryptology ePrint Archive, Report 2010/285, 2010.

[MVO96] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996.

[NBK08] Ivica Nikolić, Alex Biryukov, and Dmitry Khovratovich. Hash family LUX - Algorithm Specifications and Supporting Documentation. Submission to NIST, 2008.

[NF09] Peter Novotney and Niels Ferguson. Detectable correlations in Edon-R. Cryptology ePrint Archive, Report 2009/378, 2009.

[Nik09] Ivica Nikolić. Near collisions for the compression function of Hamsi-256. CRYPTO rump session, 2009.

[NIS] NIST. Cryptographic hash project web page. url:`http://csrc.nist.gov/groups/ST/hash/index.html`.

[NIS93a] NIST. Data Encryption Standard. FIPS Publication 46-2, December 1993.

[NIS93b] NIST. FIPS 180, Secure Hash Standard, Federal Information Processing Standard (FIPS), Publication 180. Technical report, DEPARTMENT OF COMMERCE, 1993.

[NIS95] NIST. FIPS 180-1, Secure Hash Standard, Federal Information Processing Standard (FIPS), Publication 180-1. Technical report, DEPARTMENT OF COMMERCE, 1995.

[NIS02] NIST. FIPS 180-2, Secure Hash Standard, Federal Information Processing Standard (FIPS), Publication 180-2. Technical report, DEPARTMENT OF COMMERCE, August 2002.

[NIS07] NIST. Announcing request for candidate algorithm nominations for a new cryptographic hash algorithm (SHA-3) family. Federal Register Notice, 72(112), November 2007. `http://csrc.nist.govgroups/ST/hash/documents/FR_Notice_Nov07.pdf`.

[NIS08a] NIST. *Randomized Hashing for Digital Signatures - Draft NIST Special Publication 800-106.* Federal Information Processing Standards Publication, August, 2008. `http://csrc.nist.gov/publications/drafts/800-106/2nd-Draft_SP800-106_July2008.pdf`.

[NIS08b] NIST. *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`.

[NIS09] NIST. Digital signature standard (DSS). Federal Information Processing Standards (FIPS) Publication #186-3, 2009. Available at `http://www.itl.nist.gov/fipspubs/by-num.htm`.

[NIS11] NIST. Status Report on the Second Round of the SHA-3 Cryptographic Hash Algorithm Competition. Technical report, DEPARTMENT OF COMMERCE, February 2011.

[NN03] Valtteri Niemi and Kaisa Nyberg. *UMTS security.* Wiley, 2003.

[NP10a] Mridul Nandi and Souradyuti Paul. Speeding up the wide-pipe: Secure and fast hashing. In Guang Gong and Kishan Gupta, editors, *Progress in Cryptology - INDOCRYPT 2010*, volume 6498 of *LNCS*, pages 144–162. Springer Berlin / Heidelberg, 2010.

[NP10b] María Naya-Plasencia. Scrutinizing rebound attacks: new algorithms for improving the complexities. Cryptology ePrint Archive, Report 2010/607, 2010.

[NS90] Kazuo Nishimura and Masaaki Sibuya. Probability to meet in the middle. *Journal of Cryptology*, 2:13–22, 1990.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *Symposium on Theory of Computing*, pages 33–43, 1989.

[ONH08] Sean O'Neil, Karsten Nohl, and Luca Henzen. Enrupt hash function specification. Submission to NIST, 2008.

[OS90] H. Ong and Claus-Peter Schnorr. Fast signature generation with a fiat shamir-like scheme. In *Advances in Cryptology - EUROCRYPT '90*, volume 473 of *LNCS*, pages 432–440. Springer, 1990.

[Pat95] Jacques Patarin. Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt–88. In Don Coppersmith, editor, *Advances in Cryptology – CRYPT0'95*, volume 963 of *LNCS*, pages 248–261. Springer Berlin / Heidelberg, 1995.

[Pat96] Jacques Patarin. Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms. In *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 33–48. Springer-Verlag, 1996.

[Pat97] Jacques Patarin. The Oil & Vinegar signature scheme. In *Proceedings of Dagstuhl workshop on cryptography*, 1997.

[Pat00] Jacques Patarin. Cryptanalysis of the Matsumoto and Imai public key scheme of Eurocrypt 98. *Des. Codes Cryptography*, 20:175–209, June 2000.

[PBB10] Albrecht Petzoldt, Stanislav Bulygin, and Johannes Buchmann. CyclicRainbow - A multivariate Signature Scheme with a Partially Cyclic Public Key based on Rainbow. Cryptology ePrint Archive, Report 2010/424, 2010.

[PCG01a] Jacques Patarin, Nicolas Courtois, and Louis Goubin. Flash, a fast multivariate signature algorithm. In David Naccache, editor, *Topics in Cryptology CT-RSA 2001*, volume 2020 of *LNCS*, pages 298–307. Springer Berlin / Heidelberg, 2001.

[PCG01b] Jacques Patarin, Nicolas Courtois, and Louis Goubin. Quartz, 128-bit long digital signatures. In *Proceedings of the 2001 Conference on Topics in Cryptology: The Cryptographer's Track at RSA*, volume 2020 of *CT-RSA 2001*, pages 282–297. Springer-Verlag, 2001.

[Per05] Ludovic Perret. A fast cryptanalysis of the isomorphism of polynomials with one secret problem. In *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 354–370. Springer, 2005.

[Per08] Ludovic Perret. Personal e-mail communication with Danilo Gligoroski, 2008.

[Pey07] Thomas Peyrin. Cryptanalysis of Grindahl. In *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 551–567. Springer, 2007.

[PGC98] Jacques Patarin, Louis Goubin, and Nicolas Courtois. C^*_{-+} and HM : Variations Around Two Schemes of T. Matsumoto and H. Imai. In Kazuo Ohta and Dingyi Pei, editors, *Advances in Cryptology – ASIACRYPT'98*, volume 1514 of *LNCS*, pages 35–50. Springer Berlin / Heidelberg, 1998.

[PGV93] Bart Preneel, René Govaerts, and Joos Vandewalle. Hash functions based on block ciphers: a synthetic approach. In *Proceedings of the 13th annual international cryptology conference on Advances in cryptology - CRYPTO'93*, volume 773 of *LNCS*, pages 368–378. Springer-Verlag New York, Inc., 1993.

[Pys08] Andrey Pyshkin. *Algebraic Cryptanalysis of Block Ciphers Using Groebner Bases.* PhD thesis, TU Darmstadt, July 2008.

[RDS98] Inc. RSA Data Security. PKCS#1 version 2.1:RSA cryptography standard, 1998. Available at `http://www.rsa.com/rsalabs`.

[Riv90] Ronald L. Rivest. The MD4 Message Digest Algorithm. In Alfred Menezes and Scott A. Vanstone, editors, *Proceedings of Advances in Cryptology – CRYPTO'90*, volume 537 of *LNCS*, pages 303–311. Springer, 1990.

[Riv92] Ronald L. Rivest. The MD5 Message-Digest Algorithm (RFC 1321). url:`http://www.ietf.org/rfc/rfc1321.txt?number=1321`, 1992.

[RMG$^+$00] 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.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21:120–126, February 1978.

[RSN$^+$01] Andrew Rukhin, Juan Soto, James Nechvatal, Elaine Barker, Stefan Leigh, Mark Levenson, David Banks, Alan Heckert, James Dray, San Vo, Andrew Rukhin, Juan Soto, Miles Smid, Stefan Leigh, Mark Vangel, Alan Heckert, James Dray, and Lawrence E Bassham Iii. A statistical test suite for random and pseudorandom number generators for cryptographic applications, 2001. Published by NIST. url: `http://csrc.nist.gov/publications/nistpubs/800-22-rev1a/SP800-22rev1a.pdf`.

[RTV10] Vincent Rijmen, Denis Toz, and Kerem Varc. Rebound Attack on Reduced-Round Versions of JH. In *FSE*, volume 6147 of *LNCS*, pages 286–303. Springer, 2010.

[SAB09] Bhupendra Singh, Lexy Alexander, and Sanjay Burman. On algebraic relations of Serpent S-boxes. Cryptology ePrint Archive, Report 2009/038, 2009.

[SBA09] Rajesh P Singh, B.K.Sarma, and A.Saikia. Public key cryptography using permutation p-polynomials over finite fields. Cryptology ePrint Archive, Report 2009/208, 2009. `http://eprint.iacr.org/`.

[SCG11] Simona Samardjiska, Yanling Chen, and Danilo Gligoroski. Construction of Multivariate Quadratic Quasigroups (MQQs) in Arbitrary Galois Fields. In *Proceeding of IAS 2011, Malacca, Malaysia, Dec. 5-8, 2011*, 2011.

[sha] The SHA-3 Zoo. url:`http://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo`.

[Sha49] C. E. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28(4):656—715, 1949.

[Sha93] Adi Shamir. Efficient signature schemes based on birational permutations. In *Proceedings of CRYPTO'93*, volume 773 of *LNCS*, pages 1–12. Springer-Verlag, 1993.

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26:1484–1509, October 1997.

[Sin99] Simon Singh. *The Code Book: The Evolution of Secrecy from Mary, Queen of Scots, to Quantum Cryptography.* Doubleday, New York, NY, USA, 1st edition, 1999.

[SK98] Takeshi Shimoyama and Toshinobu Kaneko. Quadratic Relation of S-box and its Application to the Linear Attack of Full Round DES. In *Advances in Cryptology - Crypto'98*, volume 1462 of *LNCS*, pages 200–211. Springer-Verlag, 1998.

[SKPI07] Makoto Sugita, Mitsuru Kawazoe, Ludovic Perret, and Hideki Imai. Algebraic Cryptanalysis of 58-Round SHA-1. In Alex Biryukov, editor, *Fast Software Encryption (FSE)*, volume 4593 of *LNCS*, pages 349–365. Springer, 2007.

[SLdW07] Marc Stevens, Arjen K. Lenstra, and Benne de Weger. Chosen-Prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities. In Moni Naor, editor, *EUROCRYPT'07*, volume 4515 of *LNCS*, pages 1–22. Springer, 2007.

[SMG10] S. Samardjiska, S. Markovski, and D. Gligoroski. Multivariate quasigroups defined by t-functions. In *Proceedings of SCC2010 - The 2nd International Conference on Symbolic Computation and Cryptography*, 2010.

[Smi07] J. D. H. Smith. *An introduction to quasigroups and their representations.* Chapman & Hall/CRC, 2007.

[Ste06] Till Stegers. Faugerè's F5 algorithm revisited. Cryptology ePrint Archive, Report 2006/404, 2006.

[SV94] C.-P. Schnorr and S. Vaudenay. Black Box Cryptoanalysis of Hash Networks Based on Multipermutations. In *Proceedings of EUROCRYPT 1994*, volume 950 of *LNCS*, pages 47–57. Springer, 1994.

[Tho07] S. S. Thomsen, May 2007. Personal communication with S. S. Thomsen.

[TU10] Meltem Sönmez Turan and Erdener Uyan. Practical Near-Collisions for Reduced Round Blake, Fugue, Hamsi and JH. Second SHA-3 Candidate Conference, 2010.

[Wag99] David Wagner. The boomerang attack. In Lars R. Knudsen, editor, *FSE*, volume 1636 of *LNCS*, pages 156–170. Springer, 1999.

[Wag02] David Wagner. A generalized birthday problem. In *Proceedings of Advances in Cryptology - CRYPTO 2002*, volume 2442 of *LNCS*, pages 288–303. Springer, 2002.

[Wat05] Brent Waters. Efficient identity-based encryption without random oracles. In *Proceedings of Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, 2005.

[WBP04] Christopher Wolf, An Braeken, and Bart Preneel. Efficient Cryptanalysis of RSE(2)PKC and RSSE(2)PKC. In *Conference on Security in Communication Networks – SCN 2004*, volume 3352 of *LNCS*, pages 294–309. Springer, 2004. Extended version:`http://eprint.iacr.org/2004/237`.

[Wil75] M. V. Wilkes. *Time Sharing Computer Systems.* Elsevier Science Inc., New York, NY, USA, 1975.

[Win84] Robert S. Winternitz. A Secure One-Way Hash Function Built from DES. In *IEEE Symposium on Security and Privacy*, pages 88–90, Los Alamitos, CA, USA, 1984. IEEE Computer Society.

[WLF$^+$05] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the hash functions MD4 and RIPEMD. In Ronald Cramer, editor, *EUROCRYPT'05*, volume 3494 of *LNCS*, pages 1–18. Springer, 2005.

[WN94] 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.

[Wol06] Christopher Wolf. Multivariate Quadratic Public Key Systems: Introduction, Nomenclatura, History (presentation), January 2006.

[WP05] Christopher Wolf and Bart Preneel. Taxonomy of Public Key Schemes based on the problem of Multivariate Quadratic equations. IACR Eprint archive, 2005. url:`http://eprint.iacr.org/2005/077`.

[Wu09] Hongjun Wu. The Hash Function JH. Submission to NIST (Round 1/2), 2009.

[WWJW09] Meiqin Wang, Xiaoyun Wang, Keting Jia, and Wei Wang. New pseudo-near-collision attack on reduced-round of Hamsi-256. Cryptology ePrint Archive, Report 2009/484, 2009.

[WY05] Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In Ronald Cramer, editor, *EUROCRYPT'05*, volume 3494 of *LNCS*, pages 19–35. Springer, 2005.

[WYY05a] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In Victor Shoup, editor, *Proceedings of CRYPTO 2005*, volume 3621 of *LNCS*, pages 17–36. Springer, 2005.

[WYY05b] Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin. Efficient collision search attacks on SHA-0. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 1–16. Springer, 2005.

[Xu08] Zijie Xu. Dynamic SHA2. Submission to NIST, 2008.

[YC05] Bo-Yin Yang and Jiun-Ming Chen. Building Secure Tame-like Multivariate Public-Key Cryptosystems: The New TTS. In *Information Security and Privacy, 10th Australasian Conference, ACISP 2005*, volume 3574 of *LNCS*, pages 518–531. Springer, 2005.

[YCCC06] Bo-Yin Yang, Chen-Mou Cheng, Bor-Rong Chen, and Jiun-Ming Chen. Implementing minimized multivariate PKC on low-resource embedded systems. In *Security in Pervasive Computing, SPC 2006*, volume 3934 of *LNCS*, pages 73–88. Springer, 2006.

[Yuv79] Gideon Yuval. How to Swindle Rabin. *Cryptologia*, 3:187–189, 1979.

[ZM02] Lintao Zhang and Sharad Malik. The Quest for Efficient Boolean Satisfiability Solvers. In *Proceedings of the 14th International Conference on Computer Aided Verification*, CAV '02, pages 17–36. Springer-Verlag, 2002.

[ZPS93] Yuliang Zheng, Josef Pieprzyk, and Jennifer Seberry. Haval - a one-way hashing algorithm with variable length of output. In Jennifer Seberry and Yuliang Zheng, editors, *ASIACRYPT 1993*, volume 718 of *LNCS*, pages 83–104. Springer, 1993.