Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DISERTAČNÍ PRÁCE

RNDr. Daniel Joščák

Využití algebraických a kombinatorických metod při studiu hašovacích funkcí

Katedra algebry

Vedoucí disertační práce: Doc. RNDr. Jiří Tůma, DrSc.

Studijní program: Matematika

Studijní obor: Algebra, teorie čísel a matematická logika

Praha 2015

Děkuji Doc. RNDr. Jiřímu Tůmovi, DrSc. za cenné rady, náměty a nespočetné hodiny konzultací, kterými přispěl k napsání této práce. Děkuji Pánu Bohu, své rodině, všem přátelům, spoluautorům publikací a kolegům za jejich podporu, trpělivost a energii pro její sepsání. Dále děkuji svým nadřízeným ve firmách S.ICZ, First Data a NN za podporu v čase mého studia.

Věnováno otci Miroslavovi Joščákovi.

Název práce: Využití algebraických a kombinatorických metod při studiu hašovacích funkcí

Autor: Daniel Joščák

Katedra: Katedra algebry

Vedoucí diplomové práce: Doc. RNDr. Jiří Tůma, DrSc.

Abstrakt: Práce shrnuje autorův výzkum v oboru hašovacích funkcí v průběhu jeho doktorského studia. První část práce představuje zobecněnou teorii rovnic sestavených ze dvou základních stavebních kamenů kryptografických primitiv: modulárního sčítaní a exkluzivní disjunkce. Druhá a třetí část byly napsány po Wangové zveřejnění kolizí v MD5 a ukazují, že drobné modifikace této hašovací funkce nefungují. Tyto dvě částí popisují kolize pro 3C a 3C+ konstrukce hashovacích funkcí, které navrhl Gauravaram, a pro tzv. "feedback ring-iterative structure" konstrukci navrženou Su a kol. Výsledky byly publikovány na konferencích ICISC 2006 a SPI 2007. Poslední část představuje nový typ kolizí pro MD5 s nově navrženými rozdíly v kolidujících zprávách. Výsledek byl publikován ve sborníku konference Indocrypt 2008.

Klíčová slova: AX-rovnice, hašovací funkce, kolize, MD5, XOR, ADD

Title: Algebraic and combinatorial methods for the study of hash functions

Author: Daniel Joščák

Department: Department of Algebra

Supervisor: Doc. RNDr. Jiří Tůma, DrSc.

Abstract: The work summarizes author's research during the doctoral studies in the field of hash functions. The first part of the thesis presents a generalised theory of equations built from two basic building blocks of cryptographic primitives: modular addition and eXclusive OR. In particular we study AX-equations of depth 1. The second and third sections were written after Wang's publication of collisions in MD5 and show that minor modifications of the hash function does not work. We present collisions in the 3C and 3C+ constructions of hash function suggested by Gauravaram and feedback ring-iterative structure by Su et al. The results were published at the conferences ICISC 2006 and SPI 2007. The last part presents a newly constructed type of collisions in MD5 with a newly proposed message differences. The result was published and presented at the conference Indocrypt 2008.

Keywords: AX-equations, hash functions, collisions, MD5, XOR, ADD

Contents

Introduction

Cryptographic hash functions play an important role in IT security protocols and algorithms. The hash function takes as input an arbitrary long binary message and maps it to a binary output of a fixed length. The length of the output message is the length of the hash function. The output is called the hash value of the input. The hash value is also known as a digital fingerprint of the input message. Just like a fingerprint can be used to identify (almost) uniquely a person, the hash value of a message can be used to identify (almost) uniquely the message.

There are several properties a well designed hash function must have.

Collision resistance. A hash function H is collision resistant if it is hard to find two distinct inputs that hash to the same output (that is, two distinct inputs m_1 and m_2, such that $H(m_1) = H(m_2)$). Definition of hard to find means that it is computationally infeasible to find a colliding messages in the sense of computational complexity theory. Every hash function with more inputs than outputs will necessarily have collisions. Collision resistance doesn't mean that no collisions exist; simply that they are hard to find. The birthday paradox sets an upper bound on collision resistance: if a hash function produces n bits of output, an attacker can find a 2 distinct messages with the same hash with probability 1/2 by performing $\sqrt{\pi/2} \cdot 2^{n/2}$ hash operations on average. If there is a faster method than this brute force attack, it is considered a flaw in the hash function.

First preimage resistance. A hash function H is said to be first preimage resistant (sometimes only preimage resistant) if given h it is hard to find any m such that $h = H(m)$.

Second preimage resistance. A hash function H is said to be second preimage resistant if given an input m_1, it is hard to find another input, m_2 ($m_1 \neq m_2$) such that $H(m_1) = H(m_2)$. A preimage attack differs from a collision attack in that there is a fixed hash or message that is being attacked and in its complexity. Optimally, a preimage attack on an n-bit hash function will take an order of 2^n operations to be successful.

Additionally the following properties are often required.

Efficiency. Computation of a hash function must be efficient or speed matters. Hash functions are widely deployed in many applications and it is important to have fast implementation on different architectures.

Resistance to length-extension attacks. Given $H(m)$ and length of m, it is computationally infeasible to compute $H(m||m')$ for any m' of a positive length. Symbol $||$ denotes concatenation of the messages.

HMAC construction. A hash function must have at least one construction to support $HMAC$ (or alternative MAC construction) as a pseudorandom function (PRF) i.e. it is hard to distinguish $HMAC_K$ based on H from a random function.

Memory requirements and code size. It is important for implementation on various embedded systems such as smart cards that a hash function can work within a memory of various limited environments.

Advances in cryptanalysis of hash functions

In 2004 a group of researchers led by Xiaoyun Wang (Shandong University, China) presented real collisions in MD5 and other hash functions at the rump session of Crypto conference and they explained the method in [29]. In 2006 the same group presented a collision attack on SHA–1 in [28] and since then a lot of progress in collision finding algorithms has been made. One of the results of this research was a competition for a new hash standard which started in 2007 and ended in 2012 by choosing of a new SHA-3 standard.

This thesis summrizes the following topics:

- AX Functions and their known properties as building blocks of hash functions.
- Multi-block Collisions in Hash functions based on 3C and 3C+ Enhancements of the Merkle-Damgård [8] published at International Conference on Information Security and Cryptology (ICISC) 2006.
- Beyond the MD5 Collisions [7] published at conference Security and Protection of Information in 2007.
- A New Type of 2-Block Collisions in MD5 [26] published at Indocrypt 2008

AX-Eqautions In this section we present some properties of two basic building blocks of hash functions: modular addition and eXclusive OR known also as XOR. These two functions together with a word rotation ($\lll$) are often referred of ARX construction and are frequently used because their efficiency and simple implementation. Computers well support these operations on 32 and 64-bit words which leads to very fast and optimal processing. Modular addition provides a non-linearity, word rotation provides diffusion within a single word and XOR provides diffusion between words and linearity. The use of modular addition also reduces the memory requirements that could be needed otherwise for substitution box table lookups or other definitions of functions with nonlinear properties needed. This gives additional points to popularity of ARX constructions as they provide more resistance against some of the possible side channels. We omitted a word rotation operation in our study, therefore the name AX-functions. We present a theorem which describes a large class of AX-equations for which the solubility condition depends only on pairs of subsequent vectors of parameters of the equations. The theorem also gives a precise value of the probability that a randomly selected unknowns solve AX-equation from this class. There are additional classes of AX-equations where we encourage future work (e.g. quadratic case and general recursion).

Multi-block Collisions in Hash functions based on 3C and 3C+ Enhancements of the Merkle-Damgård. The MD5 hash function was still widely used right after Wang's' collision discovery and an obvious question occurred. Is it possible to repair MD5 or SHA-1 without additional investments for their complete removal? There were several proposals how to repair the function. Gauravaram et al [5] have proposed a slight modification to the Merkle-Damgård construction for an improved protection against many known attacks on Merkle-Damgård based hash functions. Their idea was to add additional registers that would collect xors of all chaining variables. After the message is processed the content of additional registers is padded to provide one more message block and the extra block is used as an input for the last calculation of the compression function. Thus the original Merkle-Damgård construction remains and the extra security is supposed to be provided by the additional registers. We described in our paper Multi-block Collisions in Hash functions based on 3C and 3C+ Enhancements of the Merkle–Damgård Construction [8]. We have shown that improvements such as 3C constructions do not work and that proofs provided in [5] are based on strong assumptions which are not valid anymore. We emphasized that random properties in compression function of MD5 are no more usable for cryptographic application. The same could be applied for the case of SHA-1 function because theoretical collision searching algorithms have the same structure as in the case of MD5.

Beyond the MD5 Collisions. In this article we summarize results and a state of art of collision resistance of MD5. We recall known collision searching methods, describe and compare 3 concrete algorithms by Vlastimil Klíma, Marc Stevens and ourselves. We give a method how to theoretically calculate the complexity of the algorithms which is not based on the measured tests but which can be verified by these tests. We show that collision attack on MD5 (and also SHA-0, SHA-1) can be easily extended to the extensieons of Merkle–Damgård construction like 3C and 3C+ constructions if the same compression function is used. We present similar faults in the concept of feedback ring-iterative structure suggested in [19] and present proof of concept how collisions in this structure could look like. We discourage from usage of compression function from MD5 in any other similar extensieon of Merkle–Damgård construction.

A New Type of 2-Block Collisions in MD5. A lot of progress was made after researchers studied in depth the structure of collisions introduced by Wang in [29]. One direction of research was optimizing the collision searching algorithms to find colliding messages faster. Examples of these improvements could be found in e.g. [10], [23], [6]. Another direction of the research tried to find a different type of collisions as so far the colliding messages had the same differences e.g. [25], [20], [21]. In our contribution we combine both directions. We present a different type of collisions for MD5 which were found by a new differential path and then used the collision searching techniques to find colliding messages with a new differences. To construct a new differential path for MD5 function we have reimplemented Stevens algorithm [25] and extended it with our experience from various collision searching techniques. As the confirmation that our method works we presented a colliding pair of massages.

1 AX-Equations

1.1 Introduction

The method of differential cryptanalysis was first published by Biham and Shamir in [2] in connection with their study of the security of DES. Since then differential cryptanalysis has become one of the main tools for investigating security of various cryptographic primitives like ciphers, hash functions, message authentication codes, pseudorandom generators, etc.

The method can be described briefly in the following way. Let $(G, \oplus)$ and $(H, +)$ be two finite Abelian groups and $f : G \to H$ an arbitrary mapping. We choose some $\alpha \in G$ and consider the random variable $f(x \oplus \alpha) - f(x) \in H$, where $x \in G$ is uniformly selected. If the probability of some $\beta \in H$ is significantly higher then $|H|^{-1}$, then we say that a pair (α, β) is a *good differential* for f. The probability of $\beta = f(x \oplus \alpha) - f(x)$ is called the probability of the differential (α, β). Similarly a pair (α, β) is a *bad differential* for f if the probability of $\beta = f(x \oplus \alpha) - f(x)$ is significantly less then $|H|^{-1}$. In particular, a pair (α, β) is an *impossible differential* for f if the probability of $\beta = f(x \oplus \alpha) - f(x)$ is equal to 0. Differential cryptanalysis takes advantage of existence of good or bad differentials for a mapping f. The differential (α, β) is sometimes denoted as

$$\alpha \xrightarrow{f} \beta$$

and f is usually omitted when it is clear from the context.

Obviously, the concept of a good differential makes sense only when f is not a group homomorphism. Cryptographic algorithms often use the group $(\mathbb{Z}_2)^p$ with addition $\oplus$ over $\mathbb{Z}_2$, i.e. the additive group of the linear space $(\mathbb{Z}_2)^p$ over $\mathbb{Z}_2$. This operation is sometimes called *xor*. In some cases the algorithm uses modular addition $+$ with module 2^p as the main source of nonlinearity of the algorithm. In such a case, the operation $+ : (\mathbb{Z}_2)^p \times (\mathbb{Z}_2)^p \to (\mathbb{Z}_2)^p$ plays the role of the mapping f from the previous paragraph. Estimating the probability of a differential $(\alpha, \beta) \xrightarrow{+} \gamma$ leads to the equation

$$(x \oplus \alpha) + (y \oplus \beta) = (x + y) \oplus \gamma \ ,$$

where $x, y \in (\mathbb{Z}_2)^p$. We are interested in the probability that uniformly selected $x, y \in (\mathbb{Z}_2)^p$ satisfy the equation. Of course, this probability depends on the parameters α, β, γ.

1.2 Notation and preliminaries

Any element $(a_{p-1}, \ldots, a_0) \in (\mathbb{Z}_2)^p$ can be interpreted also as a natural number

$$a_{p-1}2^{p-1} + \cdots + a_1 2^1 + a_0 \in \mathbb{Z}_{2^p} \ .$$

This interpretation allows to define the modular addition $+$ of the elements of $\mathbb{Z}_2^p$ as the addition in the quotient ring $\mathbb{Z}_{2^p}$ of integers modulo 2^p.

We will use the vector notation $\mathbf{x} = (x_{p-1}, \ldots, x_0)$ for the elements of $(\mathbb{Z}_2)^p$. It is a well-known fact that the modular addition $\mathbf{x}+\mathbf{y}$ of two vectors from $(\mathbb{Z}_2)^p$ can be expressed in the form

$$\mathbf{x} + \mathbf{y} = \mathbf{x} \oplus \mathbf{y} \oplus \mathbf{c} \ ,$$

where $\mathbf{c} = (c_{p-1}, \ldots, c_0)$ is the *carry vector* defined by $c_0 = 0$ and

$$c_i = x_{i-1}y_{i-1} \oplus c_{i-1}(x_{i-1} \oplus y_{i-1}) \tag{1}$$

for $i = 1, ..., p-1$. Notice that the i-th carry c_i depends only on the bits $x_0, \ldots, x_{i-1}$ and $y_0, \ldots, y_{i-1}$ of the vectors $\mathbf{x}$, $\mathbf{y}$.

1.3 Examples of equations

We start with a list of equations. Most of them were already studied by various authors.

Example 1.1. The equation

$$(\mathbf{x} \oplus \boldsymbol{\alpha}) + (\mathbf{y} \oplus \boldsymbol{\beta}) \quad = \quad (\mathbf{x} + \mathbf{y}) \oplus \boldsymbol{\gamma} \tag{2}$$

was comprehensively studied by Lipmaa and Moriai in [14]. The authors presented an efficient algorithm for deciding if the equation has a solution for given parameters $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$ and for calculating the probability that a randomly selected vectors $\mathbf{x}$ and $\mathbf{y}$ from $(\mathbb{Z}_2)^p$ satisfy the equation. We will write the equation (2) in the form

$$[(\mathbf{x} \oplus \boldsymbol{\alpha}) + (\mathbf{y} \oplus \boldsymbol{\beta})] \oplus (\mathbf{x} + \mathbf{y}) \oplus \boldsymbol{\gamma} \quad = \quad \mathbf{0} \ . \tag{3}$$

Example 1.2. The equation

$$\mathbf{x} + \boldsymbol{\alpha} \quad = \quad \mathbf{x} \oplus \boldsymbol{\beta} \tag{4}$$

was studied by Leurent and Thomsen in [12] in connection with cryptanalysis of the BMW candidate for a new hash standard SHA-3. They again calculated the probability that a randomly selected vector $\mathbf{x}$ from $(\mathbb{Z}_2)^p$ solves the equation (4) for given parameters $\boldsymbol{\alpha}, \boldsymbol{\beta}$. Our form of the equation is

$$(\mathbf{x} + \boldsymbol{\alpha}) \oplus \mathbf{x} \oplus \boldsymbol{\beta} \quad = \quad \mathbf{0} \ . \tag{5}$$

Example 1.3. The equation

$$(\mathbf{x} + \boldsymbol{\alpha}) \oplus (\mathbf{y} + \boldsymbol{\beta}) \quad = \quad (\mathbf{x} \oplus \mathbf{y}) + \boldsymbol{\gamma} \tag{6}$$

was studied by Lipmaa et al. in [15] as the dual equation to (2). Also in this case they calculated the probability that a randomly selected vectors $\mathbf{x}$ and $\mathbf{y}$ from $(\mathbb{Z}_2)^p$ solve the equation (6) for given parameters $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$. We will study it in the form

$$(\mathbf{x} + \boldsymbol{\alpha}) \oplus (\mathbf{y} + \boldsymbol{\beta}) \oplus [(\mathbf{x} \oplus \mathbf{y}) + \boldsymbol{\gamma}] = \mathbf{0} \ . \tag{7}$$

Example 1.4. The equation

$$(\mathbf{x} \oplus \boldsymbol{\alpha}) + \boldsymbol{\beta} \;=\; (\mathbf{x} + \boldsymbol{\beta}) \oplus \boldsymbol{\alpha} \tag{8}$$

was formulated by Markova in her Master thesis [16] in connection with her attempt to fill in some gaps in the paper [4] analyzing two other candidates for SHA-3. She found examples of parameters $\boldsymbol{\alpha}, \boldsymbol{\beta}$ for which the equation has no solution, despite an unaproved assumption of [4] that such an equation is always solvable. We will rewrite (8) as

$$[(\mathbf{x} \oplus \boldsymbol{\alpha}) + \boldsymbol{\beta}] \oplus (\mathbf{x} + \boldsymbol{\beta}) \oplus \boldsymbol{\alpha} \;=\; \mathbf{0}\ . \tag{9}$$

Example 1.5. We will also consider the equation

$$(\mathbf{x} + \boldsymbol{\alpha}) \oplus (\mathbf{y} + \boldsymbol{\beta}) \oplus (\mathbf{x} + \mathbf{y}) \oplus \boldsymbol{\gamma} \;=\; \mathbf{0}\ . \tag{10}$$

We are not aware of any paper where this equation was studied.

1.4 A general form of equations

All the previous equations combine modular addition and xor, some unknown vectors from $(\mathbb{Z}_2)^p$ and some parameters. Such equations are called *AX-equations.* Each modular addition adds two expressions involving only unknowns, parameters and the operation xor. AX-equations of this type will be called *of depth 1.* The general form of AX-equations of depth 1 can be formulated as follows.

We assume that the equation has N unknowns $\tilde{\mathbf{x}}_1^T, \dots, \tilde{\mathbf{x}}_N^T \in (\mathbb{Z}_2)^p$ and we will write them as rows of a single unknown matrix

$$X = \begin{pmatrix} \tilde{\mathbf{x}}_1^T \\ \tilde{\mathbf{x}}_2^T \\ \vdots \\ \tilde{\mathbf{x}}_N^T \end{pmatrix} = \begin{pmatrix} x_{1,p-1} & x_{1,p-2} & \cdots & x_{1,0} \\ x_{2,p-1} & x_{2,p-2} & \cdots & x_{2,0} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N,p-1} & x_{N,p-2} & \cdots & x_{N,0} \end{pmatrix}$$

of type $N \times p$ over $\mathbb{Z}_2$.

Similarly, the parameters $\tilde{\boldsymbol{\delta}}_1^T, \dots, \tilde{\boldsymbol{\delta}}_M^T \in \mathbb{Z}_2^p$ will be written as rows of a single parameter matrix

$$\Delta = \begin{pmatrix} \tilde{\boldsymbol{\delta}}_1^T \\ \tilde{\boldsymbol{\delta}}_2^T \\ \vdots \\ \tilde{\boldsymbol{\delta}}_M^T \end{pmatrix} = \begin{pmatrix} \delta_{1,p-1} & \delta_{1,p-2} & \cdots & \delta_{1,0} \\ \delta_{2,p-1} & \delta_{2,p-2} & \cdots & \delta_{2,0} \\ \vdots & \vdots & \ddots & \vdots \\ \delta_{M,p-1} & \delta_{M,p-2} & \cdots & \delta_{M,0} \end{pmatrix}$$

of type $M \times p$ over $\mathbb{Z}_2$.

We will denote the number of modular additions in the equation by S. If the equation is of depth 1, then the left summand in the s-th modular addition can be written as

$$l_{s,1}\tilde{\mathbf{x}}_1^T \oplus \cdots \oplus l_{s,N}\tilde{\mathbf{x}}_N^T \oplus p_{s,1}\tilde{\boldsymbol{\delta}}_1^T \oplus \cdots \oplus p_{s,M}\tilde{\boldsymbol{\delta}}_M^T \ ,$$

where $l_{s,j}, p_{s,k} \in \mathbb{Z}_2$ for each $s = 1, \dots, S$. The coefficient $l_{s,j} = 1$ if and only if the unknown $\tilde{\mathbf{x}}_j$ does appear in the left summand of the s-th modular addition. Also $p_{s,k} = 1$ if and only if the parameter $\tilde{\boldsymbol{\delta}}_k$ does appear in the left summand of the s-th modular addition.

Similarly, the right summand of the s-th modular addition can be written as

$$r_{s,1}\tilde{\mathbf{x}}_1^T \oplus \cdots \oplus r_{s,N}\tilde{\mathbf{x}}_N^T \oplus q_{s,1}\tilde{\boldsymbol{\delta}}_1^T \oplus \cdots \oplus q_{s,M}\tilde{\boldsymbol{\delta}}_M^T \ ,$$

where $r_{s,j}, q_{s,k} \in \mathbb{Z}_2$.

So we can write the s-th modular addition as

$$(\tilde{\mathbf{l}}_s^T X \oplus \tilde{\mathbf{p}}_s^T \Delta) + (\tilde{\mathbf{r}}_s^T X \oplus \tilde{\mathbf{q}}_s^T \Delta) \ ,$$

where $\tilde{\mathbf{l}}_s^T = (l_{s,1}, \dots, l_{s,N})$, $\tilde{\mathbf{r}}_s^T = (r_{s,1}, \dots, r_{s,N})$ are vectors in $(\mathbb{Z}_2)^N$, and $\tilde{\mathbf{p}}_s^T = (p_{s,1}, \dots, p_{s,M})$, $\tilde{\mathbf{q}}_s^T = (q_{s,1}, \dots, q_{s,M})$ are vectors in $(\mathbb{Z}_2)^M$.

If we denote the carry vector of the s-th modular addition by $\tilde{\mathbf{c}}_s^T = (c_{s,p-1}, \dots, c_{s,1}, c_{s,0})$, then we can rewrite the s-th modular addition as

$$(\tilde{\mathbf{l}}_s^T X \oplus \tilde{\mathbf{p}}_s^T \Delta) + (\tilde{\mathbf{r}}_s^T X \oplus \tilde{\mathbf{q}}_s^T \Delta) = \tilde{\mathbf{l}}_s^T X \oplus \tilde{\mathbf{p}}_s^T \Delta \oplus \tilde{\mathbf{r}}_s^T X \oplus \tilde{\mathbf{q}}_s^T \Delta \oplus \tilde{\mathbf{c}}_s^T \ .$$

Some of the unknowns and parameters can appear in the equation as “singles”, i.e. not in a summand of any modular addition. For example, $\mathbf{x}$ and $\boldsymbol{\beta}$ are singles in (5). The presence of single unknowns in an equation can be described by a vector $\mathbf{u}^T = (u_1, \dots, u_N) \in (\mathbb{Z}_2)^N$, where $u_j = 1$ if and only if the unknown $\tilde{\mathbf{x}}_j$ appears as a single. Similarly, the presence of single parameters in an equation is described by a vector $\mathbf{v}^T = (v_1, \dots, v_M) \in (\mathbb{Z}_2)^M$, where $v_k = 1$ if and only if the parameter $\tilde{\boldsymbol{\delta}}_k$ appears in the equation also as a single.

So any AX-equation of depth 1 can be written as

$$\bigoplus_{s=1}^{S} \left[(\tilde{\mathbf{l}}_s^T X \oplus \tilde{\mathbf{p}}_s^T \Delta) + (\tilde{\mathbf{r}}_s^T X \oplus \tilde{\mathbf{q}}_s^T \Delta)\right] \oplus \mathbf{u}^T X \oplus \mathbf{v}^T \Delta = \mathbf{0} \ , \tag{11}$$

or with the carry vectors as

$$\bigoplus_{s=1}^{S} \left[\tilde{\mathbf{l}}_s^T X \oplus \tilde{\mathbf{p}}_s^T \Delta \oplus \tilde{\mathbf{r}}_s^T X \oplus \tilde{\mathbf{q}}_s^T \Delta \oplus \tilde{\mathbf{c}}_s^T\right] \oplus \mathbf{u}^T X \oplus \mathbf{v}^T \Delta = \mathbf{0} \ . \tag{12}$$

The equation can be written in a compact matrix form if we combine the vectors $\tilde{\mathbf{l}}_s$, $\tilde{\mathbf{r}}_s$ into matrices

$$L = \begin{pmatrix} \tilde{\mathbf{l}}_1^T \\ \tilde{\mathbf{l}}_2^T \\ \vdots \\ \tilde{\mathbf{l}}_S^T \end{pmatrix} = \begin{pmatrix} l_{1,1} & \cdots & l_{1,N} \\ l_{2,1} & \cdots & l_{2,N} \\ \vdots & \ddots & \vdots \\ l_{S,1} & \cdots & l_{S,N} \end{pmatrix}, \quad R = \begin{pmatrix} \tilde{\mathbf{r}}_1^T \\ \tilde{\mathbf{r}}_2^T \\ \vdots \\ \tilde{\mathbf{r}}_S^T \end{pmatrix} = \begin{pmatrix} r_{1,1} & \cdots & r_{1,N} \\ r_{2,1} & \cdots & r_{2,N} \\ \vdots & \ddots & \vdots \\ r_{S,1} & \cdots & r_{S,N} \end{pmatrix}$$

of type $S \times N$, the vectors $\tilde{\mathbf{p}}_s$ and $\tilde{\mathbf{q}}_s$ into matrices

$$P = \begin{pmatrix} \tilde{\mathbf{p}}_1^T \\ \tilde{\mathbf{p}}_2^T \\ \vdots \\ \tilde{\mathbf{p}}_S^T \end{pmatrix} = \begin{pmatrix} p_{1,1} & \cdots & p_{1,M} \\ p_{2,1} & \cdots & p_{2,M} \\ \vdots & \ddots & \vdots \\ p_{S,1} & \cdots & p_{S,M} \end{pmatrix}, \quad Q = \begin{pmatrix} \tilde{\mathbf{q}}_1^T \\ \tilde{\mathbf{q}}_2^T \\ \vdots \\ \tilde{\mathbf{q}}_M^T \end{pmatrix} = \begin{pmatrix} q_{1,1} & \cdots & q_{1,M} \\ q_{2,1} & \cdots & q_{2,M} \\ \vdots & \ddots & \vdots \\ q_{S,1} & \cdots & q_{S,M} \end{pmatrix}$$

of type $S \times M$, and the carry vectors $\tilde{\mathbf{c}}_1, \ldots \tilde{\mathbf{c}}_S$ into a carry matrix

$$C = \begin{pmatrix} \tilde{\mathbf{c}}_1^T \\ \tilde{\mathbf{c}}_2^T \\ \vdots \\ \tilde{\mathbf{c}}_S^T \end{pmatrix} = \begin{pmatrix} c_{1,p-1} & c_{1,p-2} & \cdots & q_{1,0} \\ c_{2,p-1} & c_{2,p-2} & \cdots & q_{2,0} \\ \vdots & \vdots & \ddots & \vdots \\ c_{S,p-1} & c_{S,p-2} & \cdots & q_{S,0} \end{pmatrix}$$

of type $S \times p$.

The big xor in the equation (12) is the linear combination of the rows of the matrix $LX \oplus P\Delta \oplus RX \oplus Q\Delta \oplus C$ with all coefficients equal to 1. If we denote by $\mathbf{1}_S^T$ the vector of ones $(1, 1, \ldots, 1) \in \mathbb{Z}_2^S$, then the big xor equals

$$\mathbf{1}_S^T \left(LX \oplus P\Delta \oplus RX \oplus Q\Delta \oplus C \right) \; .$$

So the equation (11) can be written as

$$\mathbf{1}_S^T \left(LX \oplus P\Delta \oplus RX \oplus Q\Delta \oplus C \right) \oplus \mathbf{u}^T X \oplus \mathbf{v}^T \Delta = \mathbf{0}. \tag{13}$$

The binary matrices L, R, P, Q and the vectors $\mathbf{u}^T$, $\mathbf{v}^T$ describe the form of the equation.

Example 1.1. In the first example the number of unknowns is $N = 2$, the number of parameters is $M = 3$, and the number of modular additions is $S = 2$. The matrices L, R, P, Q are

$$L = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}, \; R = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}, \; P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \; Q = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \; ,$$

and the vectors $\mathbf{u}, \mathbf{v}$ are

$$\mathbf{u}^T = (0, 0), \quad \mathbf{v}^T = (0, 0, 1) \; .$$

Example 1.2. In the second example we have $N = 1$, $M = 2$, $S = 1$, and the matrices L, R, P, Q are

$$L = \begin{pmatrix} 1 \end{pmatrix}, \; R = \begin{pmatrix} 0 \end{pmatrix}, \; P = \begin{pmatrix} 0 & 0 \end{pmatrix}, \; Q = \begin{pmatrix} 1 & 0 \end{pmatrix} \; .$$

The vectors $\mathbf{u}, \mathbf{v}$ are

$$\mathbf{u}^T = (1), \quad \mathbf{v}^T = (0,1) \; .$$

Example 1.3. In the third example we have $N = 2$, $M = 3$, $S = 3$, and the matrices L, R, P, Q are

$$L = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}, \; R = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}, \; P = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \; Q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \; .$$

The vectors $\mathbf{u}, \mathbf{v}$ are

$$\mathbf{u}^T = (0,0), \quad \mathbf{v}^T = (0,0,0) \; .$$

Example 1.4. In this case we have $N = 1$, $M = 2$, $S = 2$, and the matrices L, R, P, Q are

$$L = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \; R = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \; P = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \; Q = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \; .$$

The vectors $\mathbf{u}, \mathbf{v}$ are

$$\mathbf{u}^T = (0), \quad \mathbf{v}^T = (1,0) \; .$$

Example 1.5. In the fifth example we have $N = 2$, $M = 3$, $S = 3$, and the matrices L, R, P, Q are

$$L = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}, \; R = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}, \; P = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \; Q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \; .$$

The vectors $\mathbf{u}, \mathbf{v}$ are

$$\mathbf{u}^T = (0,0), \quad \mathbf{v}^T = (0,0,1) \; .$$

1.5 Trivial and non-trivial cases

We will study AX-equations of depth 1 written in the form (13). It can be written even more conveniently as

$$\left(\mathbf{1}_S^T (L \oplus R) \oplus \mathbf{u}^T\right) X \oplus \mathbf{1}_S^T C \oplus \left(\mathbf{1}_S^T (P \oplus Q) \oplus \mathbf{v}^T\right) \Delta = \mathbf{0} \; . \tag{14}$$

We write the unknown matrix X in the column form $X = (\mathbf{x}_{p-1}|\mathbf{x}_{p-2}|\dots|\mathbf{x}_0)$, where for $i = 1, \dots p-1$, $\mathbf{x}_i \in (\mathbb{Z}_2)^N$ is the vector of the i-th bits of the unknowns $\tilde{\mathbf{x}}_1^T, \dots, \tilde{\mathbf{x}}_N^T$. Similarly, $\Delta = (\boldsymbol{\delta}_{p-1}|\boldsymbol{\delta}_{p-2}|\dots|\boldsymbol{\delta}_0)$ is the column form of the parameter matrix and $C = (\mathbf{c}_{p-1}|\mathbf{c}_{p-2}|\dots|\mathbf{c}_0)$ is the column form of the carry matrix. So $\mathbf{c}_i$ is the vector of the i-th carries of the carry vectors $\tilde{\mathbf{c}}_1^T, \dots, \tilde{\mathbf{c}}_S^T$.

For a randomly selected matrix X of type $N \times p$ the equality (14) holds if and only if

$$\left(\mathbf{1}_S^T (L \oplus R) \oplus \mathbf{u}^T\right) \mathbf{x}_j \oplus \mathbf{1}_S^T \mathbf{c}_j \oplus \left(\mathbf{1}_S^T (P \oplus Q) \oplus \mathbf{v}^T\right) \boldsymbol{\delta}_j = 0 \tag{15}$$

for every $j = 0, \dots, p-1$.

If $i \in \{0, \dots, p\}$, then we say that a matrix X *solves the equation* (14) *mod* 2^i if

$$\left(\mathbf{1}_S^T (L \oplus R) \oplus \mathbf{u}^T\right) \mathbf{x}_j \oplus \mathbf{1}_S^T \mathbf{c}_j \oplus \left(\mathbf{1}_S^T (P \oplus Q) \oplus \mathbf{v}^T\right) \boldsymbol{\delta}_j = 0$$

for every $j = 0, \dots, i-1$. Thus every matrix X solves (14) mod 2^0. A matrix X solves (14) if and only if it solves it mod 2^p.

We will proceed by the following induction. We start with a matrix X solving the equation (14) mod 2^i for some $i = 0, \dots, p-1$, and find conditions under which it solves (14) also mod 2^{i+1} .

Note also that the fact that X solves (14) mod 2^i depends only on the columns $\mathbf{x}_{i-1}, \dots, \mathbf{x}_0$ and $\mathbf{c}_{i-1}, \dots, \mathbf{c}_0$. And since any column $\mathbf{c}_j$ depends only on $\mathbf{x}_{j-1}, \dots, \mathbf{x}_0$, the induction step reduces to the equation

$$\left(\mathbf{1}_S^T (L \oplus R) \oplus \mathbf{u}^T\right) \mathbf{x}_i \oplus \mathbf{1}_S^T \mathbf{c}_i \oplus \left(\mathbf{1}_S^T (P \oplus Q) \oplus \mathbf{v}^T\right) \boldsymbol{\delta}_i = 0 \ , \tag{16}$$

in the unknown vector $\mathbf{x}_i$. The induction step is easy when the vector

$$\left(\mathbf{1}_S^T (L \oplus R) \oplus \mathbf{u}^T\right) \neq \mathbf{0} \ .$$

In this case a matrix X solving (14) mod 2^i is a solution mod 2^{i+1} if and only if

$$\left(\mathbf{1}_S^T (L \oplus R) \oplus \mathbf{u}^T\right) \mathbf{x}_i = \mathbf{1}_S^T \mathbf{c}_i \oplus \left(\mathbf{1}_S^T (P \oplus Q) \oplus \mathbf{v}^T\right) \boldsymbol{\delta}_i \ . \tag{17}$$

This is a single linear equation in $\mathbf{x}_i$ with a known right-hand side not depending on $mathbfx_i$. Since the coefficient vector $\mathbf{1}_S^T (L \oplus R) \oplus \mathbf{u}^T \neq \mathbf{0}$, vectors $\mathbf{x}_i$ satisfying (17) form an afine hyperplane in $(\mathbb{Z}_2)^N$. The number of these $\mathbf{x}_i$'s is thus 2^{N-1} and the probability that a matrix X solving (14) mod 2^i is also a solution mod 2^{i+1} is exactly $1/2$.

We call the case $\mathbf{1}_S^T (L \oplus R) \oplus \mathbf{u}^T \neq \mathbf{0}$ *trivial.* Results about the trivial case of equations are summed up in the following proposition.

Proposition 1.6. *If $\mathbf{1}_S^T (L \oplus R) \oplus \mathbf{u}^T \neq \mathbf{0}$ then a randomly selected binary matrix X of type $N \times p$ satisfies the equation (14) with probability 2^{-p}.*

In particular, any equation (14) with $\mathbf{1}_S^T (L \oplus R) \oplus \mathbf{u}^T \neq \mathbf{0}$ has a solution.

1.6 The non-trivial case

The equation (14) is called *non-trivial* if $\mathbf{1}_S^T (L \oplus R) \oplus \mathbf{u}^T = \mathbf{0}$. From now on we will consider only non-trivial equations. One can easily check that our five examples of AX-equations are all non-trivial. In the non-trivial case the equation (14) reduces to

$$\mathbf{1}_S^T C = \left(\mathbf{1}_S^T (P \oplus Q) \oplus \mathbf{v}^T\right) \Delta. \tag{18}$$

To simplify further notations we set

$$\boldsymbol{\omega}^T = \mathbf{1}_S^T (P \oplus Q) \oplus \mathbf{v}^T \ .$$

Hence a matrix X is a solution of (18) mod 2^i if and only if

$$\mathbf{1}_S^T \mathbf{c}_j = \boldsymbol{\omega}^T \boldsymbol{\delta}_j \tag{19}$$

for every $j = 0, \dots, i-1$. This condition does not depend on the column $\mathbf{x}_{i-1}$ of the matrix X.

A solution X of (18) mod 2^i is also a solution 2^{i+1} if and only if

$$\mathbf{1}_S^T \mathbf{c}_i = \boldsymbol{\omega}^T \boldsymbol{\delta}_i \ . \tag{20}$$

Note that the last equation depends only on the columns $\mathbf{x}_{i-1}, \dots, \mathbf{x}_0$ of the matrix, and the induction hypothesis does not depend on $\mathbf{x}_{i-1}$.

To investigate further the equation (20) we expand the components of $\mathbf{c}_i$ using formula (1) defining the i-th carry. For every $s = 1, \dots, S$ the carry $c_{s,i}$ is the i-th carry in the modular sum

$$(\tilde{\mathbf{l}}_s^T X \oplus \tilde{\mathbf{p}}_s^T \Delta) + (\tilde{\mathbf{r}}_s^T X \oplus \tilde{\mathbf{q}}_s^T \Delta) \ ,$$

hence

$$\begin{aligned}
\mathbf{c}_{s,i} &= (\tilde{\mathbf{l}}_s^T \mathbf{x}_{i-1} \oplus \tilde{\mathbf{p}}_s^T \boldsymbol{\delta}_{i-1})(\tilde{\mathbf{r}}_s^T \mathbf{x}_{i-1} \oplus \tilde{\mathbf{q}}_s^T \boldsymbol{\delta}_{i-1}) \oplus \\
&\quad c_{s,i-1}(\tilde{\mathbf{l}}_s^T \mathbf{x}_{i-1} \oplus \tilde{\mathbf{r}}_s^T \mathbf{x}_{i-1} \oplus \tilde{\mathbf{p}}_s^T \boldsymbol{\delta}_{i-1} \oplus \tilde{\mathbf{q}}_s^T \boldsymbol{\delta}_{i-1}) \\
&= (\tilde{\mathbf{l}}_s^T \mathbf{x}_{i-1})(\tilde{\mathbf{r}}_s^T \mathbf{x}_{i-1}) \oplus (\tilde{\mathbf{l}}_s^T \mathbf{x}_{i-1})(\tilde{\mathbf{q}}_s^T \boldsymbol{\delta}_{i-1}) \oplus (\tilde{\mathbf{p}}_s^T \boldsymbol{\delta}_{i-1})(\tilde{\mathbf{r}}_s^T \mathbf{x}_{i-1}) \oplus \\
&\quad c_{s,i-1}(\tilde{\mathbf{l}}_s^T \oplus \tilde{\mathbf{r}}_s^T)\mathbf{x}_{i-1} \oplus c_{s,i-1}(\tilde{\mathbf{p}}_s^T \oplus \tilde{\mathbf{q}}_s^T)\boldsymbol{\delta}_{i-1} \oplus (\tilde{\mathbf{p}}_s^T \boldsymbol{\delta}_{i-1})(\tilde{\mathbf{q}}_s^T \boldsymbol{\delta}_{i-1}) \\
&= \mathbf{x}_{i-1}^T(\tilde{\mathbf{l}}_s \tilde{\mathbf{r}}_s^T)\mathbf{x}_{i-1} \oplus \boldsymbol{\delta}_{i-1}^T \tilde{\mathbf{q}}_s \tilde{\mathbf{l}}_s^T \mathbf{x}_{i-1} \oplus \boldsymbol{\delta}_{i-1}^T \tilde{\mathbf{p}}_s \tilde{\mathbf{r}}_s^T \mathbf{x}_{i-1} \oplus \\
&\quad c_{s,i-1}(\tilde{\mathbf{l}}_s^T \oplus \tilde{\mathbf{r}}_s^T)\mathbf{x}_{i-1} \oplus c_{s,i-1}(\tilde{\mathbf{p}}_s^T \oplus \tilde{\mathbf{q}}_s^T)\boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\delta}_{i-1}^T(\tilde{\mathbf{p}}_s \tilde{\mathbf{q}}_s^T)\boldsymbol{\delta}_{i-1} \ .
\end{aligned}$$

Since $\mathbf{c}_i^T \mathbf{1}_S = c_{1,i} \oplus c_{2,i} \oplus \cdots \oplus c_{S,i}$, by summing up the previous equation over S we obtain

$$\begin{aligned}
\mathbf{c}_i^T \mathbf{1}_S &= \bigoplus_{s=1}^{S} \left(\mathbf{x}_{i-1}^T(\tilde{\mathbf{l}}_s \tilde{\mathbf{r}}_s^T)\mathbf{x}_{i-1}\right) \oplus \bigoplus_{s=1}^{S} \left(\boldsymbol{\delta}_{i-1}^T \tilde{\mathbf{q}}_s \tilde{\mathbf{l}}_s^T \mathbf{x}_{i-1}\right) \oplus \\
&\quad \bigoplus_{s=1}^{S} \left(\boldsymbol{\delta}_{i-1}^T \tilde{\mathbf{p}}_s \tilde{\mathbf{r}}_s^T \mathbf{x}_{i-1}\right) \oplus \bigoplus_{s=1}^{S} \left(c_{s,i-1}(\tilde{\mathbf{l}}_s^T \oplus \tilde{\mathbf{r}}_s^T)\mathbf{x}_{i-1}\right) \oplus \\
&\quad \bigoplus_{s=1}^{S} \left(c_{s,i-1}(\tilde{\mathbf{p}}_s^T \oplus \tilde{\mathbf{q}}_s^T)\boldsymbol{\delta}_{i-1}\right) \oplus \bigoplus_{s=1}^{S} \left(\boldsymbol{\delta}_{i-1}^T(\tilde{\mathbf{p}}_s \tilde{\mathbf{q}}_s^T)\boldsymbol{\delta}_{i-1}\right) .
\end{aligned}$$

Next we calculate each sum on the right-hand side. We obtain

$$\bigoplus_{s=1}^{S} \mathbf{x}_{i-1}^T(\tilde{\mathbf{l}}_s\tilde{\mathbf{r}}_s^T)\mathbf{x}_{i-1} = \mathbf{x}_{i-1}^T\left(\bigoplus_{s=1}^{S}\tilde{\mathbf{l}}_s\tilde{\mathbf{r}}_s^T\right)\mathbf{x}_{i-1} = \mathbf{x}_{i-1}^T L^T R\mathbf{x}_{i-1} \ ,$$

the last equality follows from the dyadic expansion of the product $L^T R$.

Similarly we obtain

$$\bigoplus_{s=1}^{S} \boldsymbol{\delta}_{i-1}^T\tilde{\mathbf{q}}_s\tilde{\mathbf{l}}_s^T\mathbf{x}_{i-1} = \boldsymbol{\delta}_{i-1}^T(Q^T L)\mathbf{x}_{i-1} \ ,$$

$$\bigoplus_{s=1}^{S} \boldsymbol{\delta}_{i-1}^T\tilde{\mathbf{p}}_s\tilde{\mathbf{r}}_s^T\mathbf{x}_{i-1} = \boldsymbol{\delta}_{i-1}^T(P^T R)\mathbf{x}_{i-1} \ ,$$

$$\bigoplus_{s=1}^{S} c_{s,i-1}(\tilde{\mathbf{l}}_s^T \oplus \tilde{\mathbf{r}}_s^T)\mathbf{x}_{i-1} = \mathbf{c}_{i-1}^T(L \oplus R)\mathbf{x}_{i-1} \ ,$$

$$\bigoplus_{s=1}^{S} c_{s,i-1}(\tilde{\mathbf{p}}_s^T \oplus \tilde{\mathbf{q}}_s^T)\boldsymbol{\delta}_{i-1} = \mathbf{c}_{i-1}^T(P \oplus Q)\boldsymbol{\delta}_{i-1} \ ,$$

$$\bigoplus_{s=1}^{S} \boldsymbol{\delta}_{i-1}^T(\tilde{\mathbf{p}}_s\tilde{\mathbf{q}}_s^T)\boldsymbol{\delta}_{i-1} = \boldsymbol{\delta}_{i-1}^T(P^T Q)\boldsymbol{\delta}_{i-1} \ .$$

Hence

$$\begin{aligned}\mathbf{c}_i^T\mathbf{1}_S \ &= \ \mathbf{x}_{i-1}^T L^T R\mathbf{x}_{i-1} \oplus \boldsymbol{\delta}_{i-1}^T(Q^T L \oplus P^T R)\mathbf{x}_{i-1} \oplus \mathbf{c}_{i-1}^T(L \oplus R)\mathbf{x}_{i-1} \oplus \\ &\quad \mathbf{c}_{i-1}^T(P \oplus Q)\boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\delta}_{i-1}^T(P^T Q)\boldsymbol{\delta}_{i-1}.\end{aligned}$$

So $\mathbf{c}_i^T\mathbf{1}_S$ is the sum of a quadratic form $\mathbf{x}_{i-1}^T L^T R\mathbf{x}_{i-1}$ in $\mathbf{x}_{i-1}$ with constant coefficients, a linear form in $\mathbf{x}_{i-1}$ with coefficients depending on $\mathbf{c}_{i-1}$, and a constant term depending on $\mathbf{c}_{i-1}$.

Proposition 1.7. *If the equation (14) is non-trivial, then a solution X mod 2^i is a solution mod 2^{i+1} if and only if*

$$\begin{aligned}&\mathbf{x}_{i-1}^T L^T R\mathbf{x}_{i-1} \oplus \left[\boldsymbol{\delta}_{i-1}^T\left(Q^T L \oplus P^T R\right) \oplus \mathbf{c}_{i-1}^T(L \oplus R)\right]\mathbf{x}_{i-1} \\ &= \mathbf{c}_{i-1}^T(P \oplus Q)\boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\delta}_{i-1}^T(P^T Q)\boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T\boldsymbol{\delta}_i \ .\end{aligned} \tag{21}$$

Proof. In the non-trivial case a solution X mod 2^i of (14) is a solution mod 2^{i+1} if and only if the equality (20) holds. So the proof follows from the calculation preceding the proposition. □

1.7 Quadratic and linear cases

The quadratic form $\mathbf{x}_{i-1}^T L^T R\mathbf{x}_{i-1}$ is over $\mathbb{Z}_2$. It reduces to a sum of squares

$$\varepsilon_1 x_{1,i-1}^2 \oplus \varepsilon_1 x_{2,i-1}^2 \oplus \cdots \oplus \varepsilon_N x_{N,i-1}^2$$

if and only if the coefficient matrix L^TR is symmetric. Here $\boldsymbol{\varepsilon} = (\varepsilon_1, \dots, \varepsilon_N) = \operatorname{diag}(L^TR)$ is the vector of diagonal elements of L^TR.

We call a non-trivial equation (14) *quadratic*, if the matrix L^TR is not symmetric, and *linear* otherwise (i.e. if L^TR is symmetric).

In the linear case the equation (14) transforms via (21) to a linear equation in $\mathbf{x}_{i-1}$ as follows. Since $x^2 = x$ for every $x \in \mathbb{Z}_2$, the value of the quadratic form

$$\mathbf{x}_{i-1}^T L^T R \mathbf{x}_{i-1} = \varepsilon_1 x_{1,i-1}^2 \oplus \varepsilon_1 x_{2,i-1}^2 \oplus \cdots \oplus \varepsilon_N x_{N,i-1}^2$$

is the same as the value of the linear form

$$\boldsymbol{\varepsilon}^T \mathbf{x}_{i-1} = \varepsilon_1 x_{1,i-1}^2 \oplus \varepsilon_1 x_{2,i-1}^2 \oplus \cdots \oplus \varepsilon_N x_{N,i-1}^2 \ .$$

for any vector $\mathbf{x}_{i-1} \in (\mathbb{Z}_2)^N$. Hence a vector $\mathbf{x}_{i-1}$ satisfies (21) if and only if it satisfies the linear equation

$$\begin{aligned}&\boldsymbol{\varepsilon}^T \mathbf{x}_{i-1} \oplus \left[\boldsymbol{\delta}_{i-1}^T \left(Q^T L \oplus P^T R\right) \oplus \mathbf{c}_{i-1}^T (L \oplus R)\right] \mathbf{x}_{i-1} = \\ &\mathbf{c}_{i-1}^T (P \oplus Q) \boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\delta}_{i-1}^T (P^T Q) \boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T \boldsymbol{\delta}_i \ ,\end{aligned}$$

i.e.

$$\begin{aligned}&\left[\mathbf{c}_{i-1}^T (L \oplus R) \oplus \boldsymbol{\delta}_{i-1}^T \left(Q^T L \oplus P^T R\right) \oplus \boldsymbol{\varepsilon}^T\right] \mathbf{x}_{i-1} = \\ &\mathbf{c}_{i-1}^T (P \oplus Q) \boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\delta}_{i-1}^T (P^T Q) \boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T \boldsymbol{\delta}_i \ ,\end{aligned} \tag{22}$$

Similarly we can simplify the quadratic case. The value of the linear form

$$\left[\mathbf{c}_{i-1}^T (L \oplus R) \oplus \boldsymbol{\delta}_{i-1}^T \left(Q^T L \oplus P^T R\right)\right] \mathbf{x}_{i-1}$$

is for any $\mathbf{x}_{i-1} \in (\mathbb{Z}_2)^N$ the same as the value of the quadratic form

$$\mathbf{x}_{i-1}^T \operatorname{Diag}\left[\boldsymbol{\delta}_{i-1}^T \left(Q^T L \oplus P^T R\right) \oplus \mathbf{c}_{i-1}^T (L \oplus R)\right] \mathbf{x}_{i-1} \ .$$

Here $\operatorname{Diag}\left[\boldsymbol{\delta}_{i-1}^T \left(Q^T L \oplus P^T R\right) \oplus \mathbf{c}_{i-1}^T (L \oplus R)\right]$ denotes the diagonal matrix B over $\mathbb{Z}_2$ such that

$$\operatorname{diag}(B) = \boldsymbol{\delta}_{i-1}^T \left(Q^T L \oplus P^T R\right) \oplus \mathbf{c}_{i-1}^T (L \oplus R) \ .$$

So in the quadratic case the equation (21) reduces to

$$\begin{aligned}&\mathbf{x}_{i-1}^T L^T R \mathbf{x}_{i-1} \oplus \mathbf{x}_{i-1}^T B \mathbf{x}_{i-1} = \\ &\mathbf{c}_{i-1}^T (P \oplus Q) \boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\delta}_{i-1}^T (P^T Q) \boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T \boldsymbol{\delta}_i \ ,\end{aligned}$$

i.e.

$$\mathbf{x}_{i-1}^T (L^T R \oplus B) \mathbf{x}_{i-1} = \mathbf{c}_{i-1}^T (P \oplus Q) \boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\delta}_{i-1}^T (P^T Q) \boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T \boldsymbol{\delta}_i \tag{23}$$

with a non-symmetric matrix $L^TR \oplus B$. Note that $L^TR \oplus B$ is non-symmetric if and only if L^TR is non-symmetric, since B is diagonal.

First we deal with the quadratic case. Here we can use the following well-known result about quadratic forms over $\mathbb{Z}_2$, see e.g. Section 1-12 of the monograph by Kaplansky [9].

Theorem 1.8. *If $\mathbf{x}^T A\mathbf{x}$ a quadratic form of order n over $\mathbb{Z}_2$ defined by a non-symmetric matrix A, then there exists a coordinate change $\mathbf{x} = C\mathbf{y}$ with $\mathbf{y}^T = (y_1, \dots, y_n)$, such that the quadratic form $\mathbf{y}^T(C^T AC)\mathbf{y}$ is either*

$$y_1y_2 \oplus \cdots \oplus y_{2k-1}y_{2k} \oplus y_{2k+1}^2 \oplus \cdots \oplus y_{2k+l}^2$$

or

$$y_1y_2 \oplus \cdots \oplus y_{2k-3}y_{2k-2} \oplus y_{2k-1}y_{2k} \oplus y_{2k-1}^2 \oplus y_{2k}^2 \oplus y_{2k+1}^2 \oplus \cdots \oplus y_{2k+l}^2$$

for some $k \geq 1$ and $l \geq 0$.

In particular, the form $\mathbf{x}^T A\mathbf{x}$ always takes both values 0 and 1.

Since the right-hand side of (23) does not depend on $\mathbf{x}_{i-1}$, we immediately obtain the following result on quadratic AX-equations.

Proposition 1.9. *For a quadratic AX-equation (14) and any $i = 0, \dots, p-1$ there exists a solution X mod 2^i that is also a solution mod 2^{i+1}.*

In particular, every quadratic AX-equation has a solution.

Proof. To prove the first claim, let us suppose that for some $i < p-1$ there exists a solution X mod 2^i. Then $X = (\mathbf{x}_{p-1}|\mathbf{x}_{p-2}|\cdots|\mathbf{x}_0)$ is a solution mod 2^{i+1} if and only if $\mathbf{x}_{i-1}$ satisfies (23), where the matrix $L^T R \oplus B$ is non-symmetric. By Theorem 1.8 there indeed does exist some $\mathbf{x}_{i-1} \in (\mathbb{Z}_2)^N$ satisfying the equation (23). Hence there exists a solution of (14) mod 2^{i+1}.

The first claim is the induction step to prove the second claim, it is just sufficient to recall that any X is a solution of (14) mod 2^0. □

The exact evaluation of the probability that a randomly selected X satisfies a quadratic AX-equation is out of scope of the current text. It would require a detailed study of the size of various quadrics over $\mathbb{Z}_2$.

From our list of equations only the last one is quadratic.

Example 1.5. It is sufficient to calculate the matrix $L^T R$:

$$L^T R = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}^T \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} .$$

The matrix is not symmetric, hence the equation (10) is quadratic and by Proposition 1.9 is solvable for any parameters $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$.

1.8 The linear case

The remaining four equations from our list are linear. We can easily check it by calculating the matrix $L^T R$ in each case. Investigation of linear AX-equations is the core of this chapter. In the linear case, the question if a solution X mod 2^i is also a solution mod 2^i reduces to (22), i.e. to

$$\begin{aligned}&\left[\mathbf{c}_{i-1}^T(L \oplus R) \oplus \boldsymbol{\delta}_{i-1}^T\left(Q^T L \oplus P^T R\right) \oplus \boldsymbol{\varepsilon}^T\right] \mathbf{x}_{i-1} = \\ &\mathbf{c}_{i-1}^T(P \oplus Q)\boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\delta}_{i-1}^T(P^T Q)\boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T\boldsymbol{\delta}_i \ .\end{aligned}$$

This is a single linear equation for $\mathbf{x}_{i-1}$ with coefficients depending only on $\mathbf{c}_{i-1}$. To simplify slightly our notation we set

$$A = Q^T L \oplus P^T R \ ,$$

Thus A is a matrix of type $M \times N$. So we investigate the equation

$$\begin{aligned}&\left[\mathbf{c}_{i-1}^T(L \oplus R) \oplus \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T\right] \mathbf{x}_{i-1} = \\ &\mathbf{c}_{i-1}^T(P \oplus Q)\boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\delta}_{i-1}^T(P^T Q)\boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T\boldsymbol{\delta}_i \ .\end{aligned} \tag{24}$$

In the linear case we formulate a series of checks determining properties of linear AX-equations.

Check No. 1

The vector

$$\mathbf{c}_{i-1}^T(L \oplus R) \oplus \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T \tag{25}$$

of coefficients in (24) depends on the carries $\mathbf{c}_{i-1}$ and also on the parameters $\boldsymbol{\delta}_{i-1}$. If this vector is non-zero for any solution X mod 2^i, then the equation (24) has always a solution. But the assumption that X is a solution mod 2^i means that the vector $\mathbf{c}_{i-1}^T$ must satisfy the equation

$$\mathbf{c}_{i-1}^T \mathbf{1}_S = \boldsymbol{\omega}^T\boldsymbol{\delta}_{i-1} \ . \tag{26}$$

So the vector (25) is certainly non-zero for every solution X mod 2^i if the system of linear equations

$$\mathbf{y}_{i-1}^T(L \oplus R \mid \mathbf{1}_S) = (\boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T \mid \boldsymbol{\omega}^T\boldsymbol{\delta}_{i-1})$$

is unsolvable.

Proposition 1.10. *We assume that the equation (14) is linear. If*

$$\operatorname{rank}\left(\ L \oplus R \ \middle|\ \mathbf{1}_S\ \right) < \mathrm{c}_{\mathrm{i-1}}^{\mathrm{T}}\mathbf{1}_{\mathrm{S}} \left(\begin{array}{c|c} L \oplus R & \mathbf{1}_S \\ \boldsymbol{\delta}_{i-1}^T A \oplus \varepsilon^T & \boldsymbol{\omega}^T\boldsymbol{\delta}_{i-1}\end{array}\right) \ ,$$

then the equation (24) is always solvable. The set of solutions of (24) forms a hyperplane in $(\mathbb{Z}_2)^N$, hence it has cardinality 2^{N-1}. A solution of X (14) mod 2^i is a solution mod 2^{i+1} with probability 2^{-1}.

In particular, if there exists a solution of (14) mod 2^i, then there is also a solution mod 2^{i+1}.

Proof. Let X be of a solution of (14) mod 2^i. Then the vector $\mathbf{c}_{i-1}^T$ satisfies the equation (26). The matrix X is a solution mod 2^{i+1} if and only if it satisfies (16). In the non-trivial case, (16) reduces to (21) by Proposition 1.7. And since we assume the linear case, (21) reduces further to (24). By our assumption, the system of linear equations

$$\mathbf{y}_{i-1}^T(L \oplus R \mid \mathbf{1}_S) = (\boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T \mid \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1})$$

has no solution. Since X is a solution of (14) mod 2^i, the vector $\mathbf{c}_{i-1}^T$ always satisfies the last equation $\mathbf{c}_{i-1}^T \mathbf{1}_S = \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1}$. Hence it must be

$$\mathbf{c}_{i-1}^T(L \oplus R) \oplus \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T \neq \mathbf{0} \ .$$

It follows that the equation (24) is solvable and the set of it's solutions is a hyperplane in $(\mathbb{Z}_2)^N$. Thus any solution X mod 2^i is a solution mod 2^{i+1} with probability 2^{-1}.

The last claim follows directly from the previous one. □

Since the right-hand side of the system of linear equations

$$\mathbf{y}_{i-1}^T(L \oplus R \mid \mathbf{1}_S) = (\boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T \mid \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1})$$

depends on $\boldsymbol{\delta}_{i-1}^T$, a linear AX-equation may satisfy the assumptions of Proposition 1.10 for some i's and not satisfy them for other i's.

Example 1.1. As for the linear equation (3), we have

$$L = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}, \quad R = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}, \quad P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad Q = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \ ,$$

and the vectors $\mathbf{u}^T = (0,0)$, $\mathbf{v}^T = (0,0,1)$. Thus we get

$$L \oplus R = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \quad P \oplus Q = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad L^T R = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix},$$

hence also $\boldsymbol{\varepsilon}^T = (0,0)$ and $\boldsymbol{\omega}^T = \mathbf{1}_S^T\,(P \oplus Q) \oplus \mathbf{v}^T = (1,1,1)$. Moreover

$$A = Q^T L \oplus P^T R = \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{pmatrix},$$

We set further

$$\Delta = \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \\ \boldsymbol{\gamma} \end{pmatrix} \ .$$

So we get

$$\boldsymbol{\delta}_{i-1}^T A = (\alpha_{i-1}, \beta_{i-1}, \gamma_{i-1}) \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{pmatrix} = (\beta_{i-1}, \alpha_{i-1}) \ ,$$

and also

$$\boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1} = (1,1,1)\begin{pmatrix}\alpha_{i-1}\\ \beta_{i-1}\\ \gamma_{i-1}\end{pmatrix} = \alpha_{i-1} \oplus \beta_{i-1} \oplus \gamma_{i-1} \ .$$

Hence the rank of

$$\left(\begin{array}{c|c} L \oplus R & \mathbf{1}_S \\ \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T & \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1}\end{array}\right) = \left(\begin{array}{cc|c} 1 & 1 & 1 \\ 1 & 1 & 1 \\ \beta_{i-1} & \alpha_{i-1} & \alpha_{i-1} \oplus \beta_{i-1} \oplus \gamma_{i-1}\end{array}\right)$$

is the same as the rank of

$$(L \oplus R \mid \mathbf{1}_S) = \left(\begin{array}{cc|c} 1 & 1 & 1 \\ 1 & 1 & 1 \end{array}\right)$$

if and only if $\alpha_{i-1} = \beta_{i-1} = \gamma_{i-1}$.

It proves that equation (3) satisfies the assumption of Proposition 1.10 if and only if the three bits α_{i-1}, β_{i-1}, γ_{i-1} are not the same. Any such index i reduces the probability that a matrix X solves (3) by exactly 2^{-1}. As usual, we set $\alpha_{-1} = \beta_{-1} = \gamma_{-1} = 0$.

Example 1.2. As for the linear equation (5), we have

$$L = \begin{pmatrix}1\end{pmatrix},\ R = \begin{pmatrix}0\end{pmatrix},\ P = \begin{pmatrix}0 & 0\end{pmatrix},\ Q = \begin{pmatrix}1 & 0\end{pmatrix},\ \mathbf{u}^T = (1),\ \mathbf{v}^T = (0,1) \ .$$

Thus we get

$$L \oplus R = \begin{pmatrix}1\end{pmatrix}, \quad P \oplus Q = \begin{pmatrix}1 & 0\end{pmatrix}, \quad L^T R = \begin{pmatrix}0\end{pmatrix}, \quad \boldsymbol{\varepsilon}^T \begin{pmatrix}0\end{pmatrix} \ ,$$

and $\boldsymbol{\omega}^T = \mathbf{1}_S^T (P \oplus Q) \oplus \mathbf{v}^T = (1,1)$. Moreover,

$$A = Q^T L \oplus P^T R = \begin{pmatrix}1\\0\end{pmatrix} \ .$$

We set

$$\Delta = \begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} \ .$$

Then

$$\begin{aligned} \boldsymbol{\delta}_{i-1}^T A &= (\alpha_{i-1}, \beta_{i-1})\begin{pmatrix}1\\0\end{pmatrix} = (\alpha_{i-1}) \ , \\ \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1} &= (1,1)\begin{pmatrix}\alpha_{i-1}\\ \beta_{i-1}\end{pmatrix} = \alpha_{i-1} \oplus \beta_{i-1} \ . \end{aligned}$$

Hence the rank of

$$\left(\begin{array}{c|c} L \oplus R & \mathbf{1}_S \\ \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T & \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1}\end{array}\right) = \left(\begin{array}{c|c} 1 & 1 \\ \alpha_{i-1} & \alpha_{i-1} \oplus \beta_{i-1}\end{array}\right)$$

is equal to the rank of

$$(L \oplus R \mid \mathbf{1}_S) = \left(\begin{array}{c|c} 1 & 1 \end{array} \right)$$

if and only if $\beta_{i-1} = 0$. Thus the assumptions of Proposition 1.10 are satisfied whenever $\beta_{i-1} \neq 0$. Any such index i reduces the probability that a matrix X solves (5) by exactly 2^{-1}.

Example 1.3. In the linear equation (7) we have

$$L = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}, \ R = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}, \ P = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \ Q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \ ,$$

and the vectors $\mathbf{u}^T = (0,0)$, $\mathbf{v}^T = (0,0,0)$. Then we obtain

$$L \oplus R = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}, \quad P \oplus Q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad L^T R = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad \boldsymbol{\varepsilon}^T (0,0) \ .$$

The vector $\boldsymbol{\omega}^T = \mathbf{1}_S^T (P \oplus Q) \oplus \mathbf{v}^T = (1,1,1)$ and

$$A = Q^T L \oplus P^T R = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} \ .$$

We again set

$$\Delta = \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \\ \boldsymbol{\gamma} \end{pmatrix} \ .$$

Then

$$\begin{aligned} \boldsymbol{\delta}_{i-1}^T A &= (\alpha_{i-1}, \beta_{i-1}, \gamma_{i-1}) \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} = (\alpha_{i-1} \oplus \gamma_{i-1}, \beta_{i-1} \oplus \gamma_{i-1}) \ , \\ \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1} &= (1,1,1) \begin{pmatrix} \alpha_{i-1} \\ \beta_{i-1} \\ \gamma_{i-1} \end{pmatrix} = \alpha_{i-1} \oplus \beta_{i-1} \oplus \gamma_{i-1} \ . \end{aligned}$$

Since in this case

$$\operatorname{rank}(L \oplus R \mid \mathbf{1}_S) = \operatorname{rank} \left(\begin{array}{cc|c} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{array} \right) = 3 \ ,$$

We get

$$\operatorname{rank} \left(\begin{array}{c|c} L \oplus R & \mathbf{1}_S \\ \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T & \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1} \end{array} \right) = \operatorname{rank} (L \oplus R \mid \mathbf{1}_S) = 3 \ .$$

So the equation (7) never satisfies assumptions of Proposition 1.10.

Example 1.4. In the equation (9) we have

$$L = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad R = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad P = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad Q = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \ ,$$

and the vectors $\mathbf{u}^T = (0)$, $\mathbf{v}^T = (1, 0)$. We get

$$L \oplus R = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad P \oplus Q = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad L^T R = \begin{pmatrix} 0 \end{pmatrix}, \quad \boldsymbol{\varepsilon}^T \begin{pmatrix} 0 \end{pmatrix} \ .$$

The vector $\boldsymbol{\omega}^T = \mathbf{1}_S^T (P \oplus Q) \oplus \mathbf{v}^T = (0, 0)$ and

$$A = Q^T L \oplus P^T R = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \ .$$

We set

$$\Delta = \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} \ .$$

Then

$$\boldsymbol{\delta}_{i-1}^T A = (0), \quad \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1} = (0) \ .$$

In this case the rank of

$$\left(\begin{array}{c|c} L \oplus R & \mathbf{1}_S \\ \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T & \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1} \end{array} \right) = \left(\begin{array}{c|c} 1 & 1 \\ 1 & 1 \\ 0 & 0 \end{array} \right)$$

is always equal to the rank of

$$(L \oplus R \mid \mathbf{1}_S) = \left(\begin{array}{c|c} 1 & 1 \\ 1 & 1 \end{array} \right) \ .$$

It means that equation (9) never satisfies assumptions of Proposition 1.10.

If

$$\operatorname{rank} \left(\begin{array}{c|c} L \oplus R & \mathbf{1}_S \end{array} \right) = \operatorname{rank} \left(\begin{array}{c|c} L \oplus R & \mathbf{1}_S \\ \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T & \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1} \end{array} \right) \ ,$$

then we cannot exclude the possibility that for some solutions $X \bmod 2^i$ the vectors $\mathbf{c}_{i-1}^T$ satisfy the equation

$$\mathbf{c}_{i-1}^T (L \oplus R) = \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T \ , \tag{27}$$

i.e. that the left-hand side in (24) is equal to $\mathbf{0}$.

From now on we assume that

$$\operatorname{rank} \left(\begin{array}{c|c} L \oplus R & \mathbf{1}_S \end{array} \right) = \operatorname{rank} \left(\begin{array}{c|c} L \oplus R & \mathbf{1}_S \\ \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T & \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1} \end{array} \right) \ , \tag{28}$$

Check No. 2

In some cases it may even happen that every linear equation in the system (27) is a multiple of the induction hypothesis $\mathbf{c}_{i-1}^T \mathbf{1}_S = \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1}$, i.e. the left-hand side in (24) is equal to 0 for every X solving mod 2^i the equation (14). It obviously happens under the assumption of the following proposition.

Proposition 1.11. *We assume that the equation (14) is linear. If*

$$rank \left(\begin{array}{c|c} L \oplus R & \mathbf{1}_S \\ \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T & \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1} \end{array} \right) = 1 \ ,$$

then any X solving mod 2^i a linear equation (14) turns the left-hand side of (24) to $\mathbf{0}$. Hence X is a solution mod 2^{i+1} if and only if the vector $\mathbf{c}_{i-1}^T$ satisfies

$$\mathbf{c}_{i-1}^T (P \oplus Q) \boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\delta}_{i-1}^T (P^T Q) \boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T \boldsymbol{\delta}_i = 0 \ . \tag{29}$$

We check again which of the equations from our list satisfy the assumption of Proposition 1.11.

Example 1.1. We already know that (28) holds if and only if $\alpha_{i-1} = \beta_{i-1} = \gamma_{i-1}$. In this case the matrix

$$\left(\begin{array}{c|c} L \oplus R & \mathbf{1}_S \\ \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T & \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1} \end{array} \right) = \left(\begin{array}{cc|c} 1 & 1 & 1 \\ 1 & 1 & 1 \\ \beta_{i-1} & \alpha_{i-1} & \alpha_{i-1} \oplus \beta_{i-1} \oplus \gamma_{i-1} \end{array} \right)$$

indeed has rank equal to 1.

Example 1.2. The equation (5) satisfies (28) in and only if $\beta_{i-1} = 0$. Hence the matrix

$$\left(\begin{array}{c|c} L \oplus R & \mathbf{1}_S \\ \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T & \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1} \end{array} \right) = \left(\begin{array}{c|c} 1 & 1 \\ \alpha_{i-1} & \alpha_{i-1} \oplus \beta_{i-1} \end{array} \right)$$

has rank equal to 1 also in this case.

Example 1.3. Since we have already calculated that in case of the equation (7)

$$\operatorname{rank} (L \oplus R \mid \mathbf{1}_S) = \operatorname{rank} \left(\begin{array}{c|c} L \oplus R & \mathbf{1}_S \\ \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T & \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1} \end{array} \right) = 3 \ ,$$

the equation (7) never satisfies the assumption of Proposition 1.11.

Example 1.4. Here we already know that

$$\operatorname{rank} \left(\ L \oplus R \mid \mathbf{1}_S \ \right) = \operatorname{rank} \left(\begin{array}{c|c} L \oplus R & \mathbf{1}_S \\ \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T & \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1} \end{array} \right) = 1 \ ,$$

so the equation (9) satisfies the assumption of Proposition 1.11 for every $i = 0, \ldots, p-1$.

In case

$$\operatorname{rank}\left(\begin{array}{c|c} L\oplus R & \mathbf{1}_S \end{array}\right) = \operatorname{rank}\left(\begin{array}{c|c} L\oplus R & \mathbf{1}_S \\ \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T & \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1} \end{array}\right) \geq 2,$$

there may be solutions $X \bmod 2^i$ such that the left-hand side of (24) is $\mathbf{0}$ for some other solutions mod 2^i it is different from $\mathbf{0}$. For the time being we leave this case aside and continue with the case

$$\operatorname{rank}\left(\begin{array}{c|c} L\oplus R & \mathbf{1}_S \end{array}\right) = \operatorname{rank}\left(\begin{array}{c|c} L\oplus R & \mathbf{1}_S \\ \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T & \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1} \end{array}\right) = 1 \ .$$

Check No. 3

By Proposition 1.11, a solution $X \bmod 2^i$ is a solution mod 2^{i+1} of a linear equation (14) if and only if the vector $\mathbf{c}_{i-1}^T$ satisfies the equation (29). Since it must satisfy the induction hypothesis (26), we have to consider the system of two linear equations

$$\begin{aligned} \mathbf{c}_{i-1}^T \mathbf{1}_S &= \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1} \ , \\ \mathbf{c}_{i-1}^T (P\oplus Q)\boldsymbol{\delta}_{i-1} &= \boldsymbol{\delta}_{i-1}^T (P^T Q)\boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T \boldsymbol{\delta}_i \ . \end{aligned} \tag{30}$$

The following proposition follows easily from the definition of parallel affine subspaces of a linear space. The definition of parallel subspaces includes also the possibility that one of the affine subspaces is contained in the other.

Proposition 1.12. *We assume that the equation (14) is linear and*

$$\operatorname{rank}\left(\begin{array}{c|c} L\oplus R & \mathbf{1}_S \\ \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T & \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1} \end{array}\right) = 1 \ .$$

If

$$\operatorname{rank}\left(\mathbf{1}_S \mid (P\oplus Q)\boldsymbol{\delta}_{i-1}\right) = 1 \ ,$$

then the affine subspaces defined by the two equations in (30) are parallel.

Proof. If $\operatorname{rank}(\mathbf{1}_S \mid (P\oplus Q)\boldsymbol{\delta}_{i-1}) = 1$, then the vector $(P\oplus Q)\boldsymbol{\delta}_{i-1}$ is a multiple of $\mathbf{1}_S$, thus the left kernel of $\mathbf{1}_S$ is contained in the left kernel of $(P\oplus Q)\boldsymbol{\delta}_{i-1}$. Hence the two affine subspaces defined by (30) are parallel. □

Example 1.1. We already know that if $\alpha_{i-1} = \beta_{i-1} = \gamma_{i-1}$, the equation (3) satisfies the first assumption of Proposition 1.12. Since

$$(P\oplus Q)\boldsymbol{\delta}_{i-1} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \alpha_{i-1} \\ \beta_{i-1} \\ \gamma_{i-1} \end{pmatrix} = \begin{pmatrix} \alpha_{i-1} \oplus \beta_{i-1} \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \ ,$$

the equation (3) satisfies also the other assumption of Proposition 1.12, if $\alpha_{i-1} = \beta_{i-1} = \gamma_{i-1}$.

Example 1.2. We know that if $\beta_{i-1} = 0$, then the equation (5) satisfies the first assumption of Proposition 1.12. Since $S = 1$, then the matrix $(\mathbf{1}_S \mid (P \oplus Q)\boldsymbol{\delta}_{i-1})$ contains only one non-zero row, hence the equation (5) satisfies both assumptions of Proposition 1.12 if $\beta_{i-1} = 0$.

Example 1.4. Here we know that the equation (9) satisfies the first assumption of Proposition 1.12 for any i. We calculate

$$(P \oplus Q)\boldsymbol{\delta}_{i-1} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \alpha_{i-1} \\ \beta_{i-1} \end{pmatrix} = \begin{pmatrix} \alpha_{i-1} \oplus \beta_{i-1} \\ \beta_{i-1} \end{pmatrix} .$$

It follows that

$$\operatorname{rank}(\mathbf{1}_S \mid (P \oplus Q)\boldsymbol{\delta}_{i-1}) = \operatorname{rank}\left(\begin{array}{c|c} 1 & \alpha_{i-1} \oplus \beta_{i-1} \\ 1 & \beta_{i-1} \end{array} \right) = 1$$

if and only if $\alpha_{i-1} = 0$.

In case $\operatorname{rank}(\mathbf{1}_S \mid (P \oplus Q)\boldsymbol{\delta}_{i-1}) = 2$ the two equations of the system (30) define two non-parallel affine hyperplanes in $(\mathbb{Z}_2)^S$. Since $S \geq 2$, the two hyperplanes intersect in an affine subspace of codimension 2. So in the case

$$\operatorname{rank}(\mathbf{1}_S \mid (P \oplus Q)\boldsymbol{\delta}_{i-1}) = 2$$

there might be some solutions mod 2^i of (24) that satisfy the equation

$$\mathbf{c}_{i-1}^T (P \oplus Q)\boldsymbol{\delta}_{i-1} = \boldsymbol{\delta}_{i-1}^T (P^T Q)\boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T \boldsymbol{\delta}_i$$

and some that do not satisfy it. We again leave this case aside for the moment and return to the case

$$\operatorname{rank}(\mathbf{1}_S \mid (P \oplus Q)\boldsymbol{\delta}_{i-1}) = 1 \ ,$$

where we can complete the inductions step and decide if a partial solution X mod 2^i is also a partial solution mod 2^{i+1}.

Check No. 4

The final check completes our discussion of equations like (3) and (5). The complete checking tree can be seen in Figure 1.

Proposition 1.13. *We assume that the equation (14) is linear,*

$$\operatorname{rank}\left(\begin{array}{c|c} L \oplus R & \mathbf{1}_S \\ \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T & \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1} \end{array} \right) = 1 \ ,$$

and

$$\operatorname{rank}(\mathbf{1}_S \mid (P \oplus Q)\boldsymbol{\delta}_{i-1}) = 1 \ .$$

Then

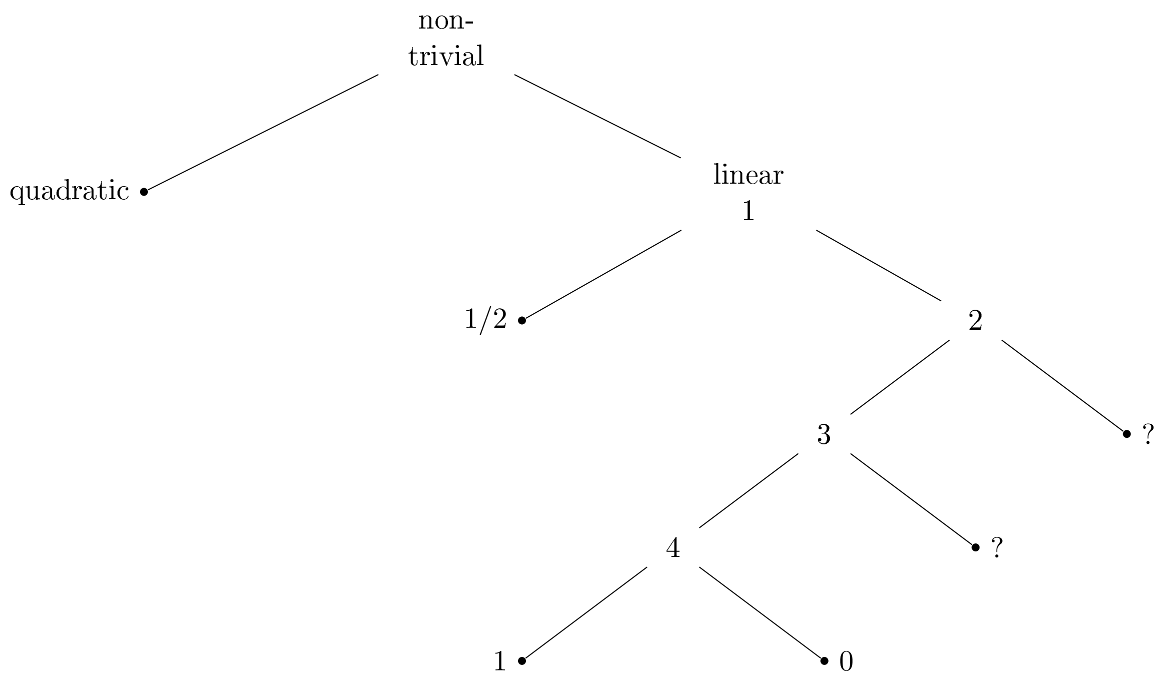

Figure 1: An illustration of the checking tree.

i. every solution X mod 2^i of (14) is also a solution mod 2^{i+1} if

$$\operatorname{rank}\left(\begin{array}{c|c} \mathbf{1}_S & (P \oplus Q)\boldsymbol{\delta}_{i-1} \\ \boldsymbol{\omega}^T\boldsymbol{\delta}_{i-1} & \boldsymbol{\delta}_{i-1}^T(P^TQ)\boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T\boldsymbol{\delta}_i \end{array}\right) = 1 \ ,$$

ii. no solution X mod 2^i of (14) is a solution mod 2^{i+1} if

$$\operatorname{rank}\left(\begin{array}{c|c} \mathbf{1}_S & (P \oplus Q)\boldsymbol{\delta}_{i-1} \\ \boldsymbol{\omega}^T\boldsymbol{\delta}_{i-1} & \boldsymbol{\delta}_{i-1}^T(P^TQ)\boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T\boldsymbol{\delta}_i \end{array}\right) = 2 \ .$$

Proof. We already know by Proposition 1.11 that a solution X mod 2^i is a solution mod 2^{i+1} if and only if the vector $\mathbf{c}_{i-1}^T$ satisfies the equation (29), i.e. if

$$\mathbf{c}_{i-1}^T(P \oplus Q)\boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\delta}_{i-1}^T(P^TQ)\boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T\boldsymbol{\delta}_i = 0 \ .$$

By Proposition 1.12, the affine subspace of $(\mathbb{Z}_2)^S$ defined by (29) is parallel to the affine hyperplane defined by induction hypothesis

$$\mathbf{c}_{i-1}^T\mathbf{1}_S = \boldsymbol{\omega}^T\boldsymbol{\delta}_{i-1} \ .$$

The hyperplane is contained in the affine subspace defined by (29) if and only if the equation (29) is a multiple of the induction hypothesis $\mathbf{c}_{i-1}^T\mathbf{1}_S = \boldsymbol{\omega}^T\boldsymbol{\delta}_{i-1}$.

So if

$$\operatorname{rank}\left(\begin{array}{c|c} \mathbf{1}_S & (P \oplus Q)\boldsymbol{\delta}_{i-1} \\ \boldsymbol{\omega}^T\boldsymbol{\delta}_{i-1} & \boldsymbol{\delta}_{i-1}^T(P^TQ)\boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T\boldsymbol{\delta}_i \end{array}\right) = 1 \ ,$$

then any solution X mod 2^i satisfies (29), hence it is a solution mod 2^{i+1} (with probability 1).

Now assume that

$$\operatorname{rank}\left(\begin{array}{c|c} \mathbf{1}_S & (P \oplus Q)\boldsymbol{\delta}_{i-1} \\ \boldsymbol{\omega}^T\boldsymbol{\delta}_{i-1} & \boldsymbol{\delta}_{i-1}^T(P^TQ)\boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T\boldsymbol{\delta}_i \end{array}\right) = 2 \ .$$

Since $\operatorname{rank}(\mathbf{1}_S \mid (P \oplus Q)\boldsymbol{\delta}_{i-1}) = 1$, we can, if necessary, add the first column to the second one to make all the elements of the second column equal to 0 except the bottom one. And since

$$\operatorname{rank}\left(\begin{array}{c|c} \mathbf{1}_S & (P \oplus Q)\boldsymbol{\delta}_{i-1} \\ \boldsymbol{\omega}^T\boldsymbol{\delta}_{i-1} & \boldsymbol{\delta}_{i-1}^T(P^TQ)\boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T\boldsymbol{\delta}_i \end{array}\right) = 2 \ ,$$

the bottom element of the second column will be equal to 1. It proves that the last row of

$$\left(\begin{array}{c|c} \mathbf{1}_S & (P \oplus Q)\boldsymbol{\delta}_{i-1} \\ \boldsymbol{\omega}^T\boldsymbol{\delta}_{i-1} & \boldsymbol{\delta}_{i-1}^T(P^TQ)\boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T\boldsymbol{\delta}_i \end{array}\right)$$

is not a linear combination of rows of the matrix $(\mathbf{1}_S \mid (P \oplus Q)\boldsymbol{\delta}_{i-1})$, hence the system of two linear equations

$$\mathbf{c}_{i-1}^T (\mathbf{1}_S \mid P \oplus Q) = (\boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1}^T \mid \boldsymbol{\delta}_{i-1}^T (P^T Q)\boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T \boldsymbol{\delta}_i)$$

is unsolvable for $\mathbf{c}_{i-1}^T$. It follows that for any solution X mod 2^i of (14) (i.e. satisfying the induction hypothesis $\mathbf{c}_{i-1}\mathbf{1}_S^T = \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1}$) we get

$$\mathbf{c}_{i-1}^T (P \oplus Q)\boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\delta}_{i-1}^T (P^T Q)\boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T \boldsymbol{\delta}_i \neq 0 \ .$$

By Proposition 1.11, X is not a solution mod 2^{i+1}. □

We apply Proposition 1.13 to the equations (3) and (5).

Example 1.1. We already know that if $\alpha_{i-1} = \beta_{i-1} = \gamma_{i-1}$, the equation (3) satisfies the assumptions of Proposition 1.13.

Since

$$P^T Q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}^T \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \ ,$$

we get

$$\boldsymbol{\delta}_{i-1}^T (P^T Q)\boldsymbol{\delta}_{i-1} = (\alpha_{i-1}, \beta_{i-1}, \gamma_{i-1}) \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \alpha_{i-1} \\ \beta_{i-1} \\ \gamma_{i-1} \end{pmatrix} = \alpha_{i-1}\beta_{i-1} = \alpha_{i-1} \ .$$

And since

$$\boldsymbol{\omega}^T \boldsymbol{\delta}_i = (1,1,1) \begin{pmatrix} \alpha_i \\ \beta_i \\ \gamma_i \end{pmatrix} = \alpha_i \oplus \beta_i \oplus \gamma_i \ ,$$

the rank of

$$\left(\begin{array}{c|c} \mathbf{1}_S & (P \oplus Q)\boldsymbol{\delta}_{i-1} \\ \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1} & \boldsymbol{\delta}_{i-1}^T (P^T Q)\boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T \boldsymbol{\delta}_i \end{array} \right) = \left(\begin{array}{c|c} 1 & 0 \\ 1 & 0 \\ \alpha_{i-1} & \alpha_{i-1} \oplus \alpha_i \oplus \beta_i \oplus \gamma_i \end{array} \right)$$

is equal to 1 if and only if $\alpha_{i-1} = \alpha_i \oplus \beta_i \oplus \gamma_i$.

Example 1.2. We know that if $\beta_{i-1} = 0$, then the equation (5) satisfies the first assumption of Proposition 1.13. Since

$$P^T Q = \begin{pmatrix} 0 \\ 0 \end{pmatrix} (1,0) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \ ,$$

we get $\boldsymbol{\delta}_{i-1}^T (P^T Q)\boldsymbol{\delta}_{i-1} = 0$. And since

$$\boldsymbol{\omega}^T \boldsymbol{\delta}_i = (1,1) \begin{pmatrix} \alpha_i \\ \beta_i \end{pmatrix} = \alpha_i \oplus \beta_i \ ,$$

We obtain

$$\left(\begin{array}{c|c} \mathbf{1}_S & (P\oplus Q)\boldsymbol{\delta}_{i-1} \\ \boldsymbol{\omega}^T\boldsymbol{\delta}_{i-1} & \boldsymbol{\delta}_{i-1}^T(P^TQ)\boldsymbol{\delta}_{i-1}\oplus\boldsymbol{\omega}^T\boldsymbol{\delta}_i \end{array}\right) = \left(\begin{array}{c|c} 1 & \alpha_{i-1} \\ \alpha_{i-1}\oplus\beta_{i-1} & \alpha_i\oplus\beta_i \end{array}\right) .$$

So if $\beta_{i-1}=0$, the matrix

$$\left(\begin{array}{c|c} 1 & \alpha_{i-1} \\ \alpha_{i-1}\oplus\beta_{i-1} & \alpha_i\oplus\beta_i \end{array}\right) = \left(\begin{array}{c|c} 1 & \alpha_{i-1} \\ \alpha_{i-1} & \alpha_i\oplus\beta_i \end{array}\right) \sim \left(\begin{array}{c|c} 1 & 0 \\ \alpha_{i-1} & \alpha_{i-1}\oplus\alpha_i\oplus\beta_i \end{array}\right)$$

has rank equal to 1 if and only if $\alpha_{i-1}=\alpha_i\oplus\beta_i$.

We can apply Proposition 1.13 also to the equation (9).

Example 1.4. We already know that equation (9) satisfies both assumptions of Proposition 1.13 if and only if $\alpha_{i-1}=0$. We get

$$P^TQ = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}^T \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} ,$$

thus $\boldsymbol{\delta}_{i-1}^T(P^TQ)\boldsymbol{\delta}_{i-1}=\alpha_{i-1}\beta_{i-1}$. Since $\boldsymbol{\omega}^T=(0,0)$, we obtain $\boldsymbol{\omega}^T\boldsymbol{\delta}_{i-1}=\boldsymbol{\omega}^T\boldsymbol{\delta}_i=0$. So the matrix

$$\left(\begin{array}{c|c} \mathbf{1}_S & (P\oplus Q)\boldsymbol{\delta}_{i-1} \\ \boldsymbol{\omega}^T\boldsymbol{\delta}_{i-1} & \boldsymbol{\delta}_{i-1}^T(P^TQ)\boldsymbol{\delta}_{i-1}\oplus\boldsymbol{\omega}^T\boldsymbol{\delta}_i \end{array}\right) = \left(\begin{array}{c|c} 1 & \alpha_{i-1}\oplus\beta_{i-1} \\ 1 & \beta_{i-1} \\ 0 & \alpha_{i-1}\beta_{i-1} \end{array}\right) = \left(\begin{array}{c|c} 1 & \beta_{i-1} \\ 1 & \beta_{i-1} \\ 0 & 0 \end{array}\right)$$

has rank 1. We conclude that if $\alpha_{i-1}=0$, then every solution X mod 2^i is also a solution mod 2^{i+1} by Proposition 1.13.i.

The following theorem describes a large class of AX-equations for which the solubility condition depends only on pairs of subsequent vectors $\boldsymbol{\delta}_{i-1}$, $\boldsymbol{\delta}_i$. It also gives a precise value of the probability that a randomly selected matrix X solves an equation from this class.

Theorem 1.14. *We assume that the equation (14) is linear, and if*

$$\operatorname{rank}\left(\begin{array}{c|c} L\oplus R & \mathbf{1}_S \\ \boldsymbol{\delta}_{i-1}^TA\oplus\boldsymbol{\varepsilon}^T & \boldsymbol{\omega}^T\boldsymbol{\delta}_{i-1} \end{array}\right) = \operatorname{rank}\left(\begin{array}{c|c} L\oplus R & \mathbf{1}_S \end{array}\right) ,$$

then both

$$\operatorname{rank}\left(\begin{array}{c|c} L\oplus R & \mathbf{1}_S \\ \boldsymbol{\delta}_{i-1}^TA\oplus\boldsymbol{\varepsilon}^T & \boldsymbol{\omega}^T\boldsymbol{\delta}_{i-1} \end{array}\right) = 1 ,$$

and

$$\operatorname{rank}\left(\mathbf{1}_S \mid (P\oplus Q)\boldsymbol{\delta}_{i-1}\right) = 1 .$$

Then the equation (14) is solvable if and only if

$$\operatorname{rank}\left(\begin{array}{c|c} \mathbf{1}_S & (P\oplus Q)\boldsymbol{\delta}_{i-1} \\ \boldsymbol{\omega}^T\boldsymbol{\delta}_{i-1} & \boldsymbol{\delta}_{i-1}^T(P^TQ)\boldsymbol{\delta}_{i-1}\oplus\boldsymbol{\omega}^T\boldsymbol{\delta}_i \end{array}\right) = 1$$

whenever

$$\operatorname{rank}\left(\begin{array}{c|c} L\oplus R & \mathbf{1}_S \\ \boldsymbol{\delta}_{i-1}^T A\oplus \boldsymbol{\varepsilon}^T & \boldsymbol{\omega}^T\boldsymbol{\delta}_{i-1}\end{array}\right) = \operatorname{rank}\left(\begin{array}{c|c} L\oplus R & \mathbf{1}_S\end{array}\right) \ .$$

If this is this case, then the probability that a randomly selected matrix X of type $N\times p$ solves the equation (14) is 2^{-p+k}, where

$$k = \#\left\{ i\in\mathbb{Z}_p : \operatorname{rank}\left(\begin{array}{c|c} L\oplus R & \mathbf{1}_S \\ \boldsymbol{\delta}_{i-1}^T A\oplus \boldsymbol{\varepsilon}^T & \boldsymbol{\omega}^T\boldsymbol{\delta}_{i-1}\end{array}\right) = \operatorname{rank}\left(\begin{array}{c|c} L\oplus R & \mathbf{1}_S\end{array}\right)\right\} \ .$$

Proof. If for some $i = 0,\dots,p-1$,

$$\operatorname{rank}\left(\begin{array}{c|c} L\oplus R & \mathbf{1}_S \\ \boldsymbol{\delta}_{i-1}^T A\oplus \boldsymbol{\varepsilon}^T & \boldsymbol{\omega}^T\boldsymbol{\delta}_{i-1}\end{array}\right) > \operatorname{rank}\left(\begin{array}{c|c} L\oplus R & \mathbf{1}_S\end{array}\right) \ ,$$

then a solution X mod 2^i is a solution mod 2^{i+1} with probability 2^{-1} by Proposition 1.10.

If

$$\operatorname{rank}\left(\begin{array}{c|c} L\oplus R & \mathbf{1}_S \\ \boldsymbol{\delta}_{i-1}^T A\oplus \boldsymbol{\varepsilon}^T & \boldsymbol{\omega}^T\boldsymbol{\delta}_{i-1}\end{array}\right) = \operatorname{rank}\left(\begin{array}{c|c} L\oplus R & \mathbf{1}_S\end{array}\right) \ ,$$

then all the assumption of Proposition 1.13 are satisfied. By Proposition 1.13.ii, no solution X mod 2^i is a solution mod 2^{i+1} if

$$\operatorname{rank}\left(\begin{array}{c|c} \mathbf{1}_S & (P\oplus Q)\boldsymbol{\delta}_{i-1} \\ \boldsymbol{\omega}^T\boldsymbol{\delta}_{i-1} & \boldsymbol{\delta}_{i-1}^T(P^TQ)\boldsymbol{\delta}_{i-1}\oplus\boldsymbol{\omega}^T\boldsymbol{\delta}_i\end{array}\right) = 2 \ .$$

So if the equation (14) is soluble, we must have

$$\operatorname{rank}\left(\begin{array}{c|c} \mathbf{1}_S & (P\oplus Q)\boldsymbol{\delta}_{i-1} \\ \boldsymbol{\omega}^T\boldsymbol{\delta}_{i-1} & \boldsymbol{\delta}_{i-1}^T(P^TQ)\boldsymbol{\delta}_{i-1}\oplus\boldsymbol{\omega}^T\boldsymbol{\delta}_i\end{array}\right) = 1 \ .$$

If this is the case, then by Proposition 1.13.i, any solution X mod 2^i is also a solution mod 2^{i+1}.

The precise value of the probability that a randomly selected matrix X solves the equation (14) follows directly from the proof. □

From the previous discussions of Example 1.1, we obtain immediately the result of [14] as a corollary of Theorem 1.14.

Corollary 1.15. *The equation (2)*

$$(\mathbf{x}\oplus\boldsymbol{\alpha}) + (\mathbf{y}\oplus\boldsymbol{\beta}) \quad = \quad (\mathbf{x}+\mathbf{y})\oplus\boldsymbol{\gamma}$$

is solvable if and only if for every $i = 0,\dots,p-1$,

$$\alpha_{i-1} = \beta_{i-1} = \gamma_{i-1} \quad \Rightarrow \quad \alpha_{i-1} = \alpha_i\oplus\beta_i\oplus\gamma_i \ . \tag{31}$$

If this is true, then a matrix X solves the equation with probability 2^{-p+k}, where $k = \#\{i\in\{0,1,\dots,p-1\} : \alpha_{i-1} = \beta_{i-1} = \gamma_{i-1}\}$.

In the same way, we obtain the results of [12] on the equation (5) as another corollary of Theorem 1.14.

Corollary 1.16. *The equation (4)*

$$\mathbf{x} + \boldsymbol{\alpha} \quad = \quad \mathbf{x} \oplus \boldsymbol{\beta}$$

is solvable if and only if for every $i = 0, \ldots, p-1$,

$$\beta_{i-1} = 0 \quad \Rightarrow \quad \alpha_{i-1} = \alpha_i \oplus \beta_i \ . \tag{32}$$

If this is true, then a matrix X solves the equation with probability 2^{-p+k}, where $k = \#\{i \in \{0, 1, \ldots, p-1\} : \beta_{i-1} = 0\}$.

1.9 Recursion

During the previous discussion of the linear case we left aside two possibilities. Starting from Check No. 3 we did not consider the case

$$\operatorname{rank}\left(\ L \oplus R \ \middle| \ \mathbf{1}_S \ \right) = \operatorname{rank}\left(\begin{array}{c|c} L \oplus R & \mathbf{1}_S \\ \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T & \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1} \end{array} \right) \geq 2 \ .$$

and under the assumption that

$$\operatorname{rank}\left(\ L \oplus R \ \middle| \ \mathbf{1}_S \ \right) = \operatorname{rank}\left(\begin{array}{c|c} L \oplus R & \mathbf{1}_S \\ \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T & \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1} \end{array} \right) = 1 \ ,$$

we left aside the possibility that

$$\operatorname{rank}\left(\mathbf{1}_S \mid (P \oplus Q)\boldsymbol{\delta}_{i-1}\right) = 2 \ . \tag{33}$$

If $\operatorname{rank}\left(\mathbf{1}_S \mid (P \oplus Q)\boldsymbol{\delta}_{i-1}\right) = 2$, then also

$$\operatorname{rank}\left(\begin{array}{c|c} \mathbf{1}_S & (P \oplus Q)\boldsymbol{\delta}_{i-1} \\ \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1} & \boldsymbol{\delta}_{i-1}^T (P^T Q)\boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T \boldsymbol{\delta}_i \end{array} \right) = 2 \ .$$

So we cannot conclude that the equation

$$\mathbf{c}_{i-1}^T (P \oplus Q)\boldsymbol{\delta}_{i-1} = \boldsymbol{\delta}_{i-1}^T (P^T Q)\boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T \boldsymbol{\delta}_i$$

has no solution like we did in Proposition 1.13 under the assumptions

$$\operatorname{rank}\left(\begin{array}{c|c} L \oplus R & \mathbf{1}_S \\ \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T & \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1} \end{array} \right) = 1 \ ,$$

and

$$\operatorname{rank}\left(\mathbf{1}_S \mid (P \oplus Q)\boldsymbol{\delta}_{i-1}\right) = 1 \ .$$

So we start with the discussion when

$$\mathbf{c}_{i-1}^T(P \oplus Q)\boldsymbol{\delta}_{i-1} = \boldsymbol{\delta}_{i-1}^T(P^TQ)\boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T\boldsymbol{\delta}_i$$

if $\operatorname{rank}(\mathbf{1}_S \mid (P \oplus Q)\boldsymbol{\delta}_{i-1}) = 2$. Then $(P \oplus Q)\boldsymbol{\delta}_{i-1}$ is neither $\mathbf{1}_S$ nor $\mathbf{0}_S$. So the number of 1's in the vector $(P \oplus Q)\boldsymbol{\delta}_{i-1}$ is positive and less than S.

Let $1 \le s_1 < s_2 < \cdots < s_m \le S$ be the indices of all coordinates in vector $(P \oplus Q)\boldsymbol{\delta}_{i-1}$ that are equal to 1. To find out if

$$\mathbf{c}_{i-1}^T(P \oplus Q)\boldsymbol{\delta}_{i-1} = \boldsymbol{\delta}_{i-1}^T(P^TQ)\boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T\boldsymbol{\delta}_i$$

we expand the carries $c_{s_j,i-1}$ for $j = 1, 2, \ldots, m$. We have

$$\begin{aligned} c_{s_j,i-1} &= \mathbf{x}_{i-2}^T(\tilde{\mathbf{l}}_{s_j}\tilde{\mathbf{r}}_{s_j}^T)\mathbf{x}_{i-2} \oplus \boldsymbol{\delta}_{i-2}^T\tilde{\mathbf{q}}_{s_j}\tilde{\mathbf{l}}_{s_j}^T\mathbf{x}_{i-2} \oplus \boldsymbol{\delta}_{i-2}^T\tilde{\mathbf{p}}_{s_j}\tilde{\mathbf{r}}_{s_j}^T\mathbf{x}_{i-2} \oplus \\ &\quad c_{s_j,i-2}(\tilde{\mathbf{l}}_{s_j}^T \oplus \tilde{\mathbf{r}}_{s_j}^T)\mathbf{x}_{i-2} \oplus c_{s_j,i-2}(\tilde{\mathbf{p}}_{s_j}^T \oplus \tilde{\mathbf{q}}_{s_j}^T)\boldsymbol{\delta}_{i-2} \oplus \boldsymbol{\delta}_{i-2}^T(\tilde{\mathbf{p}}_{s_j}\tilde{\mathbf{q}}_{s_j}^T)\boldsymbol{\delta}_{i-2} \ . \end{aligned}$$

Since $\mathbf{c}_{i-1}^T(P \oplus Q)\boldsymbol{\delta}_{i-1} = c_{s_1,i-1} \oplus c_{s_2,i-1} \oplus \cdots \oplus c_{s_m,i-1}$, we get

$$\begin{aligned} \mathbf{c}_{i-1}^T(P \oplus Q)\boldsymbol{\delta}_{i-1} &= \bigoplus_{j=1}^m \left(\mathbf{x}_{i-2}^T(\tilde{\mathbf{l}}_{s_j}\tilde{\mathbf{r}}_{s_j}^T)\mathbf{x}_{i-2}\right) \oplus \bigoplus_{j=1}^m \left(\boldsymbol{\delta}_{i-2}^T\tilde{\mathbf{q}}_{s_j}\tilde{\mathbf{l}}_{s_j}^T\mathbf{x}_{i-2}\right) \oplus \\ &\quad \bigoplus_{j=1}^m \left(\boldsymbol{\delta}_{i-2}^T\tilde{\mathbf{p}}_{s_j}\tilde{\mathbf{r}}_{s_j}^T\mathbf{x}_{i-2}\right) \oplus \bigoplus_{j=1}^m \left(c_{s_j,i-2}(\tilde{\mathbf{l}}_{s_j}^T \oplus \tilde{\mathbf{r}}_{s_j}^T)\mathbf{x}_{i-2}\right) \oplus \\ &\quad \bigoplus_{j=1}^m \left(c_{s_j,i-2}(\tilde{\mathbf{p}}_{s_j}^T \oplus \tilde{\mathbf{q}}_{s_j}^T)\boldsymbol{\delta}_{i-2}\right) \oplus \bigoplus_{j=1}^m \left(\boldsymbol{\delta}_{i-2}^T(\tilde{\mathbf{p}}_{s_j}\tilde{\mathbf{q}}_{s_j}^T)\boldsymbol{\delta}_{i-2}\right) . \end{aligned}$$

Using once again dyadic expansion of a product of two matrices, we obtain

$$\bigoplus_{j=1}^m \mathbf{x}_{i-2}^T(\tilde{\mathbf{l}}_{s_j}\tilde{\mathbf{r}}_{s_j}^T)\mathbf{x}_{i-2} = \mathbf{x}_{i-2}^T \left(\bigoplus_{j=1}^m \tilde{\mathbf{l}}_{s_j}\tilde{\mathbf{r}}_{s_j}^T\right) \mathbf{x}_{i-2} = \mathbf{x}_{i-2}^T L_{i-1}^T R_{i-1}\mathbf{x}_{i-2} \ ,$$

where the matrices L_{i-1}, R_{i-1}, respectively are obtained from L, R, respectively by omitting all rows with indices different from any s_j, $j = 1, \ldots, m$.

The matrices L_{i-1}, R_{i-1} can be described in more straight forward way using the following *selection matrix* $E_{i-1} = (e_{j,k})$ of type $m \times S$, where

$$e_{j,k} = \begin{cases} 1 & \text{if } k = s_j \\ 0 & \text{otherwise.} \end{cases}$$

Then $L_{i-1} = E_{i-1}L$ and $R_{i-1} = E_{i-1}R$. Similarly, we set $P_{i-1} = E_{i-1}P$ and $Q_{i-1} = E_{i-1}Q$.

Similarly, we obtain

$$\bigoplus_{j=1}^{m} \boldsymbol{\delta}_{i-2}^T \tilde{\mathbf{q}}_{s_j} \tilde{\mathbf{l}}_{s_j}^T \mathbf{x}_{i-2} = \boldsymbol{\delta}_{i-2}^T (Q_{i-1}^T L_{i-1}) \mathbf{x}_{i-2} \ ,$$

$$\bigoplus_{j=1}^{m} \boldsymbol{\delta}_{i-2}^T \tilde{\mathbf{p}}_{s_j} \tilde{\mathbf{r}}_{s_j}^T \mathbf{x}_{i-2} = \boldsymbol{\delta}_{i-2}^T (P_{i-1}^T R_{i-1}) \mathbf{x}_{i-2} \ ,$$

$$\bigoplus_{j=1}^{m} \boldsymbol{\delta}_{i-2}^T (\tilde{\mathbf{p}}_{s_j} \tilde{\mathbf{q}}_{s_j}^T) \boldsymbol{\delta}_{i-2} = \boldsymbol{\delta}_{i-2}^T (P_{i-1}^T Q_{i-1}) \boldsymbol{\delta}_{i-2} \ .$$

The other two sums are

$$\bigoplus_{j=1}^{m} c_{s_j,i-2} (\tilde{\mathbf{l}}_{s_j}^T \oplus \tilde{\mathbf{r}}_{s_j}^T) \mathbf{x}_{i-2} = \mathbf{c}_{i-2}^T E_{i-1}^T (L_{i-1} \oplus R_{i-1}) \mathbf{x}_{i-2} \ ,$$

$$\bigoplus_{j=1}^{m} c_{s_j,i-2} (\tilde{\mathbf{p}}_{s_j}^T \oplus \tilde{\mathbf{q}}_{s_j}^T) \boldsymbol{\delta}_{i-2} = \mathbf{c}_{i-2}^T E_{i-1}^T (P_{i-1} \oplus Q_{i-1}) \boldsymbol{\delta}_{i-2} \ .$$

Thus

$$\begin{aligned} \mathbf{c}_{i-1}^T (P \oplus Q) \boldsymbol{\delta}_{i-1} \quad = \quad & \mathbf{x}_{i-2}^T L_{i-1}^T R_{i-1} \mathbf{x}_{i-2} \oplus \boldsymbol{\delta}_{i-2}^T (Q_{i-1}^T L_{i-1} \oplus P_{i-1}^T R_{i-1}) \mathbf{x}_{i-2} \oplus \\ & \mathbf{c}_{i-2}^T E_{i-1}^T (L_{i-1} \oplus R_{i-1}) \mathbf{x}_{i-2} \oplus \mathbf{c}_{i-2}^T E_{i-1}^T (P_{i-1} \oplus Q_{i-1}) \boldsymbol{\delta}_{i-2} \oplus \\ & \boldsymbol{\delta}_{i-2}^T (P_{i-1}^T Q_{i-1}) \boldsymbol{\delta}_{i-2}. \end{aligned} \tag{34}$$

Note also that

$$E_{i-1}^T E_{i-1} = \mathrm{Diag}[(P \oplus Q) \boldsymbol{\delta}_{i-1}] \ ,$$

so $E_{i-1}^T E_{i-1}$ has at least one row equal to $\mathbf{0}_S^T$. Also $E_{i-1}^T (L_{i-1} \oplus R_{i-1}) = E_{i-1}^T E_{i-1} (L \oplus R)$ and $E_{i-1}^T (P_{i-1} \oplus Q_{i-1}) = E_{i-1}^T E_{i-1} (P \oplus Q)$. If we set $F_{i-1} = E_{i-1}^T E_{i-1}$, we can rewrite the equation (34) as

$$\begin{aligned} \mathbf{c}_{i-1}^T (P \oplus Q) \boldsymbol{\delta}_{i-1} \quad = \quad & \mathbf{x}_{i-2}^T L_{i-1}^T R_{i-1} \mathbf{x}_{i-2} \oplus \boldsymbol{\delta}_{i-2}^T (Q_{i-1}^T L_{i-1} \oplus P_{i-1}^T R_{i-1}) \mathbf{x}_{i-2} \oplus \\ & \mathbf{c}_{i-2}^T F_{i-1} (L \oplus R) \mathbf{x}_{i-2} \oplus \mathbf{c}_{i-2}^T F_{i-1}^T (P \oplus Q) \boldsymbol{\delta}_{i-2} \oplus \\ & \boldsymbol{\delta}_{i-2}^T (P_{i-1}^T Q_{i-1}) \boldsymbol{\delta}_{i-2}. \end{aligned}$$

So we have proved the following proposition.

Proposition 1.17. *If (14) is a linear equation and* $\mathrm{rank}\,(\mathbf{1}_S \mid (P \oplus Q)\boldsymbol{\delta}_{i-1}) = 2$*, then for a solution X mod 2^i of (14) the equality*

$$\mathbf{c}_{i-1}^T (P \oplus Q) \boldsymbol{\delta}_{i-1} = \boldsymbol{\delta}_{i-1}^T (P^T Q) \boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T \boldsymbol{\delta}_i$$

holds if and only if

$$\begin{aligned} & \mathbf{x}_{i-2}^T L_{i-1}^T R_{i-1} \mathbf{x}_{i-2} \oplus \boldsymbol{\delta}_{i-2}^T (Q_{i-1}^T L_{i-1} \oplus P_{i-1}^T R_{i-1}) \mathbf{x}_{i-2} \oplus \\ & \mathbf{c}_{i-2}^T F_{i-1}^T (L \oplus R) \mathbf{x}_{i-2} \oplus \mathbf{c}_{i-2}^T F_{i-1}^T (P \oplus Q) \boldsymbol{\delta}_{i-2} \oplus \\ & \boldsymbol{\delta}_{i-2}^T (P_{i-1}^T Q_{i-1}) \boldsymbol{\delta}_{i-2} = \boldsymbol{\delta}_{i-1}^T (P^T Q) \boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T \boldsymbol{\delta}_i \ . \end{aligned} \tag{35}$$

The equation (35) is of the same type as (21). So we can use the checking tree with matrices L_{i-1}, R_{i-1}, P_{i-1} and Q_{i-1} to check if the equation (35) is solvable. But we have to keep in mind that X is a solution mod 2^i of (14), hence it satisfies equations

$$\mathbf{c}_{i-1}^T \mathbf{1}_S = \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1}$$

for each $j = 0, \dots, i-1$.

Although we are convinced that general theory of solubility of the system of equations (35) along the checking tree is possible it becomes technically more complicated in case the matrix the matrix $L_{i-1}^T R_{i-1}$ is not symmetric, and we will not attempt to develop it in full. Instead we just show how the recursion works for the equation (9). A general theory of the recursive cases is a future work extending this section.

First we summarize some results of sections 1.8 and 1.9. that apply also to the case when

$$\operatorname{rank}\left(\begin{array}{c|c} L \oplus R & \mathbf{1}_S \end{array} \right) = \operatorname{rank}\left(\begin{array}{c|c} L \oplus R & \mathbf{1}_S \\ \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T & \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1} \end{array} \right) \geq 2 \ .$$

Proposition 1.18. *Let (14) be a linear equation. Then the following holds*

i. A solution X mod 2^i of equation (14) is a solution mod 2^{i+1} with probability 2^{-1} if

$$\mathbf{c}_{i-1}^T (L \oplus R) \neq \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T \ .$$

ii. If $\mathbf{c}_{i-1}^T (L \oplus R) = \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T$, then a solution X mod 2^i of equation (14) is a solution mod 2^{i+1} if and only if

$$\mathbf{c}_{i-1}^T (P \oplus Q) \boldsymbol{\delta}_{i-1} = \boldsymbol{\delta}_{i-1}^T (P^T Q) \boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T \boldsymbol{\delta}_i \ .$$

Proof. We already know that in the linear case the a solution mod 2^i of (14) is a solution mod 2^{i+1} if and only if the equation (24) holds, that is if

$$\begin{aligned} &\left[\mathbf{c}_{i-1}^T (L \oplus R) \oplus \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T\right] \mathbf{x}_{i-1} = \\ &\mathbf{c}_{i-1}^T (P \oplus Q) \boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\delta}_{i-1}^T (P^T Q) \boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T \boldsymbol{\delta}_i \ . \end{aligned}$$

i. Hence if $\mathbf{c}_{i-1}^T (L \oplus R) = \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T$, then the equation (24) has a solution. Exactly one half of possible $\mathbf{x}_{i-1}$'s solves it.

ii. If $\mathbf{c}_{i-1}^T (L \oplus R) = \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T$ then the left-hand side of (24) is equal to 0 so the equality holds if and only if the right-hand side of (24) is equal to 0.

□

We return for the last time to the equation (9).

Example 1.4. We already know that

$$(L \oplus R \mid \mathbf{1}_S) = \left(\begin{array}{c|c} 1 & 1 \\ 1 & 1 \end{array}\right) \ ,$$

$$\left(\begin{array}{c|c} L \oplus R & \mathbf{1}_S \\ \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T & \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1} \end{array}\right) = \left(\begin{array}{c|c} 1 & 1 \\ 1 & 1 \\ 0 & 0 \end{array}\right) \ ,$$

Hence $\operatorname{rank}(L \oplus R \mid \mathbf{1}_S) = \operatorname{rank}\left(\begin{array}{c|c} L \oplus R & \mathbf{1}_S \\ \boldsymbol{\delta}_{i-1}^T A \oplus \boldsymbol{\varepsilon}^T & \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1} \end{array}\right) = 1.$

Moreover

$$(\mathbf{1}_S \mid (P \oplus Q)\boldsymbol{\delta}_{i-1}) = \left(\begin{array}{c|c} 1 & \alpha_{i-1} \oplus \beta_{i-1} \\ 1 & \beta_{i-1} \end{array}\right)$$

hence $\operatorname{rank}(\mathbf{1}_S \mid (P \oplus Q)\boldsymbol{\delta}_{i-1}) = \operatorname{rank}\left(\begin{array}{c|c} 1 & \alpha_{i-1} \oplus \beta_{i-1} \\ 1 & \beta_{i-1} \end{array}\right) = 1$ if and only if $\alpha_{i-1} = 0$. In this case

$$\left(\begin{array}{c|c} \mathbf{1}_S & (P \oplus Q)\boldsymbol{\delta}_{i-1} \\ \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1} & \boldsymbol{\delta}_{i-1}^T (P^T Q)\boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T \boldsymbol{\delta}_i \end{array}\right) = \left(\begin{array}{c|c} 1 & \alpha_{i-1} \oplus \beta_{i-1} \\ 1 & \beta_{i-1} \\ 0 & \alpha_{i-1}\beta_{i-1} \end{array}\right) = \left(\begin{array}{c|c} 1 & \beta_{i-1} \\ 1 & \beta_{i-1} \\ 0 & 0 \end{array}\right)$$

hence every solution $X \bmod 2^i$ of equation (9) is a solution mod 2^{i+1} by Proposition 1.13.

It remains to consider the case $\alpha_{i-1} = 1$. Then the equation (9) does not satisfy the assumptions of Proposition 1.13, but still satisfies the assumptions of Proposition 1.11. So a solution $X \bmod 2^i$ of equation (9) is a solution mod 2^{i+1} if and only if

$$\mathbf{c}_{i-1}^T (P \oplus Q)\boldsymbol{\delta}_{i-1} = \boldsymbol{\delta}_{i-1}^T (P^T Q)\boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T \boldsymbol{\delta}_i \ .$$

Since $\mathbf{c}_{i-1}^T$ also satisfies the induction hypothesis $\mathbf{c}_{i-1}^T \mathbf{1}_S = \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1}$, the vector $\mathbf{c}_{i-1}^T$ must solve the system of linear equations with the extended matrix

$$\left(\begin{array}{c|c} \mathbf{1}_S & (P \oplus Q)\boldsymbol{\delta}_{i-1} \\ \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1} & \boldsymbol{\delta}_{i-1}^T (P^T Q)\boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T \boldsymbol{\delta}_i \end{array}\right) = \left(\begin{array}{c|c} 1 & 1 \oplus \beta_{i-1} \\ 1 & \beta_{i-1} \\ 0 & \beta_{i-1} \end{array}\right) \sim \left(\begin{array}{c|c} 1 & 1 \\ 1 & 0 \\ 0 & \beta_{i-1} \end{array}\right) \ .$$

The first equation (column) is the induction hypothesis so a solution $X \bmod 2^i$ is a solution mod 2^{i+1} (with probability 1) if and only

$$\mathbf{c}_{i-1}^T \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \beta_{i-1} \ . \tag{36}$$

Under the induction hypothesis $\mathbf{c}_{i-1}^T \mathbf{1}_S = \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-1} = 0$, this equation is equivalent to

$$\mathbf{c}_{i-1}^T (P \oplus Q) \boldsymbol{\delta}_{i-1} = \boldsymbol{\delta}_{i-1}^T (P^T Q) \boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T \boldsymbol{\delta}_i \ , \tag{37}$$

so we start we start the recursion with (36).

Now we transform (36) (i.e. (37)) to (35) using the selection matrix $E_{i-1} = (1, 0)$. Since $L_{i-1}^T R_{i-1} = (0)$, $\boldsymbol{\varepsilon}_{i-1} = \mathrm{diag}(L_{i-1}^T R_{i-1}) = (0)$, the equation (35) is in fact linear in $\mathbf{x}_{i-2}$ and can be written in the form

$$\begin{aligned} &\left[\mathbf{c}_{i-2}^T F_{i-1}^T (L \oplus R) \oplus \boldsymbol{\delta}_{i-2}^T (A_{i-1}) \oplus \boldsymbol{\varepsilon}_{i-1}^T\right] \mathbf{x}_{i-2} = \\ &\mathbf{c}_{i-2}^T F_{i-1}^T (P \oplus Q) \boldsymbol{\delta}_{i-2} \oplus \boldsymbol{\delta}_{i-2}^T (P_{i-1}^T Q_{i-1}) \boldsymbol{\delta}_{i-2} \oplus \boldsymbol{\delta}_{i-1}^T (P^T Q) \boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T \boldsymbol{\delta}_i \ . \end{aligned} \tag{38}$$

since

$$A_{i-1} = Q_{i-1}^T L_{i-1} \oplus P_{i-1}^T R_{i-1} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} (1) \oplus \begin{pmatrix} 1 \\ 0 \end{pmatrix} (0) = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \ ,$$

$$F_{i-1}^T (L \oplus R) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad F_{i-1}^T (P \oplus Q) = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \ ,$$

after further substitutions (38) takes the form

$$\left[\mathbf{c}_{i-2}^T \begin{pmatrix} 1 \\ 0 \end{pmatrix} \oplus \beta_{i-2}\right] \mathbf{x}_{i-2} = \mathbf{c}_{i-2}^T \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \alpha_{i-2} \\ \beta_{i-2} \end{pmatrix} \oplus \alpha_{i-2}\beta_{i-2} \oplus \beta_{i-1} \ , \tag{39}$$

since we consider the case $\alpha_{i-1} = 1$.

Now the column vector

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} = F_{i-1}(L \oplus R), \quad \beta_{i-2} = \boldsymbol{\delta}_{i-2}^T A_{i-1} \oplus \boldsymbol{\varepsilon}_{i-1}^T \ ,$$

so we get

$$\mathrm{rank}(F_{i-1}(L \oplus R) \mid \mathbf{1}_S) = \mathrm{rank} \left(\begin{array}{c|c} 1 & 1 \\ 0 & 1 \end{array} \right) = 2 \ ,$$

and consequently

$$\mathrm{rank} \left(\begin{array}{c|c} 1 & 1 \\ 0 & 1 \end{array} \right) = \mathrm{rank} \left(\begin{array}{c|c} 1 & 1 \\ 0 & 1 \\ \beta_{i-2} & 0 \end{array} \right) = \mathrm{rank} \left(\begin{array}{c|c} F_{i-1}(L \oplus R) & \mathbf{1}_S \\ \boldsymbol{\delta}_{i-2}^T A_{i-1} \oplus \boldsymbol{\varepsilon}_{i-1}^T & \boldsymbol{\omega}^T \boldsymbol{\delta}_{i-2} \end{array} \right) \ .$$

So we cannot use Proposition 1.11 and have to rely on Proposition 1.18.

Since X is a solution mod 2^{i-1} and $\mathbf{c}_{i-2}^T \begin{pmatrix} 1 \\ 0 \end{pmatrix} \neq \beta_{i-2}$, the equality (39), which is the form of (35) after substitutions, holds with probability 2^{-1}. Hence any X satisfying the equality (39) satisfies the equality (35) and by Proposition 1.17 satisfies

$$\mathbf{c}_{i-1}^T (P \oplus Q) \boldsymbol{\delta}_{i-1} = \boldsymbol{\delta}_{i-1}^T (P^T Q) \boldsymbol{\delta}_{i-1} \oplus \boldsymbol{\omega}^T \boldsymbol{\delta}_i \ .$$

Thus it is a solution mod 2^{i+1}.

If $\mathbf{c}_{i-2}^T \begin{pmatrix}1\\0\end{pmatrix} = \beta_{i-2}$, then the right-hand side of (39) becomes

$$\begin{aligned}
&\mathbf{c}_{i-2}^T \begin{pmatrix}1 & 1\\0 & 0\end{pmatrix} \begin{pmatrix}\alpha_{i-2}\\\beta_{i-2}\end{pmatrix} \oplus \alpha_{i-2}\beta_{i-2} \oplus \beta_{i-1} = \\
&\mathbf{c}_{i-2}^T \begin{pmatrix}1\\0\end{pmatrix} \alpha_{i-2} \oplus \mathbf{c}_{i-2}^T \begin{pmatrix}1\\0\end{pmatrix} \beta_{i-2} \oplus \alpha_{i-2}\beta_{i-2} \oplus \beta_{i-1} = \\
&\beta_{i-2}\alpha_{i-2} \oplus \beta_{i-2}\beta_{i-2} \oplus \alpha_{i-2}\beta_{i-2} \oplus \beta_{i-1} = \\
&\beta_{i-2} \oplus \beta_{i-1} \ .
\end{aligned}$$

Thus the equation (39) holds if and only if $\beta_{i-2} = \beta_{i-1}$.

So if $\beta_{i-1} \neq \beta_{i-2}$ and $\mathbf{c}_{i-2}^T \begin{pmatrix}1\\0\end{pmatrix} = \beta_{i-2}$ for every solution mod 2^i, then none of them is a solution mod 2^{i+1} and the equation (9) has no solution.

By what we have just proved, $\mathbf{c}_{i-2}^T \begin{pmatrix}1\\0\end{pmatrix} = \beta_{i-2}$ for every solution mod 2^i if $\beta_{i-2} = \beta_{i-3}$ and $\mathbf{c}_{i-3}^T \begin{pmatrix}1\\0\end{pmatrix} = \beta_{i-3}$ for every solution mod 2^i.

But if there is some $j < i-1$ such that $\alpha_j = 1$, then $\mathbf{c}_j^T \begin{pmatrix}1\\0\end{pmatrix} = \beta_j$ for every solution X mod 2^{j+1}, thus also for every solution mod 2^i. And if $\beta_j = \beta_{j+j} = \cdots \beta_{i-2}$, then every solution mod 2^i aslo satisfies $\mathbf{c}_{i-2}^T \begin{pmatrix}1\\0\end{pmatrix} = \beta_{i-2}$ and since $\beta_{i-2} \neq \beta_{i-1}$, no solution mod 2^i satisfies $\mathbf{c}_{i-1}^T \begin{pmatrix}1\\0\end{pmatrix} = \beta_{i-1}$, and therefore is not a solution mod 2^{i+1}.

Since $c_0 = 0$ for every X, we have also $\mathbf{c}_0^T \begin{pmatrix}1\\0\end{pmatrix} = \beta_0$ if $\beta_0 = 0$. And if $\beta_0 = \beta_1 = \cdots = \beta_{i-2} \neq \beta_{i-1}$ then again no solution mod 2^i is a solution mod 2^{i+1}.

Proposition 1.19. *The equation (8)*

$$(\mathbf{x} \oplus \boldsymbol{\alpha}) + \boldsymbol{\beta} = (\mathbf{x} + \boldsymbol{\beta}) \oplus \boldsymbol{\alpha}$$

is unsolvable if and only if one of the following two conditions holds

i. there exists indices $j < i < p-1$, such that $\alpha_{i-1} = \alpha_j = 1$ and $\beta_j = \beta_{j+1} = \cdots = \beta_{i-1} \neq \beta_i$,

ii. there exists an index $i < p-1$, such that $\alpha_i = 1$ and $0 = \beta_0 = \beta_1 = \cdots = \beta_{i-1} \neq \beta_i$.

2 Multi-block Collisions in Hash functions based on 3C and 3C+ Enhancements of the Merkle-Damgård

2.1 Introduction

Research in the design and analysis of cryptographic hash functions has been very active since Wang at al [27] published their first collision search algorithm for the MD5 hash function. Collision search algorithms for other hash functions have been discovered, in particular for SHA-0, see [1], [30]. An algorithm for finding collisions in SHA-1 that is significantly more efficient than the generic birthday attack was announced in [28].

In the light of these attacks Gauravaram et al [5] have proposed a slight modification to the Merkle-Damgård construction for an improved protection against many known attacks on MD based hash functions. Their idea is to add additional registers that would collect xors of all chaining variables. After the message is processed the content of additional registers is padded to provide one more message block and the extra block is used as an input for the last calculation of the compression function. Thus the original MD construction remains and the extra security is supposed to be provided by the additional registers, see Figure 1.

Since the 3C construction contains the original MD contruction, any n-block internal collision for the 3C construction must be in fact an n-block collision of the MD construction based on the same compression function. However, because of the extra use of the compression function at the end of the 3C construction one cannot claim that an n-block collision for the 3C construction must be also an n-block collision of the MD construction. To find an n-block collision (where $n \geq 2$) for the 3C construction that is not an n-block collision for the MD construction based on the same compression function would require to find a collision in the compression function with different IV's and possibly different input blocks.

In this paper we show that the 3C construction does not increase significantly resistance against multi-block collisions. In fact, under very mild assumptions we prove that if there is an algorithm that finds n-block collisions for the MD construction based on a compression function, then one can easily find $(2n)$-block collisions for the 3C construction based on the same compression function and $(2n+1)$-block collisions for its modification called 3C+. Our theorem can be applied in particular to the MD5 compression function.

We also observe that the 2-block collisions for the SHA-0 hash function published in [30] are in fact also 2-block collisions for the 3C construction based on the same collision function.

The paper is organized as follows. In section 2 we discuss the 3C and 3C+ design principles, in section 3 we point out a few important properties of the recent 2-block collision attacks on MD5. In Section 4 we prove two simple general theorems how multi-collision attacks on the MD construction can be extended to multi-collision attacks on the 3C and 3C+ constructions. We conclude the paper in section 5. In the appendix we present concrete examples of colliding messages for the 3C and 3C+ constructions based on the compression function of MD5.

2.2 Description of 3C and 3C+

The 3C construction is a generic construction designed as an enhancement to the Merkle-Damgård construction with the idea to increase its resistance against multi-block collision attacks. One of the main properties of the 3C construction is that it is as efficient as the standard hash functions when it is instantiated with the compression functions of any of these hash functions.

The 3C construction accumulates every chaining state of the MD construction by xoring it to the register already containing xor of all previous chaining states.

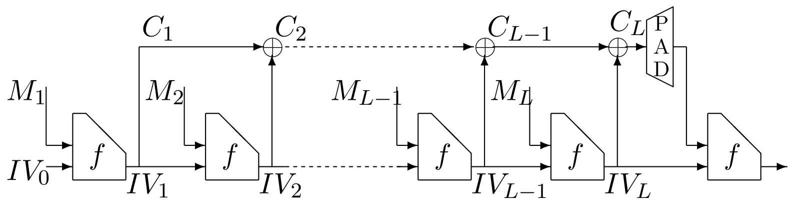

Figure 1: 3C construction of hash function

Thus if IV_i is the chaining variable obtained as the result of i-th iteration of the compression function (IV_0 is the initialization vector), then the value of the additional accumulation registers (denoted by C_i) after the i-th iteration of the compression function is $C_1 = \mathrm{IV}_1$ and

$$C_i = C_{i-1} \oplus \mathrm{IV}_i = \mathrm{IV}_1 \oplus \mathrm{IV}_2 \oplus \cdots \oplus \mathrm{IV}_i$$

for $i = 2, \ldots, L$, where L is the number of blocks of the message. The authors also suggest in their paper [5] that different variants can be obtained for 3C by replacing the xor function in the accumulation chain by other non-linear functions.

The 3C+ construction is a different modification of the 3C construction in which yet another chain D_i, $i = 1, \ldots, L$ of additional registers accumulating the values of chaining variables is added. This time $D_1 = \mathrm{IV}_0$ and

$$D_i = D_{i-1} \oplus \mathrm{IV}_i = \mathrm{IV}_0 \oplus \mathrm{IV}_2 \oplus \cdots \oplus \mathrm{IV}_i$$

for $i = 2, \ldots, L$. Thus

$$D_i = C_i \oplus \mathrm{IV}_1 \oplus \mathrm{IV}_0$$

for every $i = 2, \ldots, L$.

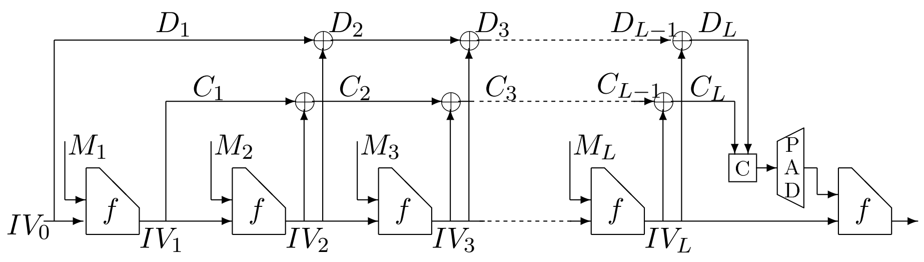

Figure 2: 3C+ construction of hash function

2.3 Multi-block collision attacks

The hash function MD5 uses four 32-bit registers to keep the value of each chaining variable IV_i. We denote them by $\mathrm{IV}_{i,0}, \mathrm{IV}_{i,1}, \mathrm{IV}_{i,2}, \mathrm{IV}_{i,3}$. Thus

$$\mathrm{IV}_i = (\mathrm{IV}_{i,0}||\mathrm{IV}_{i,1}||\mathrm{IV}_{i,2}||\mathrm{IV}_{i,3}).$$

Wang et al presented in [27] an algorithm for finding 2-block collisions in MD5. Their algorithm works for an arbitrary initialization vector IV_0. If $(M_1||M_2)$ and $(M_1'||M_2')$ are two colliding messages found by their algorithm then the modular differences of the chaining variables after processing the first blocks M_1 and M_1' are

$$\begin{aligned}
\Delta_{i,0} = \mathrm{IV}_{1,0}' - \mathrm{IV}_{1,0} &= 2^{31} \\
\Delta_{i,1} = \mathrm{IV}_{1,1}' - \mathrm{IV}_{1,1} &= 2^{31} + 2^{25} \\
\Delta_{i,2} = \mathrm{IV}_{1,2}' - \mathrm{IV}_{1,2} &= 2^{31} + 2^{25} \\
\Delta_{i,3} = \mathrm{IV}_{1,3}' - \mathrm{IV}_{1,3} &= 2^{31} + 2^{25}, \qquad (1)
\end{aligned}$$

where

$$\begin{aligned}
\mathrm{IV}_1 &= f(\mathrm{IV}_0, M_1) \\
\mathrm{IV}_1' &= f(\mathrm{IV}_0, M_1')
\end{aligned}$$

and f is the compression function used in MD5.

Wang et al in [27] also presented a set of so-called sufficient conditions for registers in computation of $f(\mathrm{IV}, M_1)$ to produce the first blocks of a pair of colliding messages. These conditions in fact were not sufficient and various authors, e.g. [13] [32] offered their sets of sufficient conditions. For our purposes only the conditions for IV_1 are important and these conditions were the same for all authors. In fact, we need only four of the sufficient conditions for IV_1 and these four conditions are described in the Table 1.

The exact value of $\mathrm{IV}_1 \oplus \mathrm{IV}_1'$ then follows from given modular differences (1) and prescribed conditions for IV_1 in the Table 1. Thus $\mathrm{IV}_1 \oplus \mathrm{IV}_1'$ is

$IV_{1,0}$	....	....	....	....
$IV_{1,1}$	......0.	....	....	....
$IV_{1,2}$	.....01.	....	....	....
$IV_{1,3}$	......0.	....	....	....

Table 1: Prescribed conditions for IV_1

a constant independent of the initialization vector IV_0 and the first blocks M_1 and M_1' of the colliding messages $(M_1||M_2)$ and $(M_1'||M_2')$.

$\delta_{1,0} = \mathrm{IV}_{1,0} \oplus \mathrm{IV}'_{1,0}$	10000000	00000000	00000000	00000000
$\delta_{1,1} = \mathrm{IV}_{1,1} \oplus \mathrm{IV}'_{1,1}$	10000010	00000000	00000000	00000000
$\delta_{1,2} = \mathrm{IV}_{1,2} \oplus \mathrm{IV}'_{1,2}$	10000110	00000000	00000000	00000000
$\delta_{1,3} = \mathrm{IV}_{1,3} \oplus \mathrm{IV}'_{1,3}$	10000010	00000000	00000000	00000000

Table 2: Prescribed δ for IV_1

The collision finding algorithm for SHA-0 by Wang et al [30] also finds 2-block colliding messages but the structure of the messages in this attack is different than in the case of MD5. The first blocks of the colliding messages $(M_1||M_2)$ and $(M_1||M_2')$ are the same and serve to obtain the chaining variable IV_1 satisfying the conditions sufficient for finding the second blocks M_2 and M_2'. The algorithm again works for an arbitrary IV_0.

In another paper [28] Wang et al propose an algorithm for finding 2-block collisions in SHA-1 that is faster than the generic birthday attack. Although no real collisions in SHA-1 have been found so far, the form of colliding messages of the proposed attack is in fact the same as in the case of MD5. It means that the algorithm should work for any IV_0 and $\mathrm{IV}_1 \oplus \mathrm{IV}_1'$ should be a constant independent of IV and M_1 and M_1'.

2.4 Multi-block collision attacks on 3C and 3C+

The idea of the attack on the 3C construction when the compression function is the same as in MD5 is very simple. First, we find 2-block colliding messages $(M_1||M_2)$ and $(M_1'||M_2')$ in MD5 using the attack by Wang et al [27]. Then we take the chaining variable $\mathrm{IV}_2 = \mathrm{IV}_2'$ as the initialization vector for the second run of the Wang et al algorithm. In this way we obtain another pair of messages $(M_3||M_4)$ and $(M_3'||M_4')$. The 4-block messages $(M_1||M_2||M_3||M_4)$ and $(M_1'||M_2'||M_3'||M_4')$ then form a collision for the 3C construction based on the MD5 compression function. The scheme of the attack and the distribution of differences are shown on Figure 3.

A formal verification of the idea is contained in the following theorem.

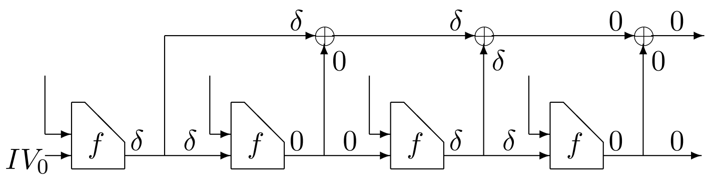

Figure 3: 4-block internal collision attack on 3C without the final processing

Theorem 2.1. *Let H be an MD hash function based on a compression function f. Suppose that for some $n \geq 2$ there exists an algorithm finding n-block collisions for H that works for any initialization vector* IV_0 *and has the property that* $\mathrm{IV}_i \oplus \mathrm{IV}'_i$ *for $i = 1, \ldots, n$ is a constant independent of* IV_0 *and the actual colliding messages (but can be dependent on i). Then there exists an algorithm that finds $(2n)$-block collisions for the 3C construction based on the same compression function f.*

The running time of the algorithm for finding collisions in the 3C construction is twice the running time of the algorithm for finding collisions in the MD construction using the same compression function.

Proof. Let $(M_1||M_2||\cdots||M_n)$ and $(M'_1||M'_2||\cdots||M'_n)$ be the colliding messages obtained by the first run of the algorithm finding collisions in H. Thus $\mathrm{IV}_n = \mathrm{IV}'_n$. We use this value as the initialization vector for the second run of the collision search algorithm for H. We obtain another pair of colliding messages $(M_{n+1}||M_{n+2}||\cdots||M_{n+n})$ and $(M'_{n+1}||M'_{n+2}||\cdots||M'_{n+n})$. We denote the chaining variables in the second run of the algorithm by IV_{n+i} and IV'_{n+i} for $i = 1, \ldots, n$.

By our assumption on the collision search algorithm for H we can write

$$\mathrm{IV}_i \oplus \mathrm{IV}'_i = \mathrm{IV}_{n+i} \oplus \mathrm{IV}'_{n+i}$$

for every $i = 1, \ldots, n$. Thus we obtain

$$C_{2n} = \bigoplus_{i=1}^{2n} \mathrm{IV}_i$$

and

$$C'_{2n} = \bigoplus_{i=1}^{2n} \mathrm{IV}'_i.$$

Hence

$$\begin{aligned} C_{2n} \oplus C'_{2n} &= \bigoplus_{i=1}^{2n} \mathrm{IV}_i \oplus \bigoplus_{i=1}^{2n} \mathrm{IV}'_i = \bigoplus_{i=1}^{2n} (\mathrm{IV}_i \oplus \mathrm{IV}'_i) \\ &= \bigoplus_{i=1}^{n} (\mathrm{IV}_i \oplus \mathrm{IV}'_i) \oplus (\mathrm{IV}_{n+i} \oplus \mathrm{IV}'_{n+i}) \\ &= 0. \end{aligned}$$

Since $\mathrm{IV}_{2n} = \mathrm{IV}'_{2n}$, the messages $(M_1||\cdots||M_n||M_{n+1}||\cdots||M_{2n})$ and $(M'_1||\cdots||M'_n||M'_{n+1}||\cdots||M'_{2n})$ form a collision for the 3C construction based on f. $\square$

In Section 2.3 we explained that the Wang et al [27] collision search algorithm for MD5 satisfied the assumptions of Theorem 2.1 for $n = 2$. Thus there exists an algorithm finding 4-block collisions in the 3C construction based on the MD5 compression function. The fastest implementation of the Wang et al algorithm known in the moment of writing the paper is by Klima [10] and finds collisions in MD5 in about 30 seconds in average. Thus at this moment collisions in the 3C construction based on the MD5 compression function can be found within a minute.

As for the 3C construction based on the SHA-0 compression function there is no need for running the algorithm twice to obtain a collision. Since the collision search algorithm for SHA-0 finds colliding messages of the form $(M_1||M_2)$ and $(M_1||M'_2)$, we get $\mathrm{IV}_1 = \mathrm{IV}'_1$ and $\mathrm{IV}_2 = \mathrm{IV}'_2$, thus $C_2 = C'_2$. Hence the SHA-0 collisions found by the algorithm are simultaneously collisions for the 3C construction based on the SHA-0 compression function.

Since the theoretical algorithm for finding collisions in SHA-1 proposed by Wang et al in [28] also satisfies the assumption of Theorem 2.1 running the algorithm twice should again produce a 4-block collision in the 3C construction based on the SHA-1 compression function.

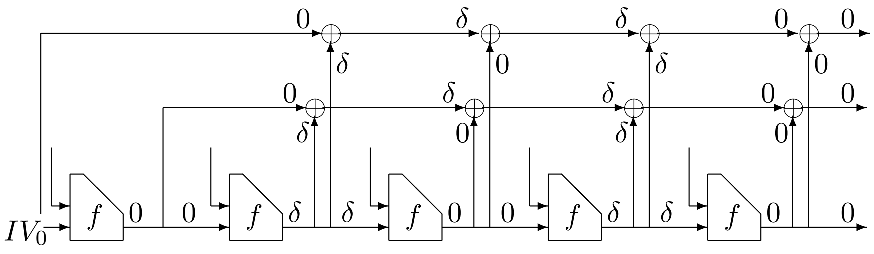

Figure 4: 5-block collision attack on 3C+ without the final processing

The Figure 4 shows how a 5-block collision for the 3C+ construction based on the $MD5$ compression function can be found. The only difference with the collision search algorithm for the 3C construction is that we start with an arbitrary message block M_1, calculate the value of the compression function for the block with given IV_0 to obtain a new initialization vector IV_1 and then we run the collision search algorithm for the 3C construction with the initialization vector IV_1. We obtain messages $(M_1||M_2||M_3||M_4||M_5)$ and $(M_1||M'_2||M'_3||M'_4||M'_5)$ such that $C_5 = C'_5$ and $\mathrm{IV}_5 = \mathrm{IV}'_5$. Since $D_5 = C_5 \oplus \mathrm{IV}_1 \oplus \mathrm{IV}_0$ and $D'_5 = C'_5 \oplus \mathrm{IV}_1 \oplus \mathrm{IV}_0$, we obtain also $D_5 = D'_5$.

From this observation one obtains the following theorem.

Theorem 2.2. *Suppose there exists an algorithm finding k-block collisions in the 3C construction based on a compression function f. Then there exists an algorithm for finding $(k+1)$-block collisions in the 3C+ construction based on the same compression function f.*

The running time of the algorithm for the 3C+ construction is equal the running time of the algorithm for the 3C+ construction plus the running time of the one calculation of the compression function.

Proof. Follows from the previous observation. □

2.5 Conclusion

3C and 3C+ constructions based on a compression function f which is not collision resistance is a bad idea. We have shown how to find collisions for 3C and 3C+ constructions based on a compression function f provided a collision search algorithm for the MD construction based on f is known. We also presented real examples of collisions for the 3C and 3C+ constructions based on the MD5 compression function in [8].

2.6 Appendix: Examples of Collisions

IV	0x67452301	0x10325476	0x98badcfe	0xefcdab89
M_1	0x4e1a8245	0x5fe0e55d	0xfe3faa53	0x0d8546b3
	0x18ccad34	0xac0bae59	0xd59d3352	0x4805693e
	0x06342cd5	0x81b41206	0x83c2bea3	0x8fd22557
	0xc41a4cd6	0x9e9a4fe1	0x818ae34d	0x1a97e731
N_1	0x4e1a8245	0x5fe0e55d	0xfe3faa53	0x0d8546b3
	0x98ccad34	0xac0bae59	0xd59d3352	0x4805693e
	0x06342cd5	0x81b41206	0x83c2bea3	0x8fd2a557
	0xc41a4cd6	0x9e9a4fe1	0x18ae34d	0x1a97e731
IV_1	0xadebbbec	0xc85d058e	0xa2672e58	0xb91d144b
IV_1'	0x2debbbec	0x4a5d058e	0x24672e58	0x3b1d144b
$\mathrm{IV}_1 \oplus \mathrm{IV}_1'$	0x80000000	0x82000000	0x86000000	0x82000000
M_2	0x06faa233	0x1c84a4bf	0xf38ee5f1	0x08deb9af
	0x467ad36b	0x4c900712	0xd6a37d26	0x11f6de56
	0x8577e045	0x299991d5	0x5940588e	0x3fd25887
	0x301fc8fa	0x77dc0e81	0xe8c1a1a7	0x13d51d82
N_2	0x06faa233	0x1c84a4bf	0xf38ee5f1	0x08deb9af
	0xc67ad36b	0x4c900712	0xd6a37d26	0x11f6de56
	0x8577e045	0x299991d5	0x5940588e	0x3fd1d887
	0x301fc8fa	0x77dc0e81	0x68c1a1a7	0x13d51d82
$\mathrm{IV}_2 = \mathrm{IV}_2'$	0xa918ce8d	0xb7ea0df6	0x69bdb806	0x713af4de
M_3	0xcd71fe0c	0x58d0f463	0xa9399e1d	0x7db79e98
	0x3622a432	0x736cb277	0x011cb460	0x6a04e9b4
	0x06332d55	0x23f47e02	0x799ab597	0xd3ba5325
	0xb9e866e6	0xde6b9cd3	0xde6cebbb	0x0b4c3783
N_3	0xcd71fe0c	0x58d0f463	0xa9399e1d	0x7db79e98
	0xb622a432	0x736cb277	0x011cb460	0x6a04e9b4
	0x06332d55	0x23f47e02	0x799ab597	0xd3bad325
	0xb9e866e6	0xde6b9cd3	0x5e6cebbb	0x0b4c3783
IV_3	0x2b30549a	0x089c590a	0x52710661	0x6932f794
IV_3'	0xab30549a	0x8a9c590a	0xd4710661	0xeb32f794
$\mathrm{IV}_3 \oplus \mathrm{IV}_3'$	0x80000000	0x82000000	0x86000000	0x82000000
M_4	0x96ded638	0x4c1be33a	0xd46e6a5f	0xdbc8da73
	0x473af92b	0x4d0da98e	0x56dd6d3e	0xd19e7bd1
	0x53f857cd	0x4c25f191	0x918be4da	0xc09e206c
	0x320b28d4	0xcc6c0e7a	0x68515c76	0x57840834
N_4	0x96ded638	0x4c1be33a	0xd46e6a5f	0xdbc8da73
	0xc73af92b	0x4d0da98e	0x56dd6d3e	0xd19e7bd1
	0x53f857cd	0x4c25f191	0x918be4da	0xc09da06c
	0x320b28d4	0xcc6c0e7a	0xe8515c76	0x57840834
$\mathrm{IV}_4 = \mathrm{IV}_4'$	0x6a1a021a	0xc81fe980	0x88e1db5b	0x512e7c88

Table 3: Collision in 3C invoked with MD5 compression function

IV	0x67452301	0x10325476	0x98badcfe	0xefcdab89
M_1	0x0634add5	0x4074c002	0x7baaf717	0x0f522d75
	0xbf6ac0ec	0xa4885903	0x7349e78b	0x2aad1b45
	0x281dfb7e	0x173e6c0c	0xab79fc54	0x39453670
	0x44fb372b	0x4d5259c8	0xf7ad2d48	0xd1254b51
N_1	0x0634add5	0x4074c002	0x7baaf717	0x0f522d75
	0xbf6ac0ec	0xa4885903	0x7349e78b	0x2aad1b45
	0x281dfb7e	0x173e6c0c	0xab79fc54	0x39453670
	0x44fb372b	0x4d5259c8	0xf7ad2d48	0xd1254b51
$\mathrm{IV}_1 = \mathrm{IV}'_1$	0xd3f4b63c	0x595f4645	0xa890d3d0	0x9cc907db
M_2	0xa72fc176	0x64b7a050	0xe266ae7a	0x1b21009e
	0xfac1ee4c	0x9e588e8e	0x076d346d	0x805529b7
	0x0633ad55	0x02342602	0x83b4ba0b	0x56d1d924
	0x82d9651a	0xba9c8de6	0xebbbe37e	0xb78c63d5
N_2	0xa72fc176	0x64b7a050	0xe266ae7a	0x1b21009e
	0x7ac1ee4c	0x9e588e8e	0x076d346d	0x805529b7
	0x0633ad55	0x02342602	0x83b4ba0b	0x56d25924
	0x82d9651a	0xba9c8de6	0x6bbbe37e	0xb78c63d5
IV_2	0xead1c69e	0xd19e34c2	0xca2e528e	0xb1790589
IV'_2	0x6ad1c69e	0x539e34c2	0x4c2e528e	0x33790589
$\mathrm{IV}_2 \oplus \mathrm{IV}'_2$	0x80000000	0x82000000	0x86000000	0x82000000
M_3	0x6dbb34a0	0x9c1b815b	0x7ceb8ffd	0x1502296c
	0x467d585b	0x4d0d8038	0xc6db2d16	0x00d11ad5
	0xd2b2eeed	0x4a04145b	0x2f79d4aa	0x00be08a0
	0xf2e830f3	0x10bc0a85	0xe9019cb8	0x4fd512a2
N_3	0x6dbb34a0	0x9c1b815b	0x7ceb8ffd	0x1502296c
	0xc67d585b	0x4d0d8038	0xc6db2d16	0x00d11ad5
	0xd2b2eeed	0x4a04145b	0x2f79d4aa	0x00bd88a0
	0xf2e830f3	0x10bc0a85	0x69019cb8	0x4fd512a2
$\mathrm{IV}_3 = \mathrm{IV}'_3$	0x46321911	0x9d317bd2	0xfde6d50e	0xeb2170d8
M_4	0x122cdc12	0x5f60de22	0xedac78fd	0xf506f854
	0x2b85436b	0x3c980908	0xda4c144d	0x03344bbe
	0x0634ad55	0x0113f402	0x80aab777	0x13888f67
	0xadea26f7	0x623cc142	0x1192759e	0x0e74317c
N_4	0x122cdc12	0x5f60de22	0xedac78fd	0xf506f854
	0xab85436b	0x3c980908	0xda4c144d	0x03344bbe
	0x0634ad55	0x0113f402	0x80aab777	0x13890f67
	0xadea26f7	0x623cc142	0x9192759e	0x0e74317c
IV_4	0x754b85c2	0x45386ef2	0x3adad7b7	0x61523316
IV'_4	0xf54b85c2	0xc7386ef2	0xbcdad7b7	0xe3523316
$\mathrm{IV}_4 \oplus \mathrm{IV}'_4$	0x80000000	0x82000000	0x86000000	0x82000000
M_5	0x65171431	0x2615affc	0x2a2519e7	0xe2e99ce8
	0x44bcf42b	0x4c4def0e	0x47aadd22	0x127d7d56
	0x62bf776d	0x6cc9d58b	0x597058d6	0x602a5867
	0x3e2bc8ce	0xb3ec1267	0x68716155	0x17a50429
N_5	0x65171431	0x2615affc	0x2a2519e7	0xe2e99ce8
	0xc4bcf42b	0x4c4def0e	0x47aadd22	0x127d7d56
	0x62bf776d	0x6cc9d58b	0x597058d6	0x6029d867
	0x3e2bc8ce	0xb3ec1267	0xe8716155	0x17a50429
$\mathrm{IV}_5 = \mathrm{IV}'_5$	0x1453b7b0	0x803e8aee	0xfd85765e	0x176ca5d9

Table 4: Collision in 3C+ invoked with MD5 compression function

3 Beyond the MD5 Collisions

3.1 Introduction

A little more than two years ago collision resistance of several widely used hash functions was broken. A group of researchers led by Xiaoyun Wang (of Shandong University, China) presented at rump session of Crypto 2004 collision resistance attacks on MD5 and other hash functions. Since then a lot of papers on various aspects of the Wang et al. attacks were published.

To explain what the consequences of [29] are, we start with several requirements a well designed hash function should satisfy. First of all, the hash function should be one-way meaning that it is very difficult to invert it, i.e. to find any input with the prescribed hash value. Further requirements we describe on the application in digital signature scheme. Every digital signature scheme is based on an asymmetric cipher. The digital signature of a document is obtained by applying the decryption function of the cipher (or signature function) to the document. As asymmetric ciphers are very slow, the decryption function is applied only to the hash value of the document rather than to the document itself. This restricts the length of the message to which the decryption function is applied to the length of the hash function used in the digital signature scheme. That means we can speedup the calculation of the signature if the underlying hash function is very fast even for extremely long input messages.

However, replacing the document by its hash value in digital signature schemes is not without danger. A digital signature of a message is simultaneously a digital signature of any other message with the same hash value as the original one. Thus it is critical that the hash functions used in digital signature schemes are collision resistant, meaning that it is computationally infeasible to find two different messages with the same hash value. Only in this case it is possible to assign uniquely a digital signature to the signed document. Wang et al in [29] showed that collision resistance of some hash function can be broken and it is possible to create two different documents with the same hash value. Therefore it is dangerous to use these hash functions for digital signatures after announcing their method.

Now if a hash function is found not to be collision resistant then it does not mean that all digital signatures which used the hash function earlier can now be repudiated. To falsify a signature of a given document requires finding a second preimage of the hash value of the document i.e. to find another message with the same hash value as the original document. This is much stronger requirement than just to find any two different messages with the same hash value and e.g. for MD5 it is not possible. However it is believed that finding collisions is the first step to finding second preimage.

In Section 3.2 and 3.3 we present various improvements of Wang et al. algorithm for finding collisions in MD5. Within a year after announcing the first collisions, the searching algorithms were much improved. Finding the first collisions took approximately an hour for very strong computer, after a year the collisions could be found in only seconds on a common PC [10]. We shortly describe these algorithms. We also present a calculation of the computational complexity of the three algorithms. Further in Section 3.4 and 3.5 we discuss suggested solutions for improving the hash functions [5, 19]. The improvements concentrate on small changes in widely used Merkle-Damgård (MD) construction and they were constructed to protect against multi-block collisions. We show that it may be dangerous to use these improved constructions because they do not deal with the main problem - weakness in the recent compression functions. In fact it may not be difficult to find collisions in such improved versions using the same method and the same algorithms than in case of MD construction. We conclude the paper in Section 3.6.

3.2 Collisions in MD5

Description of Wang et al. collision searching algorithm consisted of a detailed set of conditions, so-called differential path, that gives many conditions for the content of registers during the calculations of the MD5 hash value of a message $m = (M^0||M^1)$ consisting of two blocks, each of length of 512 bits. Although Wang et al claimed that their conditions on the content of registers were sufficient for finding another message $n = (N^0||N^1)$ of the same length as m and with the same hash value as n it was shown by various authors e.g. [13, 23] that Wang's sufficient conditions in fact were not sufficient and further conditions were needed. The specification of the MD5 uses fixed initial vector IV but the attack of Wang et al works for any initial vector. (That will be important for attacks in Sections 3.4 and 3.5.) To provide two diistinct messages $m = (M^0||M^1)$ and $n = (N^0||N^1)$ with the same hash value it is necessary and sufficient to prove that

$f(f(M^0, \mathrm{IV}), M^1) = f(f(N^0, \mathrm{IV}), N^1)$, where f is the compression function in MD5. The collisions in [29] as well as all other colliding pairs found by various authors since the first Wang's announcement have the following properties (so-called differentials)

$$\Delta_0 := M^0 - N^0 = (0, 0, 0, 0, 2^{31}, 0, 0, 0, 0, 0, 0, +2^{15}, 0, 0, 2^{31}, 0) \qquad (1)$$

$$\Delta_1 := M^1 - N^1 = (0, 0, 0, 0, 2^{31}, 0, 0, 0, 0, 0, 0, -2^{15}, 0, 0, 2^{31}, 0) \qquad (2)$$

$$\delta := f(\mathrm{IV}, M^0) - f(\mathrm{IV}, N^0) = (2^{31}, 2^{31} + 2^{25}, 2^{31} + 2^{25}, 2^{31} + 2^{25}). \qquad (3)$$

Wang et al. proceed with giving precise forms for modular differences $(R_i - R'_i)$ and xor differences $(R_i \oplus R'_i)$, for $i = 0, \ldots, 63$, where R_i and R'_i are registers obtained during the calculation $f(\mathrm{IV}, M^0)$ and $f(\mathrm{IV}, N^0)$, as well as precise forms of $\mathrm{IV}_1 \oplus \mathrm{IV}'_1$ where $\mathrm{IV}_1 := f(\mathrm{IV}, M^0)$ and $\mathrm{IV}'_1 := f(\mathrm{IV}, N^0)$ and as well as forms of modular and xor differences for registers R_i, R'_i obtained during the calculation of $f(\mathrm{IV}_1, M^1)$ and $f(\mathrm{IV}'_1, N^1)$. By registers R_i

obtained during the calculation of compression function f we mean the chaining variables aa, dd, cc, bb used in the specification of the MD5 in RFC1321 indexed from 0 to 63. This is what they call a differential path for MD5 collision. From the differential path they derived a set of conditions for particular bits in the registers $R_0, \ldots, R_{63}$, obtained when processing the first block M^0, for particular bits in IV_1 and for particular bits in the registers $R_0, \ldots, R_{63}$, obtained when processing the second block M^1 (i.e. when calculating $f(IV_1, M^1)$. They claimed these conditions were sufficient for a message $m = (M^0||M^1)$ satisfying the conditions and another message $n = (M^0 + \Delta_0||M^1 + \Delta_1)$ to have the same hash value. As we have already mentioned these conditions were not sufficient and small corrections were needed [13, 23]. However the goal was to find message $m = (M^0||M^1)$ which has the proper registers R_i'-s i.e. the conditions for these registers are satisfied.

The general method of the attack is to compute messages $M_i, 0 \leq i \leq 15$, (we use lower index for indexing 32 bit long registers in a message block) from the first 16 registers R_i, where $R_i, 0 \leq i \leq 15$, satisfies the prescribed conditions. Then we can continue in computing of registers R_i, $16 \leq i \leq 63$, and verify if the prescribed conditions are satisfied. If we find a condition that is not satisfied we stop computing and change some bits in R_i, for some $0 \leq i \leq 15$. Then we calculate a new message words M_i for indexes i affected by the change in registers R_i. We continue in computing and verifying the conditions for $R_i, i \geq 16$ and if all prescribed conditions are satisfied we have found the first (or the second) block of the colliding message m. The other colliding message n can be computed as $n = (M^0 + \Delta_0||M^1 + \Delta_1)$. The problem is that there are 45 conditions for registers R_i, $i \geq 16$, in the first message block M^0 and the probability that all of them are satisfied is estimated as 2^{-45}. Of course this is much better than 2^{-64} from the birthday paradox attack but is there any chance for an improvement?

3.3 Recent collision searching algorithms

The first collision searching algorithm we describe is by Vlastimil Klíma [10], the second one is by Marc Stevens [23] and the third one is our algorithm [6]. All three algorithms were developed independently in the beginning of 2006 and all three are using Wang's method described in [29].

The main goal of these improvements is an effort to force message M^0 to have all the sufficient conditions on registers R_i satisfied. As we said, the problem is to force conditions for registers $R_i, i \geq 16$. Details of the three presented algorithms can be found in the original papers [10, 23, 6]. Here we try to generalize the methods they are using so-called multi-message modification and tunneling. We also present a pseudocode of the three algorithms for finding the first message block M^0 at the end of this section. The algorithms for finding the second block M^1 are very similar.

By the *verification process* we call the process of calculating the values of registers starting from R_{16} and checking the prescribed conditions for $R_i, i \geq 16$. If some condition is not verified then we start changing some bits in R_i for some $0 \leq i \leq 15$. Then we compute the affected message words M_i. By the *generating of a candidate for collision* we mean the process of changing bits in registers and computing affected message words M_i, where by *candidate for collision* we mean the message block that can be calculated from the modified registers. We want to note that the computation of affected message words M_i does not have to be done before the procedure of verification is started. In fact, the changes in R_i does not have to affect the M_i needed in calculation of R_{16} (or the first step in the verification procedure). We can take advantage of that fact and calculate the affected message words only after some of the first conditions are checked. It will make the algorithm faster.

The *multi-message modification* method concentrates on preparing the message registers $M_i, 0 \leq i \leq 15$, in a way that sufficient conditions for some registers $R_i, i \geq 16$, will be satisfied with greater probability than 1/2. By setting of bits in message registers $M_i, 0 \leq i \leq 15$, we can influent bits in some registers after R_{15}, i.e. $R_{16}, R_{17}, R_{18}, R_{19}$. The problem is that further we are from R_{15} the harder it is to influent the bits in these registers and so the method is effective only for satisfying several conditions on registers $R_{16}, \ldots, R_{20?}$ [6, 29].

The *tunneling* method [10] came up with a different approach. Suppose we can find a collision candidate satisfying the conditions up to register R_{23}. Can we change some bits in registers M_i , $0 \leq i \leq 15$, then recalculate registers R_i which are affected by the change and the sufficient conditions up to R_{23} will stay satisfied? The places where this is possible are called tunnels. There are different types of tunnels. According to the numbers of changes that can be done in the tunnel the *strength* of the tunnel is defined. The tunnel of strength n can create 2^n of different collision candidates satisfying the sufficient conditions up to R_{23}. If a tunnel can satisfy the sufficient conditions up to R_{23} with some probability it is called *probabilistic* tunnel. The characteristics of tunnels described in [10] are summarized in the Table 1. We added the parameters cost which measure the expected number of recalculated registers R_i and M_i for creation of one new collision candidate satisfying the sufficient conditions up to R_{23}.

We summarize the information about the tunnels in the Table 1. We must note that the numbers in this table are very roughly estimated and further analysis would be required. However, we will show the idea of complexity calculation for the algorithm in the next section.

3.3.1 Algorithm Complexity

Basic unit in which we measure the complexity is computation of one MD5 step i.e. computation of one register R_i in compression function. We asked

tunnel	strength	probability	cost	changed registers
T_6 : Q9	3	1	3	M_8, M_9, M_{12}
T_5 : Q4	1	1	3	M_3, M_4, M_7
T_4 : Q14	8	1	8	$M_2, M_3, M_4, R_{23}, M_6, M_7, M_{13}, M_{14}$
T_3 : Q10	2	1/2	4*2 + 3	$M_{10}, R_{21}, R_{22}, R_{23}, M_9, M_{12}, M_{13}$
T_2 : Q13	> 10	1/3	3*7 + 3	$M_5, R_{20}, R_{21}, M_{15}, R_{22}, M_4, R_{23}, M_1, M_2, M_3$
T_1 : Q20	> 3	roughly1/7	7*6 + 3	$M_5, R_{20}, R_{21}, R_{22}, M_4, R_{23}, M_1, M_2, M_3$

Table 1: Tunnels characteristics

Algorithm 1 Klíma's algorithm

```
 1: Loop until the conditions for $R_{16}, \dots, R_{23}$ are fulfilled;
 2: Loop over all possible changes in $T_1$
 3:    Generate a new candidate for collision for price $c_1$;
 4:    Verify the candidate; if all conditions are true then return
       $M^0$, $N^0$, $IV_m^1$, $IV_n^1$.
 5:    Loop over all possible changes in $T_2$
 6:                      $\ddots$
 7:             Loop over all possible changes in $T_6$
 8:                Generate a new candidate for collision for price $c_6$;
 9:                Verify the candidate; if all conditions are true then return
       $M^0$, $N^0$, $IV_m^1$, $IV_n^1$.
10:             End of loop $T_6$;
11:       ...
12: End of loop $T_1$;
```

ourselves what has an influence on the duration of searching algorithms.

(a) The cost of generating one collision candidate. (random variable G)

(b) The cost of verifying the candidate. (random variable V)

(c) The number of collision candidates that need to be generated. (random variable C)

Then the expected value of waiting for collision can be calculated as

$$E(C) \cdot (E(G) + E(V))$$

We present in [6] several prepositions in which we calculate the expected values of the three variables and enumerate the values for the three described algorithms. We obtained following results.

Algorithm 2 Stevens' Block 1 search algorithm

```
1: Choose $Q_1, Q_3, \dots, Q_{16}$ fulfilling conditions;
2: Calculate $m_0, m_6, \dots, m_{15}$;
3: Loop until $Q_{18}, \dots, Q_{21}$ are fulfilling conditions:
     (a) Choose $Q_{17}$ fulfilling conditions;
     (b) Calculate $m_1$ at $t = 16$;
     (c) Calculate $Q_2$ and $m_2, m_3, m_4, m_5$;
     (d) Calculate $Q_{18}, \dots, Q_{21}$;
4: Loop over all possible $Q_9, Q_{10}$ satisfying conditions such that $m_{11}$ does
   not change:
     (a) Calculate $m_{10}$;
     (b) Calculate $Q_{22}, Q_{23}, Q_{24}, \mathbf{m_8}, \mathbf{m_9}, \mathbf{m_{12}}, \mathbf{m_{13}}, Q_{25} \dots, Q_{64}$;
     (c) Verify conditions for $Q_{22}, \dots, Q_{64}, T_{22}, T_{34}$ and the iv-conditions
         for the next block. Stop searching if all conditions are satisfied and
         a near-collision is verified.
5: Start again at step 1.
```

For the first colliding block and the original initial vector the average running time is

i. $24.5010 \cdot (2^{30})$ MD5 steps for Stevens' algorithm,

ii. $7.7153 \cdot (2^{30})$ MD5 steps for Klíma's algorithm,

iii. $29.437 \cdot (2^{30})$ MD5 stepsfor our algorithm.

For comparison running 2^{32} MD5 steps takes on 1.54 GHz processor less than one minute. We also calculated that the average running time for the second colliding block is remarkable lower that the average for the first block. It is mainly due to the lower number of the sufficient conditions for registers $R_i, i \leq 16$, in second block. That is why a lot of authors calculate just with the complexity of the algorithm for finding the first colliding block.

3.4 3C and 3C+: An Attempt to Improve Merkle-Damgård construction

In the light of the multi-block attacks Gauravaram et al [5] proposed a slight modification of the Merkle-Damgård (MD) construction called 3C and 3C+ to improve protection against known attacks on MD based hash functions.

Algorithm 3 Our searching algorithm for the first block

```
Input:   Initial vector $IV^0$
Output:  Pair of messages $M^0, N^0$, $IV^1_m$, $IV^1_n$

1: $R_{-4} \leftarrow IV^0_0$, $R_{-3} \leftarrow IV^0_3$, $R_{-2} \leftarrow IV^0_2$, $R_{-1} \leftarrow IV^0_1$;
2: Initialize $R_2, R_3, \ldots, R_{16}$ with regard to the fixed bits;
3: repeat
     (a) Randomly change free bits in $R_2$, $R_3$, $R_7$, $R_8$, $R_9$, $R_{10}$, $R_{11}$;
     (b) Calculate $M_6$, $M_7$, $M_8$, $M_9$, $M_{10}$, $M_{11}$, $M_{12}$, $R_{17}$, $R_{18}$;
     (c) if one of the conditions for $R_{17}$ and $R_{18}$ is false then
             - Try to satisfy them by specific changes in registers $R_{13}-R_{16}$;
             - Calculate $R_{17}$ and $R_{18}$ again;
     (d) if conditions for $R_{17}$ and $R_{18}$ are true then
             - Calculate $M_{13}$, $M_{14}$, $M_{15}$, $M_1$;
             - if $M_{15}[17] = 0$ then execute step 4;
4: Loop over all possible choices in $R_{19}[6-0]$ and $R_{19}[14-8]$
     (a) Change $R_{19}$;
     (b) Calculate $M_0$, $R_0$, $R_1$, $M_5$, $R_{20}$, $R_{21}$;
     (c) if $R_{20}[31] \neq 0$ or $R_{20}[17] \neq R_{19}[17]$ or $R_{21}[31] \neq 0$ then
             - Change some bits in $R_{19}$ to satisfy the conditions;
             - Calculate $M_0$, $R_0$, $R_1$, $M_5, R_{20}, R_{21}$ again;
     (d) if $R_{20}[31] = 0$ & $R_{20}[17] = R_{19}[17]$ & $R_{21}[31] = 0$ then (Verify the
         candidate)
             - Calculate $R_{22}, M_4, R_{23}, M_2, M_3, R_{24}, R_{25}, \ldots, IV^1$;
             - Verify the conditions for $R_{22}, \ldots, R_{63}, IV^1_m$;
             - if all conditions are true then return $M^0$, $N^0$, $IV^1_m$, $IV^1_n$.
```

Their idea is to add additional registers that would collect xors of all chaining variables. After the message is processed the content of additional registers is padded to provide one more message block and the extra block is used as an input for the last calculation of the compression function. Thus the original MD construction remains and the extra security is supposed to be provided by the additional registers, see Figures 1 and 2. One of the main properties of the 3C construction is that it is as efficient as the standard hash functions when it is instantiated with the compression functions of any of these hash functions. However it was shown in [8] that one must be very careful to use this construction and invoke it with compression function from MD5 or SHA-0 because the problem with the multi-block collisions was not solved.

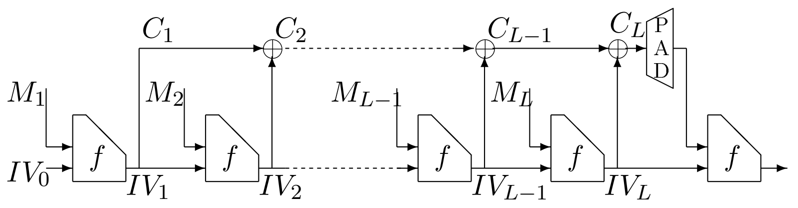

Figure 1: 3C construction of hash function

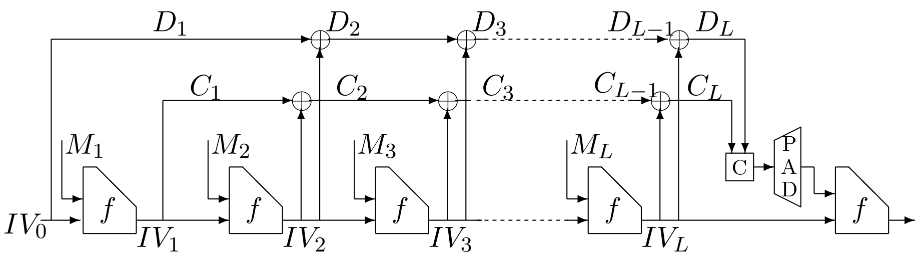

Figure 2: 3C+ construction of hash function

3.4.1 Multi-block collision attacks on 3C and 3C+

We introduce the main idea of the attack at 3C and 3C+ on the following example of 3C construction invoked with compression function taken from MD5. The case of 3C+ construction is very similar. First, 2-block colliding messages $(M^1||M^2)$ and $(N^1||N^2)$ are found using the attack by Wang et al [29]. Any of the algorithms from subsection 3.3 can be applied. It is important the algorithms can find a collision for any IV. Then we take the chaining variable $\mathrm{IV}_2 = \mathrm{IV}'_2$ as the initialization vector for the second run of the Wang et al. algorithm. In this way we obtain another pair of messages $(M^3||M^4)$ and $(N^3||N^4)$. The 4-block messages $(M^1||M^2||M^3||M^4)$ and $(N^1||N^2||N^3||N^4)$ then form the collision for the 3C construction based on the MD5 compression function. The scheme of the attack and the distribution of differences are shown on Figure 3. Basically we took advantage of the fixed differences in equation (3), the fact that $\delta \oplus \delta = 0$ and the property that Wang's collision searching algorithm works for any IV.

We want to note that the method in [8] can be applied to the 3C construction invoked with compression functions also from other hash functions

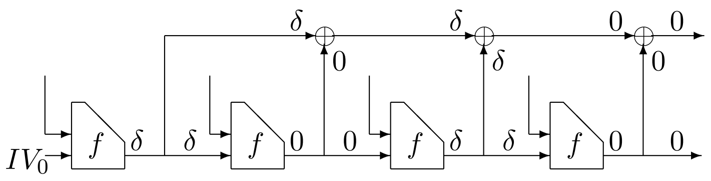

Figure 3: 4-block internal collision attack on 3C without the final processing

such as SHA-0 [30] and SHA-1 [28]. In case SHA-1 the real collisions are not known (yet) but the structure of the collisions in [28] is the same than in case of MD5 what means the collisions in 3C can be found in the same way.

It is also fair to say that the authors of the 3C and 3C+ constructions also suggested substitution of the simple xor function in the additional chaining variable for the more complex one. We agree that it would make searching for collisions more difficult but our point is the weakness is inside of the compression function. Another more complex function will also slow down the algorithm and what is probably the most important we still need some kind of proof of the collision resistance.

3.5 Feedback Ring-iterative Structure

The second example of a new construction of the hash function can be found in [19]. The authors of the paper presented new hash ring-iterative example of which are single feedback ring-iterative structure (SFRI) and multiple feedback ring-iterative structure (MFRI). Description of the constructions is pricesely given in [19]. Here we refer to Figures 3.5 and 3.5 which schematically describe these constructions. Symbol $\rightleftarrows$ means rearrangement of bits of a variable in reverse order and we recall that by symbol $\oplus$ is meant operation xor (bitwise exclusive or) and by symbol $\boxplus$ modular addition. From the description in [19] it is not clear if authors mean k addition modulo 2^{32} or one addition modulo $2^{k(32)}$, where $k = 4$ in case of MD5 or $k = 5$ in case of SHA-1 etc. Notice that message block M^1 forms the input to the compression function twice and message block M^n is processed twice in case of SFRI. For proper description see the original paper [19].

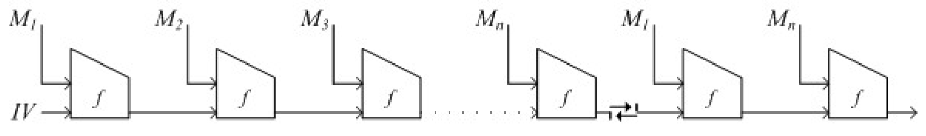

Figure 4: Single Feedback Ring-iterative Structure

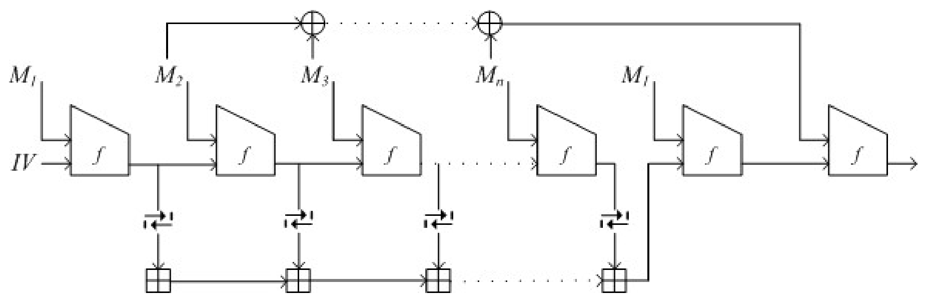

Figure 5: Multiple Feedback Ring-iterative Structure

Almost immediately we see how to use Wang's multi-collisions for colliding messages in SFRI. We start and finish with the same blocks for the both colliding messages and put a colliding pair between these two blocks. So a collision can consist of messages $(M^1||M^2||M^3||M^4)$ and $(M^1||N^2||N^3||M^4)$, where $(M^2||M^3)$ and $(N^2||N^3)$ are different 2-block colliding messages for initial vector $f(IV, M^1)$. The messages $(M^2||M^3)$ and $(N^2||N^3)$ can be obtained from any of the algorithms described in subsection 3.3.

The MFSR seems to be more complex, but in fact it is very similar idea than 3C or 3C+ construction [5] with more complex chaining function. Here we describe just the idea of possible multi-block collisions. To overcome the final additional processing of message block M^1 we use the same first message block in the both messages. 2-block collision in the upper chain (xor of message blocks) can be found because of the form of differences Δ_1 and Δ_2 from equations (2) and (3). We force the messages to have the same modular differences than xor differences what means addition of one condition on message register m_{11} in both colliding blocks. The last problem is to find collision in the lower chain of modular additions.

Let us recall how the modular difference δ from equation (3) is cancelled. Denote the last four registers in computation of compression function by variable S. Then the differentials in Wang's 2-block method for messages $(M^1||M^2)$ and $(N^1||N^2)$ can be described as follows

$$\mathrm{IV}_0 - \mathrm{IV}'_0 = 0 \tag{4}$$
$$S_1 - S'_1 := (2^{31}, 2^{31} + 2^{25}, 2^{31} + 2^{25}, 2^{31} + 2^{25}) = \delta \tag{5}$$
$$\mathrm{IV}_1 - \mathrm{IV}'_1 = (\mathrm{IV}_0 + S_1) - (\mathrm{IV}'_0 + S'_1) = \delta \tag{6}$$
$$S_2 - S'_2 = (2^{31}, 2^{31} - 2^{25}, 2^{31} - 2^{25}, 2^{31} - 2^{25}) = -\delta \tag{7}$$
$$\mathrm{IV}_2 - \mathrm{IV}'_2 = (\mathrm{IV}_1 + S_2) - (\mathrm{IV}'_1 + S'_2) = \delta - \delta = 0 \tag{8}$$

Now if somebody is able to find a colliding pair $(M^1||M^2)$ and $(N^1||N^2)$ with the changed differentials i.e. where differential $S_1 - S'_1 = -\delta$ and $S_2 -$

$S_2' = \delta$, she can create 5-block collision pair m, m' for simplified MFRI in the form

$$m = (M^1 || M^2 || M^3 || M^4 || M^5) \quad m' = (M^1 || N^2 || N^3 || N^4 || N^5),$$

where $(M^2||M^3)$ and $(N^2||N^3)$ forms the 2-block normal collision for initial vector set to IV_1 and $(M^4||M^5)$ and $(N^4||N^5)$ forms the 2-block collision with changed differential for initial vector set to IV_3. By simplified MFRI we mean the MFRI construction without reversing of bit order ($\rightleftarrows$). The problem of finding 2-block collision with changed differential seems to be equivalent to the problem of finding 2-block collision. We think that it should be possible by very small changes in the differential path.

For the construction with operation of reverse bit ordering ($\rightleftarrows$) we must ensure that differentials 2^{31} (which become 20 after reversing the bit order) have the opposite signs. This may add a little obstacle to the collision generating algorithm but we wanted to show that this construction is no more resistant to multi-block collisions than the Merkle-Damgård construction.

3.6 Conclusion

We have tried to present the latest work in cryptanalysis of MD5 and hash functions. We have shown several collision searching algorithms for MD5 and several improvements of the Merkle-Damgård construction. The goal for these improvements was to avoid multi-block collisions but in fact we presented very simple attacks against them. We recommend not using the compression function of MD5 as a part of any scheme which requires collision resistance as a property.

4 A New Type of 2-block Collisions in MD5

4.1 Introduction

At rump session of Crypto 2004 X. Wang presented two pairs of colliding messages for MD5 [27]. A more detailed description of the method for constructing colliding pairs of messages was given in the paper [29] presented at Eurocrypt 2005. Each colliding pair consisted of messages of the same length 1024 bits, i.e. two blocks $(M_1||M_2)$ and $(M_1'||M_2')$. Their modular differences were:

$$\begin{aligned} \delta M_1 = M_1' - M_1 &= (0,0,0,0,2^{31},0,0,0,0,0,0,+2^{15},0,0,2^{31},0) \\ \delta M_2 = M_2' - M_2 &= -\delta M_1 \\ &= (0,0,0,0,2^{31},0,0,0,0,0,0,-2^{15},0,0,2^{31},0) \end{aligned}$$

The most important part of [29] was the so called differential path for each block. The differential path says how modular differences and xor of registers Q_t, Q_t' evolve during the calculation of the MD5 compression function applied to M_1, M_1' and M_2, M_2' resp. However, the paper [29] gives no information about how the differential paths were found.

Since then, many improvements of the collision search algorithm based on the differential paths of Wang et al. have been published. Two most important developments were the multi-block message modification (already mentioned in [29]) and tunneling [10]. These methods decreased the time required for finding a pair of colliding messages to less than one minute on a PC. The theoretical complexity was estimated [6] to 2^{27} calculations of the MD5 compression function for Klima's algorithm [10] and 2^{29} calculations for Stevens' algorithm [23].

A new type of collisions - the so called *chosen prefix collisions* were published in [25]. In case of chosen prefix collisions one starts with different initial vectors IVand IV$'$ with modular difference $\delta\text{IV} = \text{IV} - \text{IV}' = (0,x,x,x)$ and constructs messages M, M' such that $\text{MD5}(\text{IV}, M) = \text{MD5}(\text{IV}', M')$. The number of blocks of M and M' equals the weight of x i.e. the minimal number of non-zero coefficients in any binary signed digit representation (BSDR) of x. The authors used chosen prefix collisions to construct colliding X.509 certificates. A major development of [25] is an algorithm for an automated construction of differential paths.

Another paper to mention in this context is the paper [20] by Yajima et al. The authors point out that there might be colliding pairs of messages with other differences than those of Wang et al. However their estimates of time required for finding colliding pairs with these differences were too high even if the corresponding differential paths were known.

Recently Sasaki et al. [21] made another progress in the study of MD5 collisions. They constructed a differential path that allowed them to find two message blocks M_1, M_1' with modular differences

$$\delta M_1 = M_1' - M_1 \quad = \quad (0,0,0,0,0,0,0,0,0,0,0,+2^{31},0,0,0,0)$$

such that $\mathrm{MD5}(\mathrm{IV}, M_1) - \mathrm{MD5}(\mathrm{IV}, M_1') = (2^{31}, 2^{31}, 2^{31}, 2^{31})$. Then they modify the algorithms of [3] to construct efficiently a message block M_2 such that $\mathrm{MD5}(\mathrm{IV}, M_1||M_2) = \mathrm{MD5}(\mathrm{IV}, M_1'||M_2)$. This is then applied to recover the first 31 letters of the client password used in the APOP. It was significant improvement of the result from [11] and [22], where Wang's et al. collisions enabled recovery of the first three characters of the password.

In this paper we present a new type of 2-block collisions for MD5. We choose one of the differences of messages suggested in [20] and construct the corresponding colliding message pair. In our case the colliding messages $(M_1||M_2)$ and $(M_1'||M_2')$ have differences

$$\begin{aligned}
\delta M_1 = M_1' - M_1 &= (0,0,2^{31},0,0,0,0,0,0,+2^{27},0,0,2^{31},0,0,0) \\
\delta M_2 = M_2' - M_2 &= -\delta M_1 \\
&= (0,0,2^{31},0,0,0,0,0,0,-2^{27},0,0,2^{31},0,0,0)
\end{aligned}$$

We use our own implementation of Stevens differential path searching algorithm (the original implementation has not been published yet) to construct differential paths. We also give some details of our implementation in Section 4.4. As for the algorithm finding colliding messages satisfying a given differential path we also use our own implementation of Klima's algorithm [10]. We do not provide any details since the algorithm based on tunnels is described well in e.g. [10], [24].

Recently Xie et al. announced in [31] a different type of two block colliding messages with differences

$$\begin{aligned}
\delta M_1 = M_1' - M_1 &= (0,0,0,0,0,0,+2^{8},0,0,2^{31},0,0,0,0,0,2^{31}) \\
\delta M_2 = M_2' - M_2 &= -\delta M_1 \\
&= (0,0,0,0,0,0,-2^{8},0,0,2^{31},0,0,0,0,0,2^{31})
\end{aligned}$$

Their collisions also belong among the collisions forecasted in [20], the case $t = 43$, see section 4.3.

4.2 Preliminaries

We follow description and notation from [24]. MD5 can be described as follows:

1. Pad the message with the 1-bit, then as many 0 bits until the resulting length equals 448 mod 512, and the bitlength of the original message expressed as a 64-bit integer. The total bitlength of the padded message is then multiple of 512.

2. Divide the padded message into N consecutive 512-bit blocks $M_1, M_2, \ldots, M_N$.

3. Go through $N + 1$ states IV_i, for $0 \leq i \leq N$, called the intermediate hash values. Each intermediate hash value IV_i consists of four 32-bit words a_i, b_i, c_i, d_i. For $i = 0$ these are initialized to fixed public values: $(a_0, b_0, c_0, d_0) = (\texttt{0x67452301}, \texttt{0xEFCDAB89}, \texttt{0x98BADCFE}, \texttt{0x10325476})$ and for $i = 1, 2, \ldots, N$ intermediate hash value IV_i is computed using the MD5 compression function described below: $\mathrm{IV}_i = H(\mathrm{IV}_{i-1}, M_i)$.

4. The resulting hash value is the last intermediate hash value IV_N.

4.2.1 MD5 compression function

The input for the compression function $H(\mathrm{IV}, M)$ is an intermediate hash value $\mathrm{IV} = (a, b, c, d)$ of length 128bits and a 512-bit message block M. There are 64 steps, each step uses a modular addition, a left rotation, and a non-linear function. Depending on the step t, addition constants C_t and rotation constants s_t (all defined in standard [18]) are used.

The non-linear function f_t is defined by

$$f_t = \begin{cases} F(x,y,z) &= (x \wedge y) \vee (\neg x \wedge z), & \text{for } 0 \leq t \leq 15, \\ G(x,y,z) &= (x \wedge z) \vee (y \wedge \neg z), & \text{for } 16 \leq t \leq 31, \\ H(x,y,z) &= x \oplus y \oplus z, & \text{for } 32 \leq t \leq 47, \\ I(x,y,z) &= y \oplus (x \wedge \neg z), & \text{for } 48 \leq t \leq 63, \end{cases}$$

The message block M is divided into 16 consecutive 32-bit words $m_0, m_1, \ldots, m_{15}$ and expanded to 64 words W_t, for $0 \leq t < 64$, of 32 bits each:

$$W_t = \begin{cases} m_t, & \text{for } 0 \leq t \leq 15, \\ m_{1+5t \bmod 16}, & \text{for } 16 \leq t \leq 31, \\ m_{5+3t \bmod 16}, & \text{for } 32 \leq t \leq 47, \\ m_{7t \bmod 16}, & \text{for } 48 \leq t \leq 63, \end{cases}$$

For $0 \leq t < 64$ the compression function algorithm maintains a working register with 4 state words $Q_t, Q_{t-1}, Q_{t-2}, Q_{t-3}$. These are initialized as $(Q_0, Q_{-1}, Q_{-2}, Q_{-3}) = (b, c, d, a)$ and, for $0 \leq t < 64$ in succession, updated as follows:

$$\begin{aligned} F_t &= f_t(Q_t, Q_{t-1}, Q_{t-2}), \\ T_t &= F_t + Q_{t-3} + C_t + W_t, \\ R_t &= RL(T_t, s_t), \\ Q_{t+1} &= Q_t + R_t. \end{aligned}$$

After all steps are computed, the resulting state words are added to the intermediate hash value and returned as output: $H(\mathrm{IV}, M) = (a + Q_{61}, b + Q_{64}, c + Q_{63}, d + Q_{62})$.

4.2.2 Differential Paths

A differential path for compression function H is a precise description of the propagation of differences through the 64 steps caused by δIV and δM

$$\begin{aligned}
\delta F_t &= f_t(Q'_t, Q'_{t-1}, Q'_{t-2}) - f_t(Q_t, Q_{t-1}, Q_{t-2});\\
\delta T_t &= \delta F_t + \delta Q_{t-3} + \delta W_t;\\
\delta R_t &= RL(T'_t, C_t) - RL(T_t, C_t);\\
\delta Q_{t+1} &= \delta Q_t + \delta R_t.
\end{aligned}$$

We use notation of bitconditions (also taken from [24]) on (Q_t, Q'_t) to describe differential paths, where a single bitcondition specifies directly or indirectly the values of the bits $Q_t[i]$ and $Q'_t[i]$.

$\mathsf{q}_t[i]$	condition on $(Q_t[i], Q'_t[i])$	k_i
$\cdot$	$Q_t[i] = Q'_t[i]$	0
$+$	$Q_t[i] = 0,\ Q'_t[i] = 1$	$+1$
$-$	$Q_t[i] = 1,\ Q'_t[i] = 0$	-1

Table 1: Differential bitconditions

$\mathsf{q}_t[i]$	condition on $(Q_t[i], Q'_t[i])$	direct/indirect	direction
0	$Q_t[i] = Q'_t[i] = 0$	direct	
1	$Q_t[i] = Q'_t[i] = 1$	direct	
^	$Q_t[i] = Q'_t[i] = Q_{t-1}[i]$	indirect	backward
v	$Q_t[i] = Q'_t[i] = Q_{t+1}[i]$	indirect	forward
!	$Q_t[i] = Q'_t[i] = \neg Q_{t-1}[i]$	indirect	backward
y	$Q_t[i] = Q'_t[i] = \neg Q_{t+1}[i]$	indirect	forward
m	$Q_t[i] = Q'_t[i] = Q_{t-2}[i]$	indirect	backward
w	$Q_t[i] = Q'_t[i] = Q_{t+2}[i]$	indirect	forward
#	$Q_t[i] = Q'_t[i] = \neg Q_{t-2}[i]$	indirect	backward
h	$Q_t[i] = Q'_t[i] = \neg Q_{t+2}[i]$	indirect	forward
?	$Q_t[i] = Q'_t[i] \wedge (Q_t[i] = 1 \vee Q_{t-2}[i] = 0)$	indirect	backward
q	$Q_t[i] = Q'_t[i] \wedge (Q_{t+2}[i] = 1 \vee Q_t[i] = 0)$	indirect	forward

Table 2: Boolean function bitconditions.

A *binary signed digit representation* (BSDR) of a word X is a sequence $Y = (k_i)_{i=0}^{31}$, often simply denoted as $Y = (k_i)$, of 32 digits $k_i \in \{-1, 0, +1\}$ for $0 \leq i \leq 31$, where

$$X \equiv \sum_{i=0}^{31} k_i 2^i \mod 2^{32}.$$

A particularly useful BSDR of a word X which always exists is the Non-Adjacent Form (NAF), where no two non-zero k_i's are adjacent. The NAF is not unique since we work modulo 2^{32} (making $k_{31} = -1$ equivalent to $k_{31} = +1$), however we will enforce uniqueness of the NAF by choosing $k_{31} \in \{0, +1\}$. Among the BSDRs of a word, the NAF has minimal weight (see e.g. [17]).

4.3 New 2-block Collisions in MD5

The collisions of Wang et al. [29] make use of a weakness in the message expansion of MD5, in particular in its interplay with the non-linear function in the third round (steps $t = 32, \ldots, 47$). This weakness appears for all $t = 32, \ldots, 44$, not only fot $t = 34$ used in [29]. This had been observed already by Yajima et al. in [20]. They conjectured that any $t = 32, \ldots, 44$ might possible lead to 2-block collisions with similar characteristics.

More specifically there are 13 triplets of possible differences in messages

$$d_t = (\delta W_t, \delta W_{t+1}, \delta W_{t+3}) = (\delta m_{(5+3i) \mod 16}, \delta m_{(8+3i) \mod 16}, \delta m_{(14+3i) \mod 16}),$$

where $i = t-32$, for which one can hope to construct 2-block collision similar to those ones by Wang et al. They are summarized in Table 4.3.

step	δQ_t	δF_t	δW_t	δT_t	δR_t	s_t
t	0	0	$\pm 2^{31-s_t}$	$\pm 2^{31-s_t}$	$\cdot 2^{31}$	
$t+1$	$\cdot 2^{31}$	$\cdot 2^{31}$	$\cdot 2^{31}$	0	0	
$t+2$	$\cdot 2^{31}$	0	0	0	0	
$t+3$	$\cdot 2^{31}$	$\cdot 2^{31}$	$\cdot 2^{31}$	0	0	

Table 3: Generalized Wang type differentials

We believe this pattern must had been already known to Wang et al, because they had chosen for their collisions the triplet defined by $t = 34$ which requires the smallest number of conditions in rounds 2–4 to have manageable computational complexity. An obvious strategy for choosing the t and the corresponding triplet of differences to have a differential path with manageable computational complexity is to obey the following conflicting requirements:

1. Choose the triplet d_t such that the differences in messages appear in the second round as soon as possible, in particular the difference in $\delta m_{5+3i \mod 16}$.

2. Choose the triplet d_t such that the difference in $\delta m_{5+3i \mod 16}$ appears in the fourth round as late as possible.

However, to find colliding messages with the forecasted differences requires to find a differential path including a partial collision after two rounds, the prescribed differences in the third round and their consequences in the rest of round 3 and in round 4. This is what Wang et al. did in [29]. Yajima et al. in [20] constructed a partial differential path for steps $17, \dots, 64$ in the case of differences d_{44}, and estimated the number of conditions in rounds 2 to 4 for their partial differential path. They presented a table comparing the number of conditions in rounds 2 to 4 for the differential path for the first block of Wang et al. for the differences d_{34}, and the partial differential path they proposed for the differences d_{44}.

round	Wang: d_{34}	Yajima: d_{44}	ours: d_{44}
2	15	52	50
3	0	0	0
4	20	17	16

Table 4: Number of conditions in rounds 2 to 4 for the first block

In this paper we present full differential paths for both blocks for the differences d_{44} and examples of the 2-block colliding messages with these differences. We constructed the full differential paths using our own implementation of Stevens algorithm described in [25] and [24]. The differential path for the first block we constructed differs in the second round from the partial path proposed by Yajima et al. In table 4 we also present the number of conditions on Q_t without the conditions on T_t for our differential path for the first block. The numbers taken from the paper by Yajima et al. are also the numbers of conditions on Q_t without the conditions on T_t that were completely missing in the paper by Wang et al. [29].

The differential paths we constructed and the corresponding colliding messages are presented in Section 4.6. The actual colliding messages were found by an algorithm involving Klima's tunnels which is similar to the near-collision block searching algorithm presented by Stevens in [24].

4.4 On Our Implementation of Stevens Algorithm

In this section, we discuss the details of our implementation of Stevens algorithm for generating differential paths. This algorithm can be divided into two main parts

- extending partial differential paths,
- connecting partial differential paths.

We use the following terminology for partial differential paths. An *upper path* is a partial differential path generated forward from an IV, a *lower path* is a partial differential path generated backward from the registers $64, \ldots, 61$. Note that our terminology differs from the one used by Stevens in [25].

4.4.1 Extending Partial Differential Paths

We provide more information on backward generation of lower paths. We start with a partial differential path for steps $63, \ldots, 31$. This path is kept fixed through out the whole run of the algorithm. We constructed it by hand in the simplest possible way. This lower path is presented in table 5.

t	δQ_t	δF_t	δW_t	δT_t	δR_t	s_t
28	$\cdot 2^{31}$					
29	0					
30	0					
31	0	0	$\cdot 2^{31}$	0	0	20
32–43	0	0	0	0	0	
44	0	0	$\pm 2^{27}$	$\pm 2^{27}$	$\cdot 2^{31}$	4
45	$\cdot 2^{31}$	$\cdot 2^{31}$	$\cdot 2^{31}$	0	0	11
46	$\cdot 2^{31}$	0	0	0	0	16
47	$\cdot 2^{31}$	$\cdot 2^{31}$	$\cdot 2^{31}$	0	0	23
48-51	$\cdot 2^{31}$	$\cdot 2^{31}$	0	0	0	
52	$\cdot 2^{31}$	0	$\cdot 2^{31}$	0	0	6
53-61	$\cdot 2^{31}$	$\cdot 2^{31}$	0	0	0	
62	$\cdot 2^{31}$	0	$\cdot 2^{31}$	0	0	15
63	$\cdot 2^{31}$	$\cdot 2^{31}$	$\pm 2^{27}$	$\pm 2^{27}$	$\pm 2^{16}$	21
64	$\pm 2^{16}$	-	-	-	-	

Table 5: Partial lower differential path with δ

The Stevens algorithm for extending differential paths uses 3 basic choices at each step t.

1. A choice of a BSDR of δQ_t.
2. A choice of $\delta F_t[i]$ for $i = 0, \ldots, 31$. This choice determines a BSDR of δF_t.
3. A choice of a BSDR of δT_t (in the case of generating upper paths forward) or δR_t (in the case of generating lower paths backward).

To limit the number of possible choices of BSDR's for δQ_t we use the following 4 basic parameters

(a) *max_nbr* is the maximal number of BSDR's of δQ_t,

(b) *max_ dif* is the the maximal difference between the weight of a BSDR of δQ and the weight of its NAF,

(c) *max_ len* is the maximal length of carry propagation,

(d) *max_ prp* is the maximal number of carry propagations.

To choose a BSDR of δQ_t within the limits specified by the parameters one can use different approaches. One possibility is to generate randomly a BSDR satisfying the parameters and then to continue to the next choice. Another possibility is to generate all BSDR's satisfying all four parameters and then either to choose randomly from all generated possibilities or deterministically in some prescribed order.

Stevens mentions in his thesis that he sets up *max_ dif* $= 2$ and then he chooses δQ_t randomly among all BSDR's satisfying this condition. The advantage of this approach is speed.

In our implementation we have selected the other approach and generate all BSDR's satisfying all four parameters. Then we choose a particular BSDR deterministically. Our approach gives us information about the number of possible choices of BSDR's and therefore it provides us with some information about the tree of all possible extensions of partial differential paths.

To generate all BSDR's of δQ_t (within the limits set up by the four parameters) exactly once we developed our own algorithm. The details of the algorithm and a proof of its correctness will be presented in another paper.

The second choice is to pick $\delta F_t[i]$, for $i = 0, \dots 31$, and therefore BSDR of δF_t. In what follows we use notations and definitions of sets U_{abc}, V_{abc} and $W_{abc,g}$ from subsection 5.5.2 of [25]. This choice depends on the precomputed values of the functions $FC(t, abc, g)$ and $BC(t, abc, g)$, where $a = \mathsf{q}_t[i]$, $b = \mathsf{q}_t[i-1]$, $c = \mathsf{q}_t[i-2]$ are bitconditions, and $g \in \{0, 1, -1\}$.

There are again two different approaches to choose $\delta F_t[i]$ (that is a BSDR of δF_t). One possibility is to choose $\delta F_t[i]$'s randomly from the set V_{abc} provided $|V_{abc}| > 1$. This is the approach used by Marc Stevens in his thesis [24]. This leads to random selection of BSDR's of δF_t. Our approach is to limit the number of possible choices for a BSDR of δF_t by a parameter *max_ dF* and, if $|V_{abc}| > 1$, we choose $\delta F_t[i] \in V_{abc}$ in the prescribed order 0, 1, -1. We proceed from $i = 0$ to $i = 31$.

The third choice is to pick a BSDR of δR_t. Depending on the choice of BSDR of δR_t there are at most four possibilities for δT_t. Stevens describes in his thesis how he chooses the most probable one. In our implementation we choose either the NAF of δR_t or the BSDR that differs from the NAF of δR_t in the sign at the leading bit.

The algorithm for generating upper paths forward differs from the one for generating lower paths backward in inessential details.

4.4.2 Connecting Partial Differential Paths

We generate partial upper differential paths forward up to step 12 (the last computed value is δQ_{13}) and partial lower differential paths backward up to step 17 (the last computed value is δQ_{14}). The choice of the bounds is the same as in [24].

We have implemented the algorithm for connecting differential paths described in [25] without any modifications. It should be noted however that the output of the algorithm depends on the order of some steps in the algorithm and on the data structures used to keep the intermediate results.

We supplement the connecting algorithm with the check if the rotation of δT_t, $t = 11, \ldots, 15$, leads to the correct δR_t selected in the extending parts of the algorithm. We try all possibilities for free bits in registers $Q_{11}, \ldots, Q_{16}$ and when there exists the possibility providing correct rotation of δT_t, we fix free bits and continue with the collision generating part of algorithm.

The connecting algorithm seems to have surprisingly high success rate. Stevens in a test run of his improved connecting algorithm successfully connected 52 pairs of upper and lower paths out of $2.5 \cdot 10^5 \times 5 \cdot 10^5$ attempted pairs.

In our implementation the ratio of successfully connected pairs appears to be very sensitive on the choice of parameters for generating partial differential paths, especially the parameter max_dF. The distribution of the number of successfully connected pairs in different runs of our implementation was rather irregular, but on average we constructed about 126 full differential paths out of $8 \cdot 10^4 \times 2 \cdot 10^5$ pairs of upper and lower paths for the first block and 4 full differential paths for the second block. However, without it no reasonable estimate of the success rate of the connecting partial differential path algorithm can be made and the number of test runs is not high enough to make any reasonable conclusions. In any case, this observed phenomenon calls for deeper theoretical investigation.

4.4.3 Choosing Parameters

The number of generated partial differential paths in a given time appears to be extremely sensitive on the choice of the parameters.

We present in table 6 the parameters we used for generating lower paths in each step. The parameter *max_con* for step t denotes the total number of conditions from the start of generation of partial paths to step t. We used the same parameters for both blocks.

The strategy for generating lower paths was to set the parameters in such a way that the following goals were achieved.

- The number of possible lower paths generated using chosen parameters is sufficient for the next (connecting) part of the algorithm (from about $5 \cdot 10^4$ to $2.5 \cdot 10^5$).

t	max_dif	max_len	max_prp	max_nbr	max_dF	max_con
15	-	-	-	-	-	67
16	-	-	-	-	-	60
17	2	2	2	10	1000	51
18	2	2	2	10	1000	41
19	2	2	2	10	1000	39
20	2	2	2	10	1000	35
21	2	2	2	10	1000	24
22	2	2	2	10	1000	20
23	2	2	2	10	1000	18
24	2	2	2	10	1000	11
30-25	2	1	2	10	1000	10

Table 6: The parameters for partial lower paths

- The time needed to generate sufficient number of possible lower paths using chosen parameters is feasible (less than 1 day on single PC).
- The number of conditions in steps $17, \ldots, 30$ is as small as possible.
- In particular, the number of conditions in steps $2, \ldots, 30$ is as small as possible.

Setting the parameters is not straightforward and the values were obtained after some experimentation. There might be better choices and a theoretical understanding for an automated choice of the parameters is needed.

The strategy for the generating upper paths was not formulated in such detail. The goal was to limit the number of conditions in steps $0, \ldots, 12$ in such a way that the sufficient number of upper differential paths was generated in few hours. The total number of conditions is 80 for the first block and 180 for the second block.

4.5 Conclusion

We presented a new type of 2-block MD5 collisions. We found them using our implementation of Stevens algorithm. The implementation can be used to construct differential paths for other types of differences in messages stated in [20], i.e. to construct target or 2-block collisions in MD5.

4.6 Differential paths and collision example

	first block	second block
-3	....	....
-2	....	+....... ....1 ....
-1	....	+....... ....1 ....
0	....	+....... ....+ ....
1	....v	+....... ......v+ ....v.
2	....1	+....... ......1+ .... v.....1.
3	....+	+....... v.....+. .... 1.....+.
4	...v.... ....+	+v.v.... 1.....+1 ..v..... +.....+.
5	...1.... ....+ ....v..	+1.1..v. +...v.+. ..1..... +...v.+.
6	...+.... ....+ .... v....1..	.-.-..1. +.v.1.+. .v+...v. +...1.+.
7	...+...v ....+ .... 1..v.+..	1-.-.v+v -v..+v+v .0+..v.. +.v.+.+.
8	.0.+...0 ....+ .v...v.v +..1.+..	.1.-v.+0 +1+.+0+. v+.v.0+v -v.v+v+v
9	.1.+..1+ ....v..+ .....1.1 +..+.-..	.+.-1+.+ 1-0..-.- 1-1..+0. -1+11.+.
10	.+.-.v0+ ....11.+ .+...+0- .vv+.-..	.1..-10- 1-1.1+10 -00+.-1+ .+0+1+.-
11	v+0-0.+1 10v1+0v. v1v0.+1+ 1..+000.	y-.1-1.- 11-.1-11 -1-0.-00 0+++-010
12	.11+1++1 0100++11 0-110++0 0-0111.	00v.-1v+ 0++0-101 010-v11- 11101000
13	+1+-+0- -+-11-1 ++-1+1- +11+--.	++0..+.+ -00100+- -++.++1 -0++-+.
14	1++0+-11 +-1+-- -++++- 0+01000.	+1+.1-+ +-0+++++ +10+-101 0+.+-01.
15	11+10+11 11+01010 01011v11 -1.0110.	y1-..-11 00-000+0 +000010+ 0-.1+10.
16	.11.10.. -11.0100 -100+.10 01..0...	..1..110 +0111011 +..1-..- .0.01...
17	..^..^.. ..0...1. ....+..1 1+..-...	..^..^.. ..0...1. 1...-..0 .-..+...
18	.... ^.....+. ^...-..- ....0...	.... ^.....-. 0...+..+ ....0...
19	.....0.. ..-...0. ....0... .^..+...	.....0.. ..+...0. ....0... .^..-...
20	.....1.. ..-...1. ....-..^	.....1.. ..+...1. ....+..^ ....
21	0....-.. ..+...1. ....^...	0....+.. ..-...1. ....^...
22	1....0.. ..0...-. ....^...	1....0.. ..0...+. ....^... ....
23	+....0.. ..1.....	+....0.. ..1..... ....
24	0....1.. ......^.	1....1.. ......^. ....
25	-....+..	+....-.. ....
26	-.......	-....... ....
27	+....^..	+....^.. ....
28	+.......	+....... ....
29	0.......	0....... ....
30	!.......	!....... ....

Table 7: Differential path for the first and second blocks without tunnels, registers -3 to 30

31-45	....	....	....	....
46	I.......	....	....	....
47	J.......	....	....	....
48	I.......	....	....	....
49	J.......	....	....	....
50	I.......	....	....	....
51	J.......	....	....	....
52	K.......	....	....	....
53	J.......	....	....	....
54	K.......	....	....	....
55	J.......	....	....	....
56	K.......	....	....	....
57	J.......	....	....	....
58	K.......	....	....	....
59	J.......	....	....	....
60	K.......	....	....	....
61	J.......	....	....	....
62	I.......	....	....	....
63	J.......	....	....	....
64	....	....	....	....

Table 8: Differential path for both blocks, registers 31 to 64. I, J, K $\in \{0, 1\}$, I $\neq$ K.

IV	0x67452301	0x10325476	0x98badcfe	0xefcdab89
M_1	0xCE7E83CA	0xCADE345E	0xB81D83A5	0x562EDF19
	0xB93C9D41	0xF9C4E244	0x5B9B832F	0xE16D2FE5
	0x4B286759	0xF9FE0301	0xA912EF12	0x95A85769
	0x18ADF66C	0x8B1AD802	0x291B44AB	0x732AF6A2
N_1	0xCE7E83CA	0xCADE345E	0x381D83A5	0x562EDF19
	0xB93C9D41	0xF9C4E244	0x5B9B832F	0xE16D2FE5
	0x4B286759	0x01FE0301	0xA912EF12	0x95A85769
	0x98ADF66C	0x8B1AD802	0x291B44AB	0x732AF6A2
IV_1	0xFADBF815	0x1B73566D	0x6BCF3C99	0x5D6E2DFF
IV_1'	0x7ADBF815	0x9B73566D	0xEBCF3C99	0xDD6F2DFF
$\mathrm{IV}_1 \oplus \mathrm{IV}_1'$	0x80000000	0x80000000	0x80000000	0x80010000
M_2	0x6A9B0D7D	0x9AAEEDA9	0x62255628	0xB6A85040
	0xC7E08FD1	0x077E530A	0xDEDD6809	0xD20A7D80
	0x55DFBE93	0x78571C29	0xC13D746C	0x062792C8
	0x45A152CE	0x69727500	0x351EC8F7	0xCFFFAF73
N_2	0x6A9B0D7D	0x9AAEEDA9	0xE2255628	0xB6A85040
	0xC7E08FD1	0x077E530A	0xDEDD6809	0xD20A7D80
	0x55DFBE93	0x70571C29	0xC13D746C	0x062792C8
	0xC5A152CE	0x69727500	0x351EC8F7	0xCFFFAF73
$\mathrm{IV}_2 = \mathrm{IV}_2'$	0xA5A29F9F	0xBC622670	0x54E1D520	0xE6FA818E

Table 9: Collision example

References

[1] Eli Biham, Rafi Chen, Antoine Joux, Patrick Carribault, Christophe Lemuet, and William Jalby. Collisions of sha-0 and reduced sha-1. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 36–57. Springer Berlin Heidelberg, 2005.

[2] Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.

[3] Bert den Boer and Antoon Bosselaers. Collisions for the compression function of md5. In *Workshop on the theory and application of cryptographic techniques on Advances in cryptology*, EUROCRYPT '93, pages 293–304, Secaucus, NJ, USA, 1994. Springer-Verlag New York, Inc.

[4] Brier E. et al. Linearization framework for collision attacks: Application to cubehash and md6 (extended version). Cryptology ePrint Archive, Report 2009/382, 2009. `http://eprint.iacr.org/`.

[5] Praveen Gauravaram, William Millan, Ed Dawson, and Kapali Viswanathan. Constructing secure hash functions by enhancing Merkle-Damgård Construction, url = http://dx.doi.org/10.1007/11780656_34, year = 2006. *Lecture Notes in Computer Science : Information Security and Privacy*, pages 407–420.

[6] Daniel Joščák. Finding collisions in cryptographic hash functions. Master's thesis, Charles University in Prague, 2006. `http://cryptography.hyperlink.cz/2006/diplomka.pdf`.

[7] Daniel Joščák. Beyond the md5 collisions. Security and Protection of Information, 2007. `http://spi.unob.cz/papers/2007/2007-04.pdf`.

[8] Daniel Joščák and Jiří Tůma. Multi-block collisions in hash functions based on 3c and 3c+ enhancements of the merkle-damgard construction. *Lecture Notes in Computer Science: Proceedings of ICISC*, 4296:257 – 266, 2006.

[9] Irving Kaplansky. Linear algebra and geometry, a second course. Chelsea Publishing Company, 1974.

[10] Vlastimil Klima. Tunnels in hash functions: Md5 collisions within a minute. Cryptology ePrint Archive, Report 2006/105, 2006. `http://eprint.iacr.org/`.

[11] Gaëtan Leurent. Message freedom in md4 and md5 collisions: Application to apop. In Alex Biryukov, editor, *Fast Software Encryption*, volume 4593 of *Lecture Notes in Computer Science*, pages 309–328. Springer Berlin / Heidelberg, 2007.

[12] Gaëtan Leurent and Søren S. Thomsen. Practical partial-collisions on the compression function of bmw. In *Fast Software Encryption - 18th International Workshop, FSE 2011*, volume 6733 of *Lecture Notes in Computer Science*, page 238. Springer, 2011.

[13] Jie Liang and Xue-Jia Lai. Improved collision attack on hash function md5. *J. Comput. Sci. Technol.*, 22:79–87, January 2007.

[14] Helger Lipmaa and Shiho Moriai. Efficient algorithms for computing differential properties of addition. In Mitsuru Matsui, editor, *FSE*, volume 2355 of *Lecture Notes in Computer Science*, pages 336–350. Springer, 2001.

[15] Helger Lipmaa, Johan Wallén, and Philippe Dumas. On the additive differential probability of exclusive-or. In Bimal Roy and Willi Meier, editors, *Fast Software Encryption*, volume 3017 of *Lecture Notes in Computer Science*, pages 317–331. Springer Berlin Heidelberg, 2004.

[16] Lucie Marková. Analysis of proposals of new hash functions for sha-3 competition. Master's thesis, Charles University in Prague, 2010.

[17] James A. Muir, Douglas, and R. Stinson. Minimality and other properties of the width-w nonadjacent form. *Mathematics of Computation*, 75:369–384.

[18] Ron Rivest. The md5 message-digest algorithm. *Request for Comments: 1321*, 1992.

[19] Su S., Yang Y., Yang B., and Zhang S. The design and analysis of a hash ring-iterative structure. Cryptology ePrint Archive, Report 2006/384, 2006. `http://eprint.iacr.org/2006/384`.

[20] Yu Sasaki, Yusuke Naito, Jun Yajima, Takeshi Shimoyama, Noboru Kunihiro, and Kazuo Ohta. How to construct sufficient condition in searching collisions of md5. Cryptology ePrint Archive, Report 2006/074, 2006. `http://eprint.iacr.org/`.

[21] Yu Sasaki, Lei Wang, Kazuo Ohta, and Noboru Kunihiro. Security of md5 challenge and response: Extension of apop password recovery attack. In Tal Malkin, editor, *Topics in Cryptology – CT-RSA 2008*, volume 4964 of *Lecture Notes in Computer Science*, pages 1–18. Springer Berlin / Heidelberg, 2008. `http://dx.doi.org/10.1007/978-3-540-79263-5_1`.

[22] Yu Sasaki, Go Yamamoto, and Kazumaro Aoki. Practical password recovery on an md5 challenge and response. Cryptology ePrint Archive, Report 2007/101, 2007. `http://eprint.iacr.org/`.

[23] Marc Stevens. Fast collision attack on md5. Cryptology ePrint Archive, Report 2006/104, 2006. `http://eprint.iacr.org/`.

[24] Marc Stevens. On collisions for md5. Master's thesis, Eidhoven University of Technology, 2007.

[25] Marc Stevens, Arjen Lenstra, and Benne Weger. Chosen-prefix collisions for md5 and colliding x.509 certificates for different identities. In *Proceedings of the 26th annual international conference on Advances in Cryptology*, EUROCRYPT '07, pages 1–22, Berlin, Heidelberg, 2007. Springer-Verlag.

[26] Jiří Vábek, Daniel Joščák, Milan Boháček, and Jiří Tůma. A new type of 2-block collisions in md5. In *Proceedings of the 9th International Conference on Cryptology in India: Progress in Cryptology*, INDOCRYPT '08, pages 78–90, Berlin, Heidelberg, 2008. Springer-Verlag.

[27] Xiaoyun Wang, Dengguo Feng, Xuejia Lai, and Hongbo Yu. Collisions for hash functions md4, md5, haval-128 and ripemd. Cryptology ePrint Archive, Report 2004/199, 2004. `http://eprint.iacr.org/`.

[28] Xiaoyun Wang, Yiqun L. Yin, and Hongbo Yu. Finding collisions in the full SHA-1. *Lecture Notes in Computer Science*, 3621:17–36, November 2005.

[29] Xiaoyun Wang and Hongbo Yu. How to break md5 and other hash functions. *Lecture Notes in Computer Science: Proceedings of EUROCRYPT*, 2005.

[30] Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin. Efficient collision search attacks on sha-0. *Lecture Notes in Computer Science: Proceedings of Crypto*, pages 1–16, 2005.

[31] Tao Xie, DengGuo Feng, and FanBao Liu. A new collision differential for md5 with its full differential path. Cryptology ePrint Archive, Report 2008/230, 2008. `http://eprint.iacr.org/`.

[32] Jun Yajima and Takeshi Shimoyama. Wang's sufficient conditions of md5 are not sufficient. Cryptology ePrint Archive, Report 2005/263, 2005. `http://eprint.iacr.org/`.