5 Differential cryptanalysis and paths

Contents

5.1 Introduction

In this thesis we limit ourselves to hash functions of the MD4 family and focus mainly on MD5 and SHA-1. As can be seen in Chapter 2, the construction of a near-collision attack for MD4 style compression functions has many aspects that all have to fit together. Nevertheless there is one aspect that is the key to the entire attack: the differential path. In this section we introduce a definition of a compression function family $\mathcal{F}_{\mathrm{md4cf}}$ that includes an important subset of the family of MD4 style compression functions, namely those of MD4, MD5 and SHA-1. We complement this family with a definition of differential paths for these compression functions. Such a differential path is called *valid* if there exists a solution to the differential path under a

relaxation of the message expansion. The most important contribution of this section is an algorithm that searches for complete valid differential paths for which the beginning and ending part are predetermined. In its generic form, this algorithm provides insights in differential cryptanalysis applied to compression functions. We apply this differential cryptanalysis and improve the differential path construction algorithm for MD5 and SHA-1 in Sections 6 and 7, respectively.

For MD4 style compression functions, a differential path designed for a near-collision attack consists of three segments. The most important segment, namely the end segment, consists of all steps after a certain step K whose success probability directly contributes reciprocally to the final attack complexity. E.g., the collision finding algorithm for MD5 in Chapter 2 easily fulfills the first 16 steps of the differential path and with effort can fulfill a small number of steps thereafter using message modification techniques. Since message modification techniques do not affect the fulfillment of the differential path over all steps starting at K, where K is approximately 27 for MD5, the differential path over those steps has to be fulfilled probabilistically. The expected number of attempts required, which is the reciprocal of the success probability over those steps, is a direct factor of the attack complexity. Hence, to build an efficient collision attack the success probability of a differential path over those steps should be as high as possible. Specifically, this probability must be at least $\pi^{-0.5}2^{-N/2}$ for the resulting attack not to be slower than a brute-force attack.

The first segment of the differential path consists of the initial working state defined by the input pair IHV_{in} and IHV'_{in}. Since for a near-collision attack the input pair IHV_{in} and IHV'_{in} are given, this segment is also a given.

The remaining segment thus 'connects' the begin segment and end segment of the differential path. Whereas the end segment is constructed and specifically optimized for a low attack complexity and the begin segment is pre-determined, this segment must be constructed using both extremal segments in order to build a valid complete differential path. There is no trivial solution to the problem of constructing such a connecting segment, due to the fact that each working state variable affects multiple steps of the differential path.

The first ones to solve the problem of constructing a connecting segment were Xiaoyun Wang and her team. Their paper [WY05] describes their methodology which depends mainly on expertise, intuition and patience. With their methodology they enabled the construction of identical-prefix collisions for MD5. In this section we present an algorithmic solution to this problem for a certain class of compression functions which we call $\mathcal{F}_{\text{md4cf}}$. To this end, we first define this class $\mathcal{F}_{\text{md4cf}}$ of compression functions which includes those of MD4, MD5 and SHA-1, and more formally define the concept of a differential path for this class.

Besides providing a much easier and faster way of constructing full differential paths, our algorithmic solution also allows optimization of differential paths with respect to given message modification techniques. Furthermore, the most important result of our algorithmic solution is the construction of a chosen-prefix collision attack against MD5, where the differential paths are necessarily created during the attack instead of precomputed (see Section 6.5). Compared to identical-prefix colli-

sion attacks such as the collision attack in Chapter 2, a chosen-prefix collision attack removes the identical input IHV_k and IHV'_k requirement at the start of the attack. The two equal-length input message prefixes P and P' (resulting in IHV_k and IHV'_k) can therefore be chosen independently.

First in Section 5.2 we introduce some basic definitions and notations necessary for this section. In Section 5.3 we present the formal definition of the class $\mathcal{F}_{\text{md4cf}}$ of MD4 based compression functions. This is followed by a discussion of three important properties of the step functions of the compression functions in $\mathcal{F}_{\text{md4cf}}$ in Section 5.4. Next, we formally define differential paths for the class $\mathcal{F}_{\text{md4cf}}$ of compression functions in Section 5.5. Finally, we present our differential path construction algorithm in Section 5.6.

5.2 Definitions and notation

Throughout this thesis we denote $a \bmod b$ for the least non-negative residue of a modulo b where $a \in \mathbb{Z}$ and $b \in \mathbb{N}^+$.

5.2.1 N-bit words and operators

Similar to MD5, the class $\mathcal{F}_{\text{md4cf}}$ is based on the integer working registers of modern CPU architectures. Whereas MD5 used 32-bit words, here we generalize this to N-bit words. We use the shorter notation $\mathbb{Z}_{2^N}$ for $\mathbb{Z}/2^N\mathbb{Z}$. An N-bit word $(v_i)_{i=0}^{N-1}$ consists of N bits $v_i \in \{0,1\}$. These N-bit words are identified with elements $v = \sum_{i=0}^{N-1} v_i 2^i$ of $\mathbb{Z}_{2^N}$ and we switch freely between these two representations.

On N-bit words $X = (x_i)_{i=0}^{N-1}$ and $Y = (y_i)_{i=0}^{N-1}$, we define the following operations:

- $X \wedge Y = (x_i \wedge y_i)_{i=0}^{N-1}$ is the bitwise AND of X and Y;
- $X \vee Y = (x_i \vee y_i)_{i=0}^{N-1}$ is the bitwise OR of X and Y;
- $X \oplus Y = (x_i \oplus y_i)_{i=0}^{N-1}$ is the bitwise XOR of X and Y;
- $\overline{X} = (1 - x_i)_{i=0}^{N-1}$ is the bitwise complement of X;
- $X[i]$ is the i-th bit x_i;
- $X + Y$ and $X - Y$ denote addition and subtraction, respectively, of X and Y in $\mathbb{Z}_{2^N}$;
- $RL(X,n) = (x_{(i+n \bmod N)})_{i=0}^{N-1}$ is the cyclic left rotation of X by $0 \leq n < N$ bit positions;
- $RR(X,n) = (x_{(i-n \bmod N)})_{i=0}^{N-1}$ is the cyclic right rotation of X by $0 \leq n < N$ bit positions. Equivalent to cyclic left rotation over $(N-n \bmod N)$ bit positions;
- $w(X)$ denotes the Hamming weight $\sum_{i=0}^{N} x_i$ of $X = (x_i)_{i=0}^{N}$.

5.2.2 Binary signed digit representations

We extend the notion of the BSDR to N-bit words $X \in \mathbb{Z}_{2^N}$ as a sequence $(k_i)_{i=0}^{N-1}$ such that

$$X = \sum_{i=0}^{N-1} k_i 2^i, \quad k_i \in \{-1, 0, 1\}.$$

For each non-zero $X \in \mathbb{Z}_{2^N}$ there exist many different BSDRs. The *weight* $w((k_i)_{i=0}^{N-1})$ of a BSDR $(k_i)_{i=0}^{N-1}$ is defined as the number of non-zero k_is.

A particularly useful type of BSDR is the Non-Adjacent Form (NAF), where no two non-zero k_i-values are adjacent. For any $X \in \mathbb{Z}_{2^N}$ there is no unique NAF, since we work modulo 2^N (making $k_{N-1} = +1$ equivalent to $k_{N-1} = -1$). However, uniqueness of the NAF can be enforced by the added restriction $k_{N-1} \in \{0, +1\}$. Among the BSDRs for a given $X \in \mathbb{Z}_{2^N}$, the NAF has minimal weight [MS06]. The NAF can be computed easily [Lin98] for a given $X \in \mathbb{Z}_{2^N}$ as $\mathrm{NAF}(X) = ((X+Y)[i] - Y[i])_{i=0}^{N-1}$ where Y is the N-bit word $(0\ X[N-1]\ \ldots\ X[1])$.

We use the following notation for an N-digit BSDR Z:

- $Z[i]$ is the i-th signed bit k_i of $Z = (k_i)_{i=0}^{N-1}$;
- $RL(Z, n) = (x_{(i+n) \bmod N})_{i=0}^{N-1}$ is the cyclic left rotation of Z by $0 \leq n < N$ digit positions;
- $RR(Z, n) = (x_{(i-n) \bmod N})_{i=0}^{N-1}$ is the cyclic right rotation of Z by $0 \leq n < N$ digit positions and is equivalent to $RL(Z, (N - n \bmod N))$;
- $w(Z) = \sum_{i=0}^{N-1} |k_i|$ is the weight of Z.
- $\sigma(Z) = \sum_{i=0}^{N-1} k_i 2^i \in \mathbb{Z}_{2^N}$ is the N-bit word for which Z is a BSDR.

5.2.3 Function families over N-bit words

As an aid we define three families $\mathcal{F}_{\mathrm{sumrot}}$, $\mathcal{F}_{\mathrm{bool}}$ and $\mathcal{F}_{\mathrm{boolrot}}$ of functions

$$f : (\mathbb{Z}_{2^N})^J \to \mathbb{Z}_{2^N}, \quad J \in \mathbb{N}$$

that map an input tuple of J N-bit words to a single N-bit word.

The family $\mathcal{F}_{\mathrm{sumrot}}$ consists of functions f that perform a selective sum over bitwise cyclic rotated input words and a chosen constant value $C \in \mathbb{Z}_{2^N}$:

$$f(X_1, \ldots, X_J) \mapsto C + \sum_{j=1}^{J} c_j \cdot RL(X_j, r_j), \quad c_j \in \{-1, 0, 1\},\ r_j \in \{0, \ldots, N-1\}.$$

We restrict r_j to zero whenever $c_j = 0$ for $j = 1, \ldots, J$, so that all non-trivial rotations ($r_j \neq 0$) contribute in the sum.

The family $\mathcal{F}_{\text{bool}}$ consists of functions f that extend a boolean functions $g : \{0,1\}^J \to \{0,1\}$ to words:

$$f(X_1, \ldots, X_J) \mapsto (g(X_1[i], \ldots, X_J[i]))_{i=0}^{N-1}.$$

The family $\mathcal{F}_{\text{boolrot}}$ consists of functions that first cyclically rotate their input words and then pass them to a function $g \in \mathcal{F}_{\text{bool}}$:

$$f(X_1, \ldots, X_J) \mapsto g(RL(X_1, r_1), \ldots, RL(X_J, r_J)), \quad r_j \in \{0, \ldots, N-1\}.$$

Observation 5.1. *For any $f \in \mathcal{F}_{sumrot}$ if we fix all input values except X_i, $i \in \{1, \ldots, J\}$, then the resulting f_i is either bijective and easily invertible or a constant function with respect to the remaining single input value. More formally, let $f \in \mathcal{F}_{sumrot}$:*

$$f(X_1, \ldots, X_J) = C + \sum_{j=1}^{J} c_j \cdot RL(X_j, r_j), \quad c_j \in \{-1, 0, 1\},\ r_j \in \{0, \ldots, N-1\}.$$

For any $i \in \{1, \ldots, J\}$ and given values of all inputs except X_i we define the function $f_i : \mathbb{Z}_{2^N} \to \mathbb{Z}_{2^N}$ as:

$$f_i : X_i \mapsto f(X_1, \ldots, X_J) = C_i + c_i \cdot RL(X_i, r_i),$$

where

$$C_i = C + \sum_{\substack{j=0 \\ j \neq i}}^{J} c_j \cdot RL(X_j, r_j).$$

If $c_i = 0$ then $f_i(X_i) = C_i$ is a constant function. Otherwise, if $c_i \in \{-1, 1\}$ then $f_i(X_i) = C_i + c_i RL(X_i, r_i)$ is bijective and easily invertible:

$$f_i^{-1} : A \mapsto RR(c_i \cdot (A - C_i), r_i).$$

5.3 $\mathcal{F}_{\text{md4cf}}$: MD4 based compression functions

The class $\mathcal{F}_{\text{md4cf}}$ of compression functions can be seen as a subfamily of the family of MD4-style compression functions and consists of all compression functions Compress as defined in this section. The class $\mathcal{F}_{\text{md4cf}}$ includes the compression functions of MD4, MD5 and SHA-1. It does not include the compression functions of the SHA-2 family, since these compression functions update two state variables per step instead of one.

A compression function Compress uses only fixed sized N-bit words, $N \in \mathbb{N}^+$, and the above listed N-bit word operations. From now on, we also use the shorthand *word* for N-bit word.

For fixed $L, K \in \mathbb{N}^+$, Compress(IHV_{in}, M) maps an L tuple of words IHV_{in} and a K tuple of words M to an L tuple of words referred to as IHV_{out}:

$$\text{Compress} : (\mathbb{Z}_{2^N})^L \times (\mathbb{Z}_{2^N})^K \to (\mathbb{Z}_{2^N})^L.$$

To compute IHV_{out} given IHV_{in} and M, Compress computes a sequence of $S+L$ working state words $(Q_i)_{i=-L+1}^{S} \in (\mathbb{Z}_{2^N})^{S+L}$, where S is a fixed multiple of K. The first L words $Q_{-L+1}, \ldots, Q_0$ are initialized using IHV_{in} in Section 5.3.1. The remaining S words $Q_1, \ldots, Q_S$ are sequentially computed using *step functions* in Section 5.3.4. The output L tuple IHV_{out} is computed based on the last L words $Q_{S-L+1}, \ldots, Q_S$ and IHV_{in} in Section 5.3.2.

In the computation of each word Q_i for $i = 1, \ldots, S$ a single word W_i is used that is derived from the K tuple M. Section 5.3.3 describes the mapping of M to $(W_i)_{i=0}^{S}$ called the *message expansion*. Note that the problem of endianness (Section 2.1.2) is avoided by defining M as a tuple of words instead of a bit string.

As an example, the value of the tuple (N, L, K, S) for MD4, MD5 and SHA-1 is $(32, 4, 16, 48)$, $(32, 4, 16, 64)$ and $(32, 5, 16, 80)$, respectively.

5.3.1 Working state initialization

Compress initializes the first L working state words $Q_{-L+1}, \ldots, Q_0$ using the L-tuple $IHV_{\text{in}} = (ihvin_0, \ldots, ihvin_{L-1})$:

$$Q_i = f_{\text{in},i}(ihvin_0, \ldots, ihvin_{L-1}), \quad f_{\text{in},i} \in \mathcal{F}_{\text{sumrot}},$$

for $i \in \{-L+1, \ldots, 0\}$ such that

$$(f_{\text{in},i})_{i=-L+1}^{0} : (\mathbb{Z}_{2^N})^L \to (\mathbb{Z}_{2^N})^L$$

is bijective.

In the case of MD4, MD5 and SHA-1, this initialization function $(f_{\text{in},i})_{i=-L+1}^{0}$ forms a non-trivial permutation of the L input words.

5.3.2 Finalization

After all S steps are performed, the output $IHV_{\text{out}} = (ihvout_0, \ldots, ihvout_{L-1})$ is determined as a function of the last L working state words $Q_{S-L+1}, \ldots, Q_S$ and IHV_{in}:

$$ihvout_i = f_{\text{out},i}(ihvin_0, \ldots, ihvin_{L-1}, Q_{S-L+1}, \ldots, Q_S), \quad f_{\text{out},i} \in \mathcal{F}_{\text{sumrot}},$$

for $i \in \{0, \ldots, L-1\}$. For all values of $Q_{S-L+1}, \ldots, Q_S$ we require that the following mapping is bijective:

$$(ihvin_0, \ldots, ihvin_{L-1}) \mapsto (f_{\text{out},i}(ihvin_0, \ldots, ihvin_{L-1}, Q_{S-L+1}, \ldots, Q_S))_{i=0}^{L-1}.$$

Also, for all values of $ihvin_0, \ldots, ihvin_{L-1}$ we require that the following mapping is bijective:

$$(Q_{S-L+1}, \ldots, Q_S) \mapsto (f_{\text{out},i}(ihvin_0, \ldots, ihvin_{L-1}, Q_{S-L+1}, \ldots, Q_S))_{i=0}^{L-1}.$$

The finalization of the compression functions of MD4, MD5 and SHA-1 is as follows. First the inverse initialization permutation, namely $((f_{\text{in},i})_{i=-L+1}^{0})^{-1}$, is applied

to $(Q_{S-L+1}, \ldots, Q_S)$ resulting in $(\widehat{Q}_{S-L+1}, \ldots, \widehat{Q}_S)$. The value of IHV_{out} is computed as the word-wise sum of the tuples $(\widehat{Q}_{S-L+1}, \ldots, \widehat{Q}_S)$ and $(ihvin_0, \ldots, ihvin_{L-1})$, thus $ihvout_i = \widehat{Q}_{S-L+1+i} + ihvin_i$ for $i = 0, \ldots, L-1$.

5.3.3 Message expansion

In each of the S steps $t = 0, \ldots, S-1$ a single word W_t is used that is derived from the message word tuple M:

$$W_t = f_{\text{msgexp},t}(m_0, \ldots, m_{K-1}) \in \mathbb{Z}_{2^N}.$$

The functions $f_{\text{msgexp},t}$ are arbitrary functions

$$f_{\text{msgexp},t} : (\mathbb{Z}_{2^N})^K \to \mathbb{Z}_{2^N}, \quad t \in \{0, \ldots, S-1\}$$

under the restriction that for $k = 0, K, \ldots, S-K$ the following function is bijective:

$$f_{\text{msgexpblock},k}(M) = (f_{\text{msgexp},k}(M), f_{\text{msgexp},k+1}(M), \ldots, f_{\text{msgexp},k+K-1}(M)).$$

MD4 and MD5 divide their S steps into S/K rounds, thus $48/16 = 3$ and $64/16 = 4$ rounds respectively. The first round uses the message words in order: $W_0 = m_0, \ldots, W_{15} = m_{15}$. The remaining rounds $r = 1, \ldots, \frac{S}{16} - 1$ apply a fixed permutation on the message words $(m_0, \ldots, m_{15})$ to obtain $(W_{r \cdot K}, \ldots, W_{r \cdot K 15})$. SHA-1 also uses $W_0 = m_0, \ldots, W_{15} = m_{15}$, however it computes the remaining words with the following linear relation:

$$W_i = RL(W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}, 1), \quad \text{for } i = 16, \ldots, 79.$$

This linear relation can be used backwards:

$$W_{i-16} = RR(W_i, 1) \oplus W_{i-3} \oplus W_{i-8} \oplus W_{i-14}, \quad \text{for } i = 16, \ldots, 79.$$

It follows that any 16 consecutive $W_i, \ldots, W_{i+15}$ fully determine $W_0, \ldots, W_{79}$ and in particular $W_0, \ldots, W_{15}$. This implies that for $k = 0, 16, 32, 48, 64$ the mapping from $m_0, \ldots, m_{15}$ to $W_{k+0}, \ldots, W_{k+15}$ is bijective.

5.3.4 Step function

In each of the S steps $t = 0, \ldots, S-1$, Compress computes a single boolean function over the last $L-1$ state words $Q_{t-L+2}, \ldots, Q_t$:

$$F_t = f_{\text{bool},t}(Q_{t-L+2}, \ldots, Q_t) \in \mathbb{Z}_{2^N}, \quad f_{\text{bool},t} \in \mathcal{F}_{\text{boolrot}}.$$

Each step t computes $V \in \mathbb{N}^+$ (V is fixed and independent of the step t) intermediate variables $T_{t,i}$ where $i \in \{1, \ldots, V\}$ starting with $T_{t,0} = 0$:

$$T_{t,i} \quad = \quad f_{\text{temp},t,i}(Q_{t-L+1}, \ldots, Q_t, F_t, W_t, T_{t,i-1}), \quad f_{\text{temp},t,i} \in \mathcal{F}_{\text{sumrot}}.$$

The new working state word Q_{t+1} is set to the final intermediate variable $T_{t,V}$. Furthermore, the functions $f_{\text{temp},t,i}$ have the following restrictions:

- In each function $f_{\text{temp},t,i}$, the selective sum coefficient of $T_{t,i-1}$ is non-zero.
- Over all selective sums in $(f_{\text{temp},t,i})_{i=1}^{V}$, the variable Q_{t-L+1} is selected exactly once. More formally, for $i = 1, \ldots, V$, let $c_i \in \{-1, 0, 1\}$ be the selective sum coefficient of Q_{t-L+1} in $f_{\text{temp},t,i}$ (see Section 5.2.3) then $\sum_{i=1}^{V} |c_i|$ must be 1.
- For $i = 1, \ldots, V$, let c_i' be the selective sum coefficient of W_t in $f_{\text{temp},t,i}$ then $\sum_{i=1}^{V} |c_i'|$ must be 1.
- For $i = 1, \ldots, V$, let c_i'' be the selective sum coefficient of F_t in $f_{\text{temp},t,i}$ then $\sum_{i=1}^{V} |c_i''|$ must be 1.

For MD4, $V = 2$ and functions $f_{\text{bool},t}$, $f_{\text{temp},t,1}$ and $f_{\text{temp},t,2}$ are defined as:

$$\begin{aligned} f_{\text{temp},t,1}(\ldots) &= Q_{t-3} + F_t + W_t + T_{t,0} + AC_t; \\ f_{\text{temp},t,2}(\ldots) &= RL(T_{t,1}, RC_t); \\ f_{\text{bool},t}(Q_{t-2}, Q_{t-1}, Q_t) &= \\ &\begin{cases} (Q_t \wedge Q_{t-1}) \oplus (\overline{Q_t} \wedge Q_{t-2}) & \text{for } 0 \le t < 16, \\ (Q_t \wedge Q_{t-1}) \vee (Q_t \wedge Q_{t-2}) \vee (Q_{t-1} \wedge Q_{t-2}) & \text{for } 16 \le t < 32, \\ Q_t \oplus Q_{t-1} \oplus Q_{t-2} & \text{for } 32 \le t < 48, \end{cases} \end{aligned}$$

where $AC_t \in \mathbb{Z}_{2^{32}}$ and $RC_t \in \{0, \ldots, 31\}$ are constants. For MD5, also $V = 2$ and the functions $f_{\text{bool},t}$, $f_{\text{temp},t,1}$ and $f_{\text{temp},t,2}$ are defined as:

$$\begin{aligned} f_{\text{temp},t,1}(\ldots) &= Q_{t-3} + F_t + W_t + T_{t,0} + AC_t; \\ f_{\text{temp},t,2}(\ldots) &= Q_t + RL(T_{t,1}, RC_t); \\ f_{\text{bool},t}(Q_{t-2}, Q_{t-1}, Q_t) &= \\ &\begin{cases} (Q_t \wedge Q_{t-1}) \oplus (\overline{Q_t} \wedge Q_{t-2}) & \text{for } 0 \le t < 16, \\ (Q_{t-2} \wedge Q_t) \oplus (\overline{Q_{t-2}} \wedge Q_{t-1}) & \text{for } 16 \le t < 32, \\ Q_t \oplus Q_{t-1} \oplus Q_{t-2} & \text{for } 32 \le t < 48, \\ Q_{t-1} \oplus (Q_t \vee \overline{Q_{t-2}}) & \text{for } 48 \le t < 64, \end{cases} \end{aligned}$$

where $AC_t \in \mathbb{Z}_{2^{32}}$ and $RC_t \in \{0, \ldots, 31\}$ are constants. For SHA-1, $V = 1$ and the functions $f_{\text{bool},t}$ and $f_{\text{temp},t,1}$ are defined as:

$$\begin{aligned} f_{\text{temp},t,1}(\ldots) &= RL(Q_{t-4}, 30) + RL(Q_t, 5) + F_t + W_t + T_{t,0} + AC_t; \\ f_{\text{bool},t}(\ldots) &= f_{\text{sha1},t}(Q_{t-1}, RL(Q_{t-2}, 30), RL(Q_{t-3}, 30)); \\ f_{\text{sha1},t}(X, Y, Z) &= \\ &\begin{cases} F(X,Y,Z) = Z \oplus (X \wedge (Y \oplus Z)) & \text{for } 0 \le t < 20, \\ G(X,Y,Z) = X \oplus Y \oplus Z & \text{for } 20 \le t < 40, \\ H(X,Y,Z) = (X \wedge Y) \vee (Z \wedge (X \vee Y)) & \text{for } 40 \le t < 60, \\ I(X,Y,Z) = X \oplus Y \oplus Z & \text{for } 60 \le t < 80, \end{cases} \end{aligned}$$

where $AC_t \in \mathbb{Z}_{2^{32}}$ is a constant.

5.4 Properties of the step functions

For each $t = 0, \ldots, S-1$, the mapping

$$f_t : (Q_{t-L+1}, \ldots, Q_t, F_t, W_t) \mapsto Q_{t+1}$$

as defined by the sequence $(f_{\text{temp},t,i})_{i=1}^{V}$ has three properties that are crucial to our differential path construction algorithm and are design criteria to thwart certain attacks. The first property of f_t is that the output Q_{t+1} can take on all possible values in $\mathbb{Z}_{2^N}$ by varying only $Q_{t-L+1} \in \mathbb{Z}_{2^N}$ and fixing all other input values. An important implication of this property is that Q_{t-L+1} can be uniquely determined given the output value Q_{t+1} and all other input values. The other two properties are similar to the first with respect to F_t and W_t instead of Q_{t-L+1}. These three properties are treated in detail in Section 5.4.1, 5.4.2 and 5.4.3.

5.4.1 Full dependency on Q_{t-L+1}

Theorem 5.1 (Full dependency on Q_{t-L+1})**.** *Given values of the variables Q_{t-L+2}, ..., Q_t, F_t, W_t the following mapping is bijective:*

$$f_{t,Q_{t-L+1}} : Q_{t-L+1} \mapsto Q_{t+1} = f_t(Q_{t-L+1}, \ldots, Q_t, F_t, W_t).$$

Furthermore, there exists a sequence of functions $(f_{invQ,t,i})_{i=1}^{2V+1} \in \mathcal{F}_{sumrot}^{2V+1}$ that computes the inverse of $f_{t,Q_{t-L+1}}$ given values of $Q_{t-L+2}, \ldots, Q_t$, F_t, W_t and the value of Q_{t+1}:

$$\begin{aligned} T_{invQ,t,0} &= 0; \\ T_{invQ,t,i} &= f_{invQ,t,i}(Q_{t-L+2}, \ldots, Q_t, F_t, W_t, Q_{t+1}, T_{invQ,t,0}, \ldots, T_{invQ,t,i-1}); \\ Q_{t-L+1} &= T_{invQ,t,2V+1}. \end{aligned}$$

Proof. We prove the theorem by providing a construction of a sequence of functions $(f_{\text{invQ},t,i})_{i=1}^{2V+1}$ that computes the inverse of $f_{t,Q_{t-L+1}}$. In our construction not all $f_{\text{invQ},t,i}$ may be needed in which case we define those as the zero-function. Using the second restriction in Section 5.3.4, let $j \in \{1, \ldots, V\}$ be the unique index such that the selection coefficient[15] of Q_{t-L+1} in $f_{\text{temp},t,j}$ is non-zero. Given values of $Q_{t-L+2}, \ldots, Q_t, F_t$ and W_t, we can compute the values of $T_{t,0}, \ldots, T_{t,j-1}$, since these do not depend on the value of Q_{t-L+1}. We define $f_{\text{invQ},t,1}, \ldots, f_{\text{invQ},t,j-1}$ to compute the values of $T_{t,0}, \ldots, T_{t,j-1}$:

$$f_{\text{invQ},t,i}(\ldots) = f_{\text{temp},t,i}(0, Q_{t-L+2}, \ldots, Q_t, F_t, W_t, T_{\text{invQ},t,i-1}),\ i \in \{1, \ldots, j-1\}.$$

Thus $T_{\text{invQ},t,j-1} = T_{t,j-1}$ after these steps. The functions $(f_{\text{invQ},t,i})_{i=j}^{2j-2}$ are not needed and defined as the zero function.

15. See Section 5.2.3.

For $i = j+1, \ldots, V$ the values of all inputs except T_{i-1} and Q_{t-L+1} of $f_{\text{temp},t,i}$ are known, but Q_{t-L+1} can be ignored as these functions do not depend on the value of Q_{t-L+1}. Using Observation 5.1, the following mappings are bijective:

$$g_i : T_{i-1} \mapsto f_{\text{temp},t,i}(0, Q_{t-L+2}, \ldots, Q_t, F_t, W_t, T_{i-1}), \quad i \in \{j+1, \ldots, V\}.$$

By inverting $g_V, \ldots, g_{j+1}$ we can compute the values of $T_{t,V-1}, \ldots, T_{t,j}$ in that order. First let $f_{\text{invQ},t,2j-1}(\ldots) = T_{t,V} = Q_{t+1}$. For $i = V, \ldots, j+1$, let $k = j+V-i$ and let c_i and r_i be the select coefficient and rotation constant of the variable $T_{t,i-1}$ in $f_{\text{temp},t,i}$ then we define $f_{\text{invQ},t,2(j+V-i)}$ and $f_{\text{invQ},t,2(j+V-i)+1}$ as follows:

$$\begin{aligned} f_{\text{invQ},t,2k}(\ldots) &= c_i \cdot T_{\text{invQ},t,2k-1} \\ &\quad -c_i \cdot f_{\text{temp},t,i}(0, Q_{t-L+2}, \ldots, Q_t, F_t, W_t, 0); \\ f_{\text{invQ},t,2k+1}(\ldots) &= RL(T_{\text{invQ},t,2k}, N - r_i). \end{aligned}$$

Similar to Observation 5.1, the first and second function compute the $c_j \cdot (A - C_j)$ part and the RR part, respectively, of the inverse of g_i. Given the values of $Q_{t-L+2}, \ldots, Q_t$, F_t, W_t and $T_{\text{invQ},t,2k-1} = T_{t,i}$, this results in $T_{\text{invQ},t,2k+1} = T_{t,i-1}$. Thus for $i = j+1$ this results in $T_{\text{invQ},t,2V-2} = T_j$.

Again using Observation 5.1, the following mapping is bijective:

$$g_j : Q_{t-L+1} \mapsto f_{\text{temp},t,j}(Q_{t-L+1}, \ldots, Q_t, F_t, W_t, T_{j-1}).$$

Thus it follows that $f_{t,Q_{t-L+1}}$ is bijective as

$$f_{t,Q_{t-L+1}}(Q_{t-L+1}) = g_V(\cdots(g_{j+1}(g_j(Q_{t-L+1})))\cdots).$$

Let c_Q and r_Q be the select and rotation constant of the variable Q_{t-L+1} in $f_{\text{temp},t,j}$ then we define $f_{\text{invQ},t,2V}$ and $f_{\text{invQ},t,2V+1}$ as follows:

$$\begin{aligned} f_{\text{invQ},t,2V}(\ldots) &= c_Q \cdot T_{\text{invQ},t,2V-1} \\ &\quad -c_Q \cdot f_{\text{temp},t,j}(0, Q_{t-L+2}, \ldots, Q_t, F_t, W_t, T_{\text{invQ},t,j-1}); \\ f_{\text{invQ},t,2V+1}(\ldots) &= RL(T_{\text{invQ},t,2V}, N - r_Q). \end{aligned}$$

Given the values of $Q_{t-L+2}, \ldots, Q_t$, F_t, W_t, $T_{\text{invQ},t,2V-1} = T_{t,j}$ and $T_{\text{invQ},t,j-1} = T_{t,j-1}$, this results in $T_{\text{invQ},t,2V+1} = Q_{t-L+1}$. □

Using the above theorem it follows that the following mapping is bijective and its inverse is easily computable using only additions in $\mathbb{Z}_{2^N}$ and bitwise rotations for all values of $Q_{t-L+2}, \ldots, Q_t, F_t, W_t \in \mathbb{Z}_{2^N}$:

$$f_{t,Q_{t-L+1}} : Q_{t-L+1} \mapsto f_t(Q_{t-L+1}, \ldots, Q_t, F_t, W_t).$$

5.4.2 Full dependency on F_t

Theorem 5.2 (Full dependency on F_t)**.** *Given values of the variables $Q_{t-L+1}, \ldots, Q_t$ and W_t the following mapping is bijective:*

$$f_{t,F_t} : F_t \mapsto Q_{t+1} = f_t(Q_{t-L+1}, \ldots, Q_t, F_t, W_t).$$

Furthermore, there exists a sequence of functions $(f_{invF,t,i})_{i=1}^{2V+1} \in \mathcal{F}_{sumrot}^{2V+1}$ that computes the inverse of f_{t,F_t} given values of $Q_{t-L+1}, \ldots, Q_t$, W_t and the value of Q_{t+1}:

$$\begin{aligned} T_{invF,t,0} &= 0; \\ T_{invF,t,i} &= f_{invF,t,i}(Q_{t-L+1}, \ldots, Q_t, W_t, Q_{t+1}, T_{invF,t,0}, \ldots, T_{invF,t,i-1}); \\ F_t &= T_{invF,t,2V+1}. \end{aligned}$$

The proof is analogous to the proof of Theorem 5.1.

5.4.3 Full dependency on W_t

Theorem 5.3 (Full dependency on W_t)**.** *Given values of the variables $Q_{t-L+1}, \ldots, Q_t$ and F_t the following mapping is bijective:*

$$f_{t,W_t} : W_t \mapsto Q_{t+1} = f_t(Q_{t-L+1}, \ldots, Q_t, F_t, W_t).$$

Furthermore, there exists a sequence of functions $(f_{invW,t,i})_{i=1}^{2V+1} \in \mathcal{F}_{sumrot}^{2V+1}$ that computes the inverse of f_{t,W_t} given values of $Q_{t-L+1}, \ldots, Q_t$, F_t and the value of Q_{t+1}:

$$\begin{aligned} T_{invW,t,0} &= 0; \\ T_{invW,t,i} &= f_{invW,t,i}(Q_{t-L+1}, \ldots, Q_t, W_t, Q_{t+1}, T_{invW,t,0}, \ldots, T_{invW,t,i-1}); \\ W_t &= T_{invW,t,2V+1}. \end{aligned}$$

The proof is analogous to the proof of Theorem 5.1.

5.4.4 Properties as step function design criteria

The above three properties can be seen as step function design criteria, since without these properties the compression function would be more vulnerable to attacks. Below we outline how the security of the compression function may be compromised if either one of these three properties does not hold. These design criteria may not be often pointed out, since the topic of step function design (criteria) is not treated very well in the literature. Nevertheless it should come as no surprise that these criteria hold for all members of the MD4 hash function family.

Full dependency on Q_{t-L+1}. From a hash function design perspective it is important that the mapping $f_{t,Q_{t-L+1}}$ is bijective. Suppose this mapping is not bijective, then there are values of $Q_{t-L+2}, \ldots, Q_t, F_t, W_t$ and Q_{t+1} such that inverting $f_{t,Q_{t-L+1}}$ results in multiple pre-images:

$$|\{Q_{t-L+1} \in \mathbb{Z}_{2^N} \mid f_t(Q_{t-L+1}, \ldots, Q_t, F_t, W_t) = Q_{t+1}\}| > 1.$$

Barring further complications, this leads to a pre-image attack of Compress which is about 2^{K-L} faster than the brute-force pre-image attack. The pre-image attack finds a message M given values $\widehat{IHV}_{\text{in}}$, $\widehat{IHV}_{\text{out}}$ such that $\widehat{IHV}_{\text{out}} = \text{Compress}(\widehat{IHV}_{\text{in}}, M)$ with high probability using about $2^{(N \cdot L)-(K-L)}$ calls to Compress. Below we sketch this attack.

Since finalization is bijective and based on $\mathcal{F}_{\text{sumrot}}$, it is easy to find values of $Q_{S-L+1}, \ldots, Q_S$ such that

$$\widehat{ihvout}_i = f_{\text{out},i}(\widehat{ihvin}_0, \ldots, \widehat{ihvin}_{L-1}, Q_{S-L+1}, \ldots, Q_S), \quad i \in \{0, \ldots, L-1\}.$$

Furthermore, we can expect that an attacker has a sufficiently fast way of choosing the last $K-L$ words $\widehat{W}_{S-K+L}, \ldots, \widehat{W}_{S-1}$ such that inverting the last $K-L$ steps leads to at least 2^{K-L} different pre-images $(Q_{S-K+1}, \ldots, Q_{S-K+l})$.[16]

The third property in Section 5.4.3 implies that in each step t all values of Q_{t+1} can be obtained by varying W_t. Therefore we can safely assume that on average a message M has probability $2^{(K-L)-(N \cdot L)}$ of resulting in one of the 2^{K-L} found pre-images $(Q_{S-K+1}, \ldots, Q_{S-K+l})$ during the computation of $\text{Compress}(\widehat{IHV}_{\text{in}}, M)$. By choosing only messages M such that

$$(W_{S-K+L}, \ldots, W_{S-1}) = (\widehat{W}_{S-K+L}, \ldots, \widehat{W}_{S-1}),$$

then with probability $2^{(K-L)-(N \cdot L)}$ the computation of $\text{Compress}(\widehat{IHV}_{\text{in}}, M)$ results in one of the 2^{K-L} found pre-images and thereby also results in $IHV_{\text{out}} = \widehat{IHV}_{\text{out}}$. There are in total $2^{N \cdot L}$ such messages M among which we can expect to find 2^{K-L} solutions. It follows that finding a single pre-image of Compress takes about $2^{(N \cdot L)-(K-L)}$ attempts.

Full dependency on F_t. From a hash function designers perspective, the boolean function is used as a security measure to prevent the hash function to be described as a linear system over $\mathbb{Z}_{2^N}$. If not all possible values of Q_{t+1} can be obtained by varying F_t then F_t does not fully contribute to Q_{t+1} and the security measure is not used to its full potential. An obvious attack technique is to replace the boolean function with a linear function that approximates the boolean function. In general it is easier to linearly approximate the boolean function as the influence of F_t on the output Q_{t+1} gets smaller.

16. Actually, instead of 2^{K-L} pre-images, any set of $2^Z > 2$ pre-images qualifies and here 'sufficiently fast' simply requires that finding the $\widehat{W}_i$ and the 2^Z pre-images is faster than performing $2^{(N \cdot L)-Z}$ calls to Compress.

Full dependency on W_t. Designing a hash function without having the message words W_t fully contribute in each step t implies that IHV_{out} depends in a simpler manner on the message than possible. This increases the likelihood of significant weaknesses.

In extreme cases, say only the first bit $W_t[0]$ actually contributes, this reduces the complexity of a brute force (second) pre-image attack to $\min(2^S, 2^{L \cdot N})$ calls to Compress. $S = 1/2(L \cdot N)$ for common designs like MD5 and SHA-1, thus this attack is significantly faster than the desired $2^{L \cdot N}$. In each step there are only two possible outcomes for Q_{t+1}, as in each step $t \in \{0, \ldots, S-1\}$ only a single bit of the message is used. Over S steps this means that given IHV_{in} there are at most 2^S possible $IHV_{\text{out}} = \text{Compress}(IHV_{\text{in}}, M)$ by varying M instead of the expected $2^{L \cdot N}$ possible IHV_{out}. Hence, a brute force (second) pre-image attack has a success probability of at most $\max(2^{-S}, 2^{-L \cdot N})$ for each try of M.

Rather than brute-forcing it, for given IHV_{in} and M in each step $t \in \{0, \ldots, S-1\}$ there are $2^{N-1} - 1$ other $W'_t \neq W_t$ that result in the same Q_{t+1}. Let $\mathcal{W}_{\text{sec},t}$ be the set of all W'_t that result in the same Q_{t+1} in step t. Finding a second pre-image $M' \neq M$ is thus reduced to the problem of finding $M' \neq M$ such that its message expansion $(W'_t)_{t=0}^{S-1}$ is in the set $\prod_{t=0}^{S-1} \mathcal{W}_{\text{sec},t}$. In the case of the message expansion of either MD5 or SHA-1 this would be a simple if not trivial exercise.

5.4.5 Properties in differential cryptanalysis

The reason we treat these three properties in detail is that we need them in our differential cryptanalysis. As outlined above, these three properties can be seen as step function design criteria and as such are inherent to the MD4 hash function family. Below we outline the use of these properties in our differential cryptanalysis.

Full dependency on Q_{t-L+1}. In our algorithm to construct full differential paths, we need to be able to extend partial differential paths over steps $t = A, \ldots, B$ with the preceding step $t = A - 1$. This can easily be done due to the existence of the functions $(f_{\text{invQ},t,i})_{i=1}^{2V+1} \in \mathcal{F}_{\text{sumrot}}^{2V+1}$ that compute the inverse of $f_{t,Q_{t-L+1}}$.

Full dependency on F_t. In our algorithm to construct full differential paths, we construct two partial differential paths independently, the first over steps $t = 0, \ldots, A$ and the second over steps $t = A + L + 1, \ldots, S - 1$. This implies that the difference in any Q_i is either given by the first or the second partial differential path. Due to the existence of the functions $(f_{\text{invF},t,i})_{i=1}^{2V+1} \in \mathcal{F}_{\text{sumrot}}^{2V+1}$, one can determine target differences for the boolean function outcomes $F_{A+1}, \ldots, F_{A+L}$. A full valid differential path is obtained whenever these boolean function outcome differences and the Q_i differences can be simultaneously fulfilled. Construction of a full differential path can thus be reduced to finding two partial differential paths for which these target differences in $F_{A+1}, \ldots, F_{A+L}$ can actually be obtained given the Q_i differences.

Full dependency on W_t. Although this property is not directly used in our algorithm to construct full differential paths, it is important in collision finding algorithms that search for messages M that fulfill a given differential path. This property allows one to choose working state words $Q_{t-L+1}, \ldots, Q_{t+1}$ which fulfill sufficient conditions for a given differential path and then compute the corresponding W_t.

5.5 Differential paths

In this section we formalize the concept of a differential path as a precise description of how differences propagate through the S steps of Compress $\in \mathcal{F}_{\text{md4cf}}$. These differences are taken between instances Compress(IHV_{in}, M) and Compress(IHV'_{in}, M'). For each variable $X \in \mathbb{Z}_{2^N}$ in the computation of Compress(IHV_{in}, M) we denote the respective variable in the computation of Compress(IHV'_{in}, M') as X'. Their difference is denoted as $\delta X = X' - X$ and the bitwise integer difference as $\Delta X = (X'[i] - X[i])_{i=0}^{N-1}$ which is a BSDR of δX. In the following differential cryptanalysis we refer to such δX and ΔX as variables themselves without directly implying values for X and X'.

First we analyze the only two non-trivial operations of Compress with respect to differences in $\mathbb{Z}_{2^N}$: the bitwise rotations $RL(X, n)$ in all $((f_{\text{temp},t,i})_{i=1}^{V})_{t=0}^{S-1}$ and the boolean functions $(f_{\text{bool},t})_{t=0}^{S-1}$.

5.5.1 Rotation of word differences

To determine the difference $RL(X', n) - RL(X, n)$ given δX, we distinguish two situations. First, suppose besides δX also ΔX is known. In this case

$$\begin{aligned}
\Delta RL(X, n) &= (RL(X', n)[i] - RL(X, n)[i])_{i=0}^{N-1} \\
&= (X'[(i+n) \bmod N] - X[(i+n) \bmod N])_{i=0}^{N-1} \\
&= (\Delta X[(i+n) \bmod N])_{i=0}^{N-1} \\
&= RL(\Delta X, n).
\end{aligned}$$

The remaining case is where ΔX is undetermined. In this case, up to four different $\delta Y = RL(X', n) - RL(X, n)$ are possible. Here, we determine the possible δY and their probabilities $\Pr_X[\delta Y = RL(X + \delta X, n) - RL(X, n)]$. We define $dRL(Z, n)$ for $Z \in \mathbb{Z}_{2^N}, n \in \{0, \ldots, N-1\}$ as the set of possible differences $RL(X+Z, n) - RL(X, n)$ after rotation with non-zero probability as in the following lemma.

Lemma 5.4 (Rotation of differences)**.** *Given $\delta X \in \mathbb{Z}_{2^N}$ and rotation constant $n \in \{0, \ldots, N-1\}$ then for uniformly chosen random $X \in \mathbb{Z}_{2^N}$, there are at most four possible differences $\delta Y = RL(X + \delta X, n) - RL(X, n)$ after rotation. Let $Z_{low} = \sum_{i=0}^{N-n-1} 2^i \cdot \delta X[i] \in \mathbb{Z}$ and $Z_{high} = \sum_{i=N-n}^{N-1} 2^i \cdot \delta X[i] \in \mathbb{Z}$. Then δY as above may*

attain one of four values as given below along with the corresponding probabilities:

δY		$\Pr_X[RL(X+Z,n) - RL(X,n) = \delta Y]$
$D_1 =$	$RL(\delta X, n)$	$2^{-2N+n} \cdot (2^{N-n} - Z_{low}) \cdot (2^N - Z_{high})$
$D_2 =$	$RL(\delta X, n) - 2^n$	$2^{-2N+n} \cdot (2^{N-n} - Z_{low}) \cdot Z_{high}$
$D_3 =$	$RL(\delta X, n) + 1$	$2^{-2N+n} \cdot Z_{low} \cdot (2^N - Z_{high} - 2^{N-n})$
$D_4 =$	$RL(\delta X, n) - 2^n + 1$	$2^{-2N+n} \cdot Z_{low} \cdot (Z_{high} + 2^{N-n})$

Depending on the value of δX, some of these probabilities may be zero thus reducing the number of possible outcomes, thus:

$$dRL(\delta X, n) = \left\{ D_i \;\middle|\; \Pr_X[RL(X+Z,n) - RL(X,n) = D_i] \neq 0, i \in \{1,2,3,4\} \right\}.$$

This lemma follows directly from Corollary 4.12 of Magnus Daum's Ph.D. thesis [Dau05]. We provide a different proof using BSDRs:

Proof. As above, any BSDR $(k_i)_{i=0}^{N-1}$ of δX gives rise to a candidate δY given by the BSDR

$$RL((k_i), n) = (k_{N-n-1}, \ldots, k_0, k_{N-1}, \ldots, k_{N-n}).$$

Two BSDRs $(k_i)_{i=0}^{N-1}$ and $(l_i)_{i=0}^{N-1}$ of δX result in the same δY if and only if

$$\sum_{i=0}^{N-n-1} 2^i k_i = \sum_{i=0}^{N-n-1} 2^i l_i \quad \text{and} \quad \sum_{i=N-n}^{N-1} 2^i k_i = \sum_{i=N-n}^{N-1} 2^i l_i. \tag{5.1}$$

To analyze this property, we define a *partition* as a pair $(\alpha, \beta) \in \mathbb{Z}^2$ such that $\alpha + \beta = \delta X \bmod 2^N$, $|\alpha| < 2^{N-n}$, $|\beta| < 2^N$ and $2^{N-n} | \beta$. For any partition (α, β), values $k_i \in \{-1, 0, 1\}$ for $0 \leq i < N$ can be found such that

$$\alpha = \sum_{i=0}^{N-n-1} 2^i k_i \quad \text{and} \quad \beta = \sum_{i=N-n}^{N-1} 2^i k_i. \tag{5.2}$$

In particular, there is at least one such $(k_i)_{i=0}^{N-1}$ which can be constructed by looking at the binary representation of $|\alpha|$ and $|\beta|$ and applying the sign of α and β on the respective bits. With $\alpha + \beta = \delta X \bmod 2^N$ it follows that any such $(k_i)_{i=0}^{N-1}$ is a BSDR of δX. Conversely, with Equation 5.2 any BSDR (k_i) of δX defines a partition, which we denote $(k_i) \equiv (\alpha, \beta)$. Moreover, any BSDR (k_i) of δX that leads to the same partition gives rise to the same candidate δY due to Equation 5.1. The rotation of a partition (α, β) is defined as

$$RL((\alpha, \beta), n) = 2^n \alpha + 2^{n-N} \beta \in \mathbb{Z}_{2^N}.$$

If $(k_i) \equiv (\alpha, \beta)$, this matches $RL((k_i), n)$.

The partition constraints imply there are at most two possible values for α such that $\alpha + \beta = \delta X$, namely Z_{low} and when $Z_{\text{low}} \neq 0$ also $Z_{\text{low}} - 2^{N-n}$. For each value of

α there are also at most two possible values for β namely $(\delta X - \alpha \bmod 2^N)$ and when this is non-zero also $(\delta X - \alpha \bmod 2^N) - 2^N$. Note that $Z_{\text{high}} = (\delta X - Z_{\text{low}} \bmod 2^N)$ and let $Z'_{\text{high}} = (\delta X - Z_{\text{low}} + 2^{N-n} \bmod 2^N)$. Together this gives rise to at most 4 partitions:

$$\begin{array}{lll}
\text{p1.} & (\alpha,\beta) = (Z_{\text{low}}, Z_{\text{high}}); & \\
\text{p2.} & (\alpha,\beta) = (Z_{\text{low}}, Z_{\text{high}} - 2^N), & \text{if } Z_{\text{high}} \neq 0; \\
\text{p3.} & (\alpha,\beta) = (Z_{\text{low}} - 2^{N-n}, Z'_{\text{high}}), & \text{if } Z_{\text{low}} \neq 0; \\
\text{p4.} & (\alpha,\beta) = (Z_{\text{low}} - 2^{N-n}, Z'_{\text{high}} - 2^N), & \text{if } Z_{\text{low}} \neq 0 \wedge Z'_{\text{high}} \neq 0.
\end{array}$$

Note that $RL((Z_{\text{low}}, Z_{\text{high}}), n) = RL(\delta X, n)$.

To find the probability of each δY, we define

$$p_{(\alpha,\beta)} = \Pr_X[RL((\alpha,\beta),n) = RL(X + \delta X, n) - RL(X, n)]$$

and show how $p_{(\alpha,\beta)}$ can be calculated. For each of the four possibilities this is done by counting the number of N-bit words X such that the BSDR defined by $k_i = (X+\delta X)[i] - X[i]$ satisfies $(k_i) \equiv (\alpha,\beta)$. This can be expressed in two equations:

$$\alpha = \sum_{i=0}^{N-n-1} ((X + \alpha + \beta)[i] - X[i])2^i, \quad \beta = \sum_{i=N-n}^{N-1} ((X + \alpha + \beta)[i] - X[i])2^i.$$

Since $2^{N-n} | \beta$, the β term can be ignored in the first equation. The first equation then implies that adding α to X only affects the low-order $N-n$ bits and thus α can be ignored in the second equation:

$$\alpha = \sum_{i=0}^{N-n-1} ((X + \alpha)[i] - X[i])2^i, \quad \beta = \sum_{i=N-n}^{N-1} ((X + \beta)[i] - X[i])2^i.$$

These two equations hold if and only if:

$$0 \leq \alpha + \sum_{i=0}^{N-n-1} X[i]2^i < 2^{N-n}, \quad 0 \leq \beta + \sum_{i=N-n}^{N-1} X[i]2^i < 2^N.$$

Considering the $(N-n)$ low-order bits, we determine the number r of X_{low}-values with $0 \leq X_{\text{low}} < 2^{N-n}$ such that $0 \leq \alpha + X_{\text{low}} < 2^{N-n}$: if $\alpha < 0$ then $r = 2^{N-n} + \alpha$ and if $\alpha \geq 0$ then $r = 2^{N-n} - \alpha$. Hence only $r = 2^{N-n} - |\alpha|$ out of 2^{N-n} possible X_{low}-values satisfy $0 \leq \alpha + X_{\text{low}} < 2^{N-n}$. The same argument can be used for the n high-order bits to determine that there are exactly $2^n - |\beta|2^{n-N}$ number of values for $X_{\text{high}} = 2^{n-N} \cdot \sum_{i=N-n}^{N-1} X[i]2^i$ such that $0 \leq 2^{n-N} \cdot \beta + X_{\text{high}} < 2^n$. Hence, we conclude

$$p_{(\alpha,\beta)} = \frac{2^{N-n} - |\alpha|}{2^{N-n}} \cdot \frac{2^n - |\beta|2^{n-N}}{2^n} = \frac{2^{N-n} - |\alpha|}{2^{N-n}} \cdot \frac{2^N - |\beta|}{2^N}.$$

Assume that $Z'_{\text{high}} \neq 0$, then $Z'_{\text{high}} = Z_{\text{high}} + 2^{N-n}$. One can now immediately verify that D_1, D_2, D_3 and D_4 match the rotations of the 4 partitions p1, p2, p3 and p4, respectively. Furthermore, the partition probabilities match those given in the lemma, and whenever a partition is excluded, its probability is zero.

When $Z'_{\text{high}} = 0$, then $Z_{\text{high}} = 2^N - 2^{N-n}$. Now as above, D_1 and D_2 match the rotations of partitions p1 and p2, respectively. We can show that D_4 matches the rotation of partition p3:

$$\begin{aligned} RL((Z_{\text{low}} - 2^{N-n}, Z'_{\text{high}}), n) &= 2^n \cdot (Z_{\text{low}} - 2^{N-n}) + 2^{n-N} \cdot (0) \\ &= 2^n \cdot Z_{\text{low}} + 2^{n-N} \cdot (Z_{\text{high}} - 2^N + 2^{N-n}) \\ &= 2^n \cdot Z_{\text{low}} + 2^{n-N} \cdot Z_{\text{high}} - 2^n + 1 \\ &= RL(\delta X, n) - 2^n + 1. \end{aligned}$$

And also the probability of partition p3 matches the probability of D_4 given in the lemma:

$$\begin{aligned} p_{\text{p3}} &= 2^{n-N} \cdot (2^{N-n} - (Z_{\text{low}} - 2^{N-n})) \cdot 2^{-N} \cdot (2^N - Z'_{\text{high}}) \\ &= 2^{-2N+n} \cdot Z_{\text{low}} \cdot (2^N) \\ &= 2^{-2N+n} \cdot Z_{\text{low}} \cdot (Z_{\text{high}} + 2^{N-N}). \end{aligned}$$

The remaining partition p4 is excluded and D_3 is also excluded as its probability is zero. □

5.5.2 Boolean function differences

For any step t, let $f_{\text{bool},t} \in \mathcal{F}_{\text{boolrot}}$ be given as:

$$f_{\text{bool},t}(Q_{t-L+2}, \ldots, Q_t) = \sum_{i=0}^{N-1} 2^i \cdot g(RL(Q_{t-L+2}, r_{t-L+2})[i], \ldots, RL(Q_t, r_t)[i]),$$

where $g : \{0,1\}^{L-1} \to \{0,1\}$ is an arbitrary boolean function. Since each bit of the output depends only on a single bit of each of the variables $Q_{t-L+2}, \ldots, Q_t$, for a precise differential description we need the bitwise integer difference ΔQ_i for these inputs and the output difference is determined bitwise as ΔF_t.

For $j \in \{0, \ldots, L-2\}$, define $X_j = RL(Q_{t-L+2+j}, r_{t-L+2+j})$ and thus $\Delta X_j = RL(\Delta Q_{t-L+2+j}, r_{t-L+2+j})$ to simplify notation a bit, then the difference ΔF_t can be determined using

$$\Delta F_t[i] = g(X'_0[i], \ldots, X'_{L-2}[i]) - g(X_0[i], \ldots, X_{L-2}[i]).$$

Let $i \in \{0, \ldots, N-1\}$, we consider the set U_i of possible input values:

$$U_i = \{((X'_0[i], \ldots, X'_{L-2}[i]), (X_0[i], \ldots, X_{L-2}[i])) \mid X'_j[i] = X_j[i] + \Delta X_j[i]\}.$$

This set U_i may be further restricted to $\widetilde{U}_i$ by auxiliary conditions imposed over the bits $X_j[i]$ for which $\Delta X_j[i] = 0$. (When $\Delta X_j[i] \neq 0$, $X'_j[i]$ and $X_j[i]$ are already

determined and otherwise $X'_j[i] = X_j[i]$ holds.) The cardinality of the set $\widetilde{U}_i$ indicates the amount of freedom left. If $\widetilde{U}_i = \emptyset$ then the auxiliary conditions together with ΔQ_j are contradictory and are thus of no interest.

The set $\widetilde{U}_i$ of possible input values induces a set $V_{\widetilde{U}_i}$ of possible output differences:

$$V_{\widetilde{U}_i} = \{g(X'_0[i], \ldots) - g(X_0[i], \ldots) \mid ((X'_0[i], \ldots), (X_0[i], \ldots)) \in \widetilde{U}_i\}.$$

If $|V_{\widetilde{U}_i}| = 1$ then $\Delta F_t[i] \in V_{\widetilde{U}_i}$ is uniquely determined and no auxiliary conditions are necessary. This is always the case if $\Delta X_0[i] = \ldots = \Delta X_{L-2}[i] = 0$ as then $V_{\widetilde{U}_i} = \{0\}$. Otherwise one can choose any value $\Delta\widehat{F}_t[i] \in V_{\widetilde{U}_i}$ and pose auxiliary conditions on $X_0[i], \ldots, X_{L-2}[i]$ such that the resulting new set of possible input values $\widehat{U}_i$:

- uniquely determines $\Delta F_t[i]$: $V_{\widehat{U}_i} = \{\Delta\widehat{F}_t[i]\}$.
- is as large as possible: $V_{\widetilde{U}_i \setminus \widehat{U}_i} \cap V_{\widehat{U}_i} = \emptyset$.

Once all $\Delta F_t[i]$ are uniquely determined by adding auxiliary conditions on Q_{t-L+2}, …, Q_t as necessary, also ΔF_t is completely determined. Depending on the boolean function g, some of the input variables of $Q_{t-L+2}, \ldots, Q_t$ may not affect the value of $f_{\mathrm{bool},t}(Q_{t-L+2}, \ldots, Q_t)$ and thus their respective variables $Q_j, Q'_j, \Delta Q_j$, can be ignored entirely in the above analysis.

5.5.3 Differential steps

Differential paths essentially are a sequence of *differential steps*. The message differences are chosen in some clever manner and must be given in the form of a sequence $(\mathcal{W}_t)_{t=0}^{S-1}$ of allowed message word differences $\delta W_t \in \mathcal{W}_t$ for each step t. A differential step is a precise description of how differences in the working state propagate through a single step $t \in \{0, \ldots, S-1\}$ and is defined as the tuple

$$((\delta Q_j)_{j=t-L+1}^{t+1},\ (\Delta Q_j)_{j \in I_t},\ \delta W_t,\ \Delta F_t,\ ((\delta Y_{j,i})_{i=1}^{L+3})_{j=1}^{V}),$$

where

- $I_t \subseteq \{t-L+2, \ldots, t\}$ is the set of indices j such that the input variable Q_j affects the outcome $f_{\mathrm{bool},t}(Q_{t-L+2}, \ldots, Q_t)$;
- For $j \in I_t$, ΔQ_j is a BSDR of δQ_j;
- $\delta W_t \in \mathcal{W}_t$;
- $\Delta F_t[i] \in \{g(X'_0[i], \ldots) - g(X_0[i], \ldots) \mid X'_j[i] = X_j[i] + \Delta X_j[i]\}$ where g is the underlying boolean function of $f_{\mathrm{bool},t}$ and $RL(Q_{t-L+2+j}, r_{t-L+2+j})$ is denoted by X_j as in Section 5.5.2;
- For $j \in \{1, \ldots, V\}$ and $i \in \{1, \ldots, L+3\}$, let $r_{j,i}$ and $c_{j,i}$ denote the rotation constant and selection constant associated with the i-th input variable in $f_{\mathrm{temp},t,j}$;

- The variable $Y_{j,i}$ denotes the i-th input variable of $f_{\text{temp},t,j}$ *after* the rotation as in $f_{\text{temp},t,j}$, thus for $i = 1, \ldots, L$, $Y_{j,i}$ denotes $RL(Q_{t-L+i}, r_{j,i})$. The remaining $Y_{j,L+1}$, $Y_{j,L+2}$, $Y_{j,L+3}$ denote $RL(F_t, r_{j,L+1})$, $RL(W_t, r_{j,L+2})$ and $RL(T_{t,j-1}, r_{j,L+3})$, respectively.

The intermediate variable differences $\delta T_{t,j}$ as in Section 5.3.4 are hereby determined: $\delta T_{t,0} = 0$ by definition, $\delta T_{t,j} = \sum_{i=1}^{L+3} c_{j,i} \delta Y_{j,i}$ for $j = 1, \ldots, V$. Thus together with δQ_j, δW_t and ΔF_t all differences of the inputs of $f_{\text{temp},t,j}$ and the differences $\delta Y_{j,j}$ of the rotations of these inputs are known.

A differential step is called *valid* if there exist values $(\widehat{Q}_j)_{j=t-L+1}^{t+1}$, $(\widehat{Q}'_j)_{j=t-L+1}^{t+1}$, $\widehat{W}_t$, $\widehat{W}'_t$ such that:

- $\widehat{Q}_{t+1}$ and $\widehat{Q}'_{t+1}$ are the correct outputs of the stepfunction in Section 5.3.4 for inputs $((\widehat{Q}_j)_{j=t-L+1}^{t}, \widehat{W}_t)$ and $((\widehat{Q}'_j)_{j=t-L+1}^{t}, \widehat{W}'_t)$, respectively;
- $\delta Q_j = \widehat{Q}'_j - \widehat{Q}_j$ for $j = t - L + 1, \ldots, t + 1$;
- $\Delta Q_j[i] = \widehat{Q}'_j[i] - \widehat{Q}_j[i]$ for $j \in I_t$ and $i = 0, \ldots, N - 1$;
- $\delta W_t = \widehat{W}'_t - \widehat{W}_t$;
- $\Delta F_t[i] = \widehat{F}'_t[i] - \widehat{F}_t[i]$ for $i = 0, \ldots, N - 1$, where $\widehat{F}_t = f_{\text{bool},t}(\widehat{Q}_{t-L+2}, \ldots, \widehat{Q}_t)$ and $\widehat{F}'_t = f_{\text{bool},t}(\widehat{Q}'_{t-L+2}, \ldots, \widehat{Q}'_t)$;
- $\delta Y_{j,i} = RL(\widehat{Q}'_{t-L+i}, r_{j,i}) - RL(\widehat{Q}_{t-L+i}, r_{j,i})$ for $j = 1, \ldots, V$ and $i = 1, \ldots, L$;
- $\delta Y_{j,L+1} = RL(\widehat{F}'_t, r_{j,L+1}) - RL(\widehat{F}_t, r_{j,L+1})$ for $j = 1, \ldots, V$;
- $\delta Y_{j,L+2} = RL(\widehat{W}'_t, r_{j,L+2}) - RL(\widehat{W}_t, r_{j,L+2})$ for $j = 1, \ldots, V$;
- $\delta Y_{j,L+3} = RL(\widehat{T}'_{t,j-1}, r_{j,L+3}) - RL(\widehat{T}_{t,j-1}, r_{j,L+3})$ for $j = 1, \ldots, V$, where $\widehat{T}_{t,i}$ and $\widehat{T}'_{t,i}$ for $i = 0, \ldots, V$ are computed as in Section 5.3.4;

Such values $(\widehat{Q}_j)_{j=t-L+1}^{t+1}$, $(\widehat{Q}'_j)_{j=t-L+1}^{t+1}$, $\widehat{W}_t$, $\widehat{W}'_t$ are called a solution for the differential step. This implies that for $j = 1, \ldots, V$ and each rotation in $f_{\text{temp},t,j}$, the difference $\delta Y_{j,i}$ is a possible outcome after rotation using the respective input difference: δQ_{t-L+1}, …, δQ_t, δF_t, δW_t or $\delta T_{t,j}$. Moreover this also implies that if a particular BSDR Z is given for such an input difference associated with $\delta Y_{j,i}$, then $RL(Z, r_{j,i})$ is a BSDR of $\delta Y_{j,i}$.

A partial differential path is defined as a sequence of differential steps for a subrange $t = t_{\text{begin}}, \ldots, t_{\text{end}}$, where $t_{\text{begin}}, t_{\text{end}} \in \{0, \ldots, S - 1\}$, $t_{\text{end}} \geq t_{\text{begin}}$. A partial differential path over the range $t = t_{\text{begin}}, \ldots, t_{\text{end}}$ is called *valid* if there exists a solution consisting of values

$$(\widehat{Q}_j)_{j=t_{\text{begin}}-L+1}^{t_{\text{end}}+1}, \quad (\widehat{Q}'_j)_{j=t_{\text{begin}}-L+1}^{t_{\text{end}}+1}, \quad (\widehat{W}_j)_{j=t_{\text{begin}}}^{t_{\text{end}}}, \quad (\widehat{W}'_j)_{j=t_{\text{begin}}}^{t_{\text{end}}}$$

which simultaneously provides solutions for all differential steps of the partial differential path. Such a solution does not imply a solution for the near-collision attack as $(\widehat{W}_j)_{j=t_{\text{begin}}}^{t_{\text{end}}}$ and $(\widehat{W}'_j)_{j=t_{\text{begin}}}^{t_{\text{end}}}$ with almost certainty are not part of valid message expansions for two message blocks M and M'. A (full) differential path is defined as a partial differential path over the range $t = 0, \ldots, S-1$.

5.6 Differential path construction

In this section we present the algorithm for constructing valid full differential paths for a compression function Compress $\in \mathcal{F}_{\text{md4cf}}$ based on two valid partial differential paths: $\mathcal{P}_{\text{l}}$ over steps $t = 0, \ldots, t_b$ and $\mathcal{P}_{\text{u}}$ over steps $t = t_e, \ldots, S-1$ where $t_b < t_e - L$. It will do so by independently extending $\mathcal{P}_{\text{l}}$ and $\mathcal{P}_{\text{u}}$ to $t = 0, \ldots, t_c - 1$ and $t_c + L, \ldots, S-1$, respectively, for some chosen value of t_c so they do not overlap. Actually, we construct large sets $\mathcal{E}_{\text{l}}$ and $\mathcal{E}_{\text{u}}$ of such extended differential paths. For the remaining steps $t = t_c, \ldots, t_c + L - 1$ we check for combinations $\mathcal{P}_{\text{l}} \in \mathcal{E}_{\text{l}}$, $\mathcal{P}_{\text{u}} \in \mathcal{E}_{\text{u}}$ whether a valid full differential path can be constructed.

5.6.1 Forward

For $\widetilde{t} = t_b + 1, \ldots, t_c - 1$, we extend a valid partial differential path $\mathcal{P}$ over steps $t = 0, \ldots, \widetilde{t} - 1$ with step $\widetilde{t}$. The partial differential path gives us values $(\delta Q_t)_{t=-L+1}^{\widetilde{t}}$ and $(\Delta Q_t)_{t \in I}$ where $I = \bigcup_{t=0}^{\widetilde{t}-1} I_t$. For $i \in \{\widetilde{t} - L + 1, \ldots, \widetilde{t} - 1\}$, $\mathcal{P}$ also gives us (multiple) differences after rotation of δQ_i. For each non-trivial rotation of δQ_i, which we index by $j \in J_{\mathcal{P},i}$, we denote the rotation constant as $r_{\mathcal{P},i,j}$ and the outcome difference as $\delta Y_{\mathcal{P},i,j} \in dRL(\delta Q_i, r_{\mathcal{P},i,j})$.[17] Since such values can also be determined from given IHV_{in} and IHV'_{in} we can also allow t_b to be -1.

For $i \in I_{\widetilde{t}} \setminus I$, we need to choose a BSDR ΔQ_i of δQ_i. We may choose any BSDR ΔQ_i of δQ_i such that $RL(\Delta Q_i, r_{\mathcal{P},i,j})$ is a BSDR of $\delta Y_{\mathcal{P},i,j}$ for all $j \in J_{\mathcal{P},i}$. Low weight BSDRs are the preferable choice as they in general lead to fewer sufficient conditions for the differential path.

Let X_j denote $RL(Q_{\widetilde{t}-L+2+j}, r_{\text{bool},\widetilde{t}-L+2+j})$ and g be the underlying boolean function of $f_{\text{bool},\widetilde{t}}$ as in Section 5.5.2. For $i = 0, \ldots, N-1$, the set

$$U_i = \{((X'_0[i], \ldots), (X_0[i], \ldots)) \mid X'_j[i] = X_j[i] + \Delta X_j[i]\}$$

defines all possible input bit values associated with the output bit $F_t[i]$. Previous steps restrict this set U_i further to $\widetilde{U}_i$ by allowing only those values for which there is a matching solution of $\mathcal{P}$. Choose any

$$k_i \in \left\{ g(X'_0[i], \ldots) - g(X_0[i], \ldots) \mid ((X'_0[i], \ldots), (X_0[i], \ldots)) \in \widetilde{U}_i \right\}.$$

After all k_i have been chosen, $\Delta F_t = (k_i)_{i=0}^{N-1}$ is fully determined. For $j \in \{1, \ldots, V\}$, let $\delta Y_{j,L+1} = \sigma(RL(\Delta F_t, r_{\text{temp},j,L+1}))$ where $r_{\text{temp},j,L+1}$ is the rotation constant as-

17. For dRL see Section 5.5.1.

sociated with F_t in $f_{\mathrm{temp},\widetilde{t},j}$.[18] As F_t is selected only once in total, there is at most one non-trivial case for which $r_{\mathrm{temp},j,L+1} \neq 0$.

Choose any $\delta W_{\widetilde{t}} \in \mathcal{W}_{\widetilde{t}}$. For $j \in \{1,\ldots,V\}$, let $r_{\mathrm{temp},j,L+2}$ and $c_{\mathrm{temp},j,L+2}$ be the rotation and selection constant associated with $W_{\widetilde{t}}$ in $f_{\mathrm{temp},\widetilde{t},j}$. Choose any difference $\delta Y_{j,L+2} \in dRL(\delta W_{\widetilde{t}}, r_{\mathrm{temp},j,L+2})$, preferably one with the highest probability. As W_t is selected only once in total, there is at most one non-trivial case for which $r_{\mathrm{temp},j,L+2} \neq 0$.

For $k \in \{1,\ldots,V\}$ and $i \in \{\widetilde{t}-L+1,\ldots,\widetilde{t}\}$, let $r_{\mathrm{temp},k,i-\widetilde{t}+L}$ be the rotation constant associated with Q_i in $f_{\mathrm{temp},\widetilde{t},k}$. If $i \notin (I_{\widetilde{t}} \cup I)$ then ΔQ_i is not yet chosen, we need to choose V differences

$$\delta Y_{k,(i-\widetilde{t}+L)} \in dRL(\delta Q_i, r_{\mathrm{temp},k,i-\widetilde{t}+L}).$$

Consider the set of BSDRs Z of δQ_i such that $\sigma(RL(Z, r_{P,i,j})) = \delta Y_{P,i,j}$ for all $j \in J_{P,i}$. Each such BSDR Z leads directly to values

$$(\delta Y_{k,i-\widetilde{t}+L})_{k=1}^{V} = (\sigma(RL(Z, r_{\mathrm{temp},k,i-\widetilde{t}+L})))_{k=1}^{V}.$$

One can choose any such values $(\delta Y_{k,i-\widetilde{t}+L})_{k=1}^{V}$, the preferable choice is one which results from the largest number of possible Z-values. Otherwise, if $i \in (I_{\widetilde{t}} \cup I)$ let $\delta Y_{k,i-\widetilde{t}+L} = \sigma(RL(\Delta Q_i, r_{\mathrm{temp},k,i-\widetilde{t}+L}))$.

The differential step $t = \widetilde{t}$ is now easily determined. By definition $\delta T_{\widetilde{t},0} = 0$. For $i = 1,\ldots,V$, choose a $\delta Y_{i,L+3} \in dRL(\delta T_{\widetilde{t},i-1}, r_{\mathrm{temp},i,L+3})$ with the highest probability where $r_{\mathrm{temp},i,L+3}$ is the rotation constant associated with $T_{\widetilde{t},i-1}$ in $f_{\mathrm{temp},\widetilde{t},i}$. Note that:

$$\begin{aligned}
\delta T_{\widetilde{t},i} &= f_{\mathrm{temp},\widetilde{t},i}(Q'_{\widetilde{t}-L+1},\ldots,Q'_{\widetilde{t}},F'_{\widetilde{t}},W'_{\widetilde{t}},T'_{\widetilde{t},i-1}) \\
&\quad -f_{\mathrm{temp},\widetilde{t},i}(Q_{\widetilde{t}-L+1},\ldots,Q_{\widetilde{t}},F_{\widetilde{t}},W_{\widetilde{t}},T_{\widetilde{t},i-1}) \\
&= \sum_{j=1}^{L} c_{i,j}(RL(Q'_{\widetilde{t}-L+j}, r_{\mathrm{temp},i,j}) - RL(Q_{\widetilde{t}-L+j}, r_{\mathrm{temp},i,j})) \\
&\quad +c_{i,L+1}(RL(F'_{\widetilde{t}}, r_{\mathrm{temp},i,L+1}) - RL(F_{\widetilde{t}}, r_{\mathrm{temp},i,L+1})) \\
&\quad +c_{i,L+2}(RL(W'_{\widetilde{t}}, r_{\mathrm{temp},i,L+2}) - RL(W_{\widetilde{t}}, r_{\mathrm{temp},i,L+2})) \\
&\quad +c_{i,L+3}(RL(T'_{\widetilde{t},i-1}, r_{\mathrm{temp},i,L+3}) - RL(T_{\widetilde{t},i-1}, r_{\mathrm{temp},i,L+3})) \\
&= \sum_{j=1}^{L+3} c_{i,j}\delta Y_{i,j}
\end{aligned}$$

All $\delta Y_{i,j}$ have been determined already. Thus $\delta T_{\widetilde{t},i}$ is hereby determined and for $i = V$ also $\delta Q_{\widetilde{t}+1}$.

The extended differential path $\widetilde{\mathcal{P}}$ consists of $\mathcal{P}$ and the differential step $t = \widetilde{t}$. If there exists no solution of $\widetilde{\mathcal{P}}$ then the extended differential path is not valid and of no further interest. We collect many valid differential paths $\widetilde{\mathcal{P}}$ by varying the

18. For σ see Section 5.2.2.

choices made and for different input differential paths $\mathcal{P}$. As this set can theoretically grow exponentially over subsequent steps $\widetilde{t}$, keep only the R "best" differential paths for some feasibly large R. Here "best" should be tailored toward the near-collision attack construction, which implies a low number of sufficient conditions for the partial differential path and a large degree of freedom for message modification techniques.

5.6.2 Backward

For $\widetilde{t} = t_e - 1, \ldots, t_c + L$, we extend a valid partial differential path $\mathcal{P}$ over steps $t = \widetilde{t}+1, \ldots, S-1$ with step $\widetilde{t}$. The partial differential path gives us values $(\delta Q_t)_{t=\widetilde{t}-L+2}^{S}$, $(\Delta Q_t)_{t\in I}$ where $I = \bigcup_{t=\widetilde{t}+1}^{S-1} I_t$. For $i \in \{\widetilde{t} - L + 2, \ldots, \widetilde{t} + 1\}$, the differential path $\mathcal{P}$ also gives us (multiple) differences after rotation of δQ_i. For each non-trivial rotation of δQ_i, which we index by $j \in J_{\mathcal{P},i}$, we denote the rotation constant as $r_{\mathcal{P},i,j}$ and the outcome difference as $\delta Y_{\mathcal{P},i,j} \in dRL(\delta Q_i, r_{\mathcal{P},i,j})$. The following steps are basically the same as in Section 5.6.1 except initially $(f_{\mathrm{invQ},\widetilde{t},k})_{k=1}^{2V+1}$ (as constructed in Section 5.4.1) are used instead of $(f_{\mathrm{temp},\widetilde{t},k})_{k=1}^{V}$. Only at the end the results are translated to a differential step over $(f_{\mathrm{temp},\widetilde{t},k})_{k=1}^{V}$.

For $i \in I_{\widetilde{t}} \setminus I$, we need to choose a BSDR ΔQ_i of δQ_i. We may choose any BSDR ΔQ_i of δQ_i such that $RL(\Delta Q_i, r_{\mathcal{P},i,j})$ is a BSDR of $\delta Y_{\mathcal{P},i,j}$ for all $j \in J_{\mathcal{P},i}$. As before, low weight BSDRs are the preferable choice as they in general lead to fewer sufficient conditions for the differential path.

Let X_j denote $RL(Q_{\widetilde{t}-L+2+j}, r_{\mathrm{bool},\widetilde{t}-L+2+j})$ and g be the underlying boolean function of $f_{\mathrm{bool},\widetilde{t}}$ as in Section 5.5.2. For $i = 0, \ldots, N-1$, the set

$$U_i = \{((X'_0[i], \ldots), (X_0[i], \ldots)) \mid X'_j[i] = X_j[i] + \Delta X_j[i]\}$$

defines all possible input bit values associated with the output bit $F_t[i]$. The known differential steps $t = \widetilde{t}+1, \ldots, S-1$ restrict this set U_i further to $\widetilde{U}_i$ by allowing only those values for which there is a matching solution of $\mathcal{P}$. Choose any

$$k_i \in \left\{ g(X'_0[i], \ldots) - g(X_0[i], \ldots) \mid ((X'_0[i], \ldots), (X_0[i], \ldots)) \in \widetilde{U}_i \right\}.$$

After $k_0, \ldots, k_{N-1}$ have been chosen, $\Delta F_t = (k_i)_{i=0}^{N-1}$ is fully determined. For $j \in \{1, \ldots, 2V+1\}$, let $\delta Y_{j,L+1} = \sigma(RL(\Delta F_t, r_{\mathrm{invQ},j,L+1}))$ where $r_{\mathrm{invQ},j,L+1}$ is the rotation constant associated with F_t in $f_{\mathrm{invQ},\widetilde{t},j}$.

Choose any $\delta W_{\widetilde{t}} \in \mathcal{W}_{\widetilde{t}}$. For $k \in \{1, \ldots, 2V+1\}$, let $r_{\mathrm{invQ},k,L+2}$ and $c_{\mathrm{invQ},k,L+2}$ be the rotation and selection constant associated with $W_{\widetilde{t}}$ in $f_{\mathrm{invQ},\widetilde{t},k}$. Choose any difference $\delta Y_{k,L+2} \in dRL(\delta W_{\widetilde{t}}, r_{\mathrm{invQ},k,L+2})$, preferably one with the highest probability.

For $k \in \{1, \ldots, 2V+1\}$ and $i \in \{\widetilde{t} - L + 2, \ldots, \widetilde{t} + 1\}$, let $r_{\mathrm{invQ},k,(i-\widetilde{t}+L-1)}$ be the rotation constant associated with Q_i in $f_{\mathrm{invQ},\widetilde{t},k}$. If $i \notin (I_{\widetilde{t}} \cup I)$ then ΔQ_i is not yet chosen, we need to choose $2V+1$ differences

$$\delta Y_{k,(i-\widetilde{t}+L-1)} \in dRL(\delta Q_i, r_{\mathrm{invQ},k,(i-\widetilde{t}+L-1)}).$$

Consider the set of BSDRs Z of δQ_i such that $\sigma(RL(Z, r_{\mathcal{P},i,j})) = \delta Y_{\mathcal{P},i,j}$ for all $j \in J_{\mathcal{P},i}$. Each such BSDR Z leads directly to values

$$(\delta Y_{k,(i-\widetilde{t}+L-1)})_{k=1}^{2V+1} = (\sigma(RL(Z, r_{\mathrm{invQ},k,(i-\widetilde{t}+L-1)})))_{k=1}^{2V+1}.$$

One can choose any such values $(\delta Y_{k,(i-\widetilde{t}+L-1)})_{k=1}^{2V+1}$, the preferable choice is one which results from the largest number of possible Z-values. Otherwise, if $i \in (I_{\widetilde{t}} \cup I)$ let $\delta Y_{k,(i-\widetilde{t}+L-1)} = \sigma(RL(\Delta Q_i, r_{\mathrm{invQ},k,(i-\widetilde{t}+L-1)}))$.

By definition $\delta T_{\mathrm{invQ},\widetilde{t},0} = 0$. For $i = 1, \ldots, 2V+1$ and $j = 0, \ldots, i-1$, let $r_{\mathrm{T},i,j}$ be the rotation constant associated with $T_{\mathrm{invQ},\widetilde{t},j}$ in $f_{\mathrm{invQ},\widetilde{t},i}$. Choose a most probable $\delta Y_{i,L+3+j} \in dRL(\delta T_{\mathrm{invQ},\widetilde{t},j}, r_{\mathrm{T},i,j})$. Note that:

$$\begin{aligned}
\delta T_{\mathrm{invQ},\widetilde{t},i} &= \sum_{j=1}^{L} c_{i,j}(RL(Q'_{\widetilde{t}-L+j+1}, r_{\mathrm{invQ},i,j}) - RL(Q_{\widetilde{t}-L+j+1}, r_{\mathrm{invQ},i,j})) \\
&\quad + c_{i,L+1}(RL(F'_{\widetilde{t}}, r_{\mathrm{invQ},i,L+1}) - RL(F_{\widetilde{t}}, r_{\mathrm{invQ},i,L+1})) \\
&\quad + c_{i,L+2}(RL(W'_{\widetilde{t}}, r_{\mathrm{invQ},i,L+2}) - RL(W_{\widetilde{t}}, r_{\mathrm{invQ},i,L+2})) \\
&\quad + \sum_{j=0}^{i-1} c_{i,L+3+j}(RL(T'_{\mathrm{invQ},\widetilde{t},j}, r_{\mathrm{T},i,j}) - RL(T_{\mathrm{invQ},\widetilde{t},j}, r_{\mathrm{T},i,j})) \\
&= \sum_{j=1}^{L+2+i} c_{i,j}\delta Y_{i,j}.
\end{aligned}$$

All $\delta Y_{i,j}$ have been determined already. Thus $\delta T_{\mathrm{invQ}\widetilde{t},i}$ is hereby determined and for $i = 2V+1$ also $\delta Q_{\widetilde{t}-L+1}$.

Most of the information for the differential step $t = \widetilde{t}$ is a direct result of the above steps: the values $(\delta Q_j)_{j=\widetilde{t}-L+1}^{\widetilde{t}+1}$, $(\Delta Q_j)_{j \in I_t}$, δW_t and ΔF_t. It remains to determine the $\delta \widetilde{Y}_{i,j}$ corresponding to differences after rotation of the inputs of $f_{\mathrm{temp},t,i}$. In the proof of Theorem 5.1 the original inputs of $f_{\mathrm{temp},t,j}$ are used in the $f_{\mathrm{invQ},t,i}$ in one of two ways. In the first way they are used as a direct input to a call of $f_{\mathrm{temp},t,j}$ within $f_{\mathrm{invQ},t,i}$. In the second way, they form the outcome of some $f_{\mathrm{invQ},t,i}$ that is used together with $f_{\mathrm{invQ},t,i-1}$ to invert $f_{\mathrm{temp},t,j}$. Thus all differences $\delta \widetilde{Y}_{i,j}$ after rotation of the inputs of $f_{\mathrm{temp},t,i}$ are already determined above.

The extended differential path $\widetilde{\mathcal{P}}$ consists of $\mathcal{P}$ and the differential step $t = \widetilde{t}$. If there exists no solution of $\widetilde{\mathcal{P}}$ then the extended differential path is not valid and of no further interest. Similar to extending forward, we collect many valid differential paths $\widetilde{\mathcal{P}}$ by varying the choices made and for different input differential paths $\mathcal{P}$. As this set can theoretically grow exponentially over subsequent steps $\widetilde{t}$, keep only the R "best" differential paths for some feasibly large R. Here "best" differential paths should be read as the differential paths with the highest success probability.

5.6.3 Connect

In the final stage, we try many combinations of lower valid differential paths $\mathcal{P}_\mathrm{l}$ over steps $t = 0, \ldots, t_c - 1$ and upper valid differential paths $\mathcal{P}_\mathrm{u}$ over steps $t_c + L, \ldots, S - 1$. For each such combination we have differential steps $t = 0, \ldots, t_c - 1, t_c + L, \ldots, S - 1$. This means that all $(\delta Q_t)_{t=-L+1}^{S}$ are known either from $\mathcal{P}_\mathrm{l}$ or $\mathcal{P}_\mathrm{u}$, there is no overlap. Together $\mathcal{P}_\mathrm{l}$ and $\mathcal{P}_\mathrm{u}$ also provide values $(\Delta Q_t)_{t \in I}$ where $I = \bigcup_{t=t_c+L}^{S-1} I_t \cup \bigcup_{t=0}^{t_c-1} I_t$. We use $((f_{\mathrm{invF},j,i})_{i=1}^{2V+1})_{j=t_c}^{t_c+L-1}$ to determine target differences $\delta F_{t_c}, \ldots, \delta F_{t_c+L-1}$. By using the remaining freedom in yet undetermined BSDRs $\Delta F_{t_c}, \ldots, \Delta F_{t_c+L-1}$, we try to achieve these target differences.

For $\widetilde{t} = t_c, \ldots, t_c + L - 1$, we do the following and exhaustively try all possible choices. For $i \in I_{\widetilde{t}} \setminus (I \cup \bigcup_{t=t_c}^{\widetilde{t}-1} I_t)$, we need to choose a BSDR ΔQ_i of δQ_i. For $i \in \{\widetilde{t} - L + 1, \ldots, \widetilde{t} + 1\}$, the preceding differential steps $t = t_c, \ldots, \widetilde{t} - 1$ together with $\mathcal{P}_\mathrm{l}$ and $\mathcal{P}_\mathrm{u}$ give us (multiple) differences after rotation of δQ_i. For each non-trivial rotation of δQ_i, which we index by $j \in J_{\mathcal{P},i}$, we denote the rotation constant as $r_{\mathcal{P},i,j}$ and the outcome difference as $\delta Y_{\mathcal{P},i,j} \in dRL(\delta Q_i, r_{\mathcal{P},i,j})$. Choose any BSDR ΔQ_i of δQ_i such that $RL(\Delta Q_i, r_{\mathcal{P},i,j})$ is a BSDR of $\delta Y_{\mathcal{P},i,j}$ for all $j \in J_{\mathcal{P},i}$.

Choose any $\delta W_{\widetilde{t}} \in \mathcal{W}_{\widetilde{t}}$. For $k \in \{1, \ldots, 2V + 1\}$, let $r_{\mathrm{invF},k,L+2}$ and $c_{\mathrm{invF},k,L+2}$ be the rotation and selection constant associated with $W_{\widetilde{t}}$ in $f_{\mathrm{invF},\widetilde{t},k}$. Choose any difference $\delta Y_{k,L+2} \in dRL(\delta W_{\widetilde{t}}, r_{\mathrm{invF},k,L+2})$.

For $k \in \{1, \ldots, 2V + 1\}$ and $i \in \{\widetilde{t} - L + 1, \ldots, \widetilde{t} + 1\}$, let $r_{\mathrm{invF},k,(i-\widetilde{t}+L)}$ be the rotation constant associated with Q_i in $f_{\mathrm{invF},\widetilde{t},k}$. If $i \notin (I \cup \bigcup_{t=t_c}^{\widetilde{t}} I_t)$ then ΔQ_i is not yet chosen and we need to choose $2V + 1$ differences

$$\delta Y_{k,(i-\widetilde{t}+L)} \in dRL(\delta Q_i, r_{\mathrm{invF},k,(i-\widetilde{t}+L)}).$$

Consider the set of BSDRs Z of δQ_i such that $\sigma(RL(Z, r_{P,i,j})) = \delta Y_{P,i,j}$ for all $j \in J_{P,i}$. Each such BSDR Z leads directly to values

$$(\delta Y_{k,(i-\widetilde{t}+L)})_{k=1}^{2V+1} = (\sigma(RL(Z, r_{\mathrm{invF},k,(i-\widetilde{t}+L)})))_{k=1}^{2V+1}.$$

Choose any such values $(\delta Y_{k,(i-\widetilde{t}+L)})_{k=1}^{2V+1}$. Otherwise, if $i \in (I \cup \bigcup_{t=t_c}^{\widetilde{t}} I_t)$ then let $\delta Y_{k,(i-\widetilde{t}+L)} = \sigma(RL(\Delta Q_i, r_{\mathrm{invF},k,(i-\widetilde{t}+L)}))$.

By definition $\delta T_{\mathrm{invF},\widetilde{t},0} = 0$. For $i = 1, \ldots, 2V + 1$ and $j = 0, \ldots, i - 1$, let $r_{\mathrm{T},i,j}$ be the rotation constant associated with $T_{\mathrm{invF},\widetilde{t},j}$ in $f_{\mathrm{invF},\widetilde{t},i}$. Choose any $\delta Y_{i,L+3+j} \in dRL(\delta T_{\mathrm{invF},\widetilde{t},j}, r_{\mathrm{T},i,j})$, here we do not limit the choices to high probability values to increase the success probability of our connection algorithm. This can be compensated by searching for many valid full differential paths and choosing the one most suitable

for a near-collision attack. Note that:

$$\begin{aligned}\delta T_{\mathrm{invF},\widetilde{t},i} &= \sum_{j=1}^{L+1} c_{i,j}(RL(Q'_{\widetilde{t}-L+j}, r_{\mathrm{invF},i,j}) - RL(Q_{\widetilde{t}-L+j}, r_{\mathrm{invF},i,j})) \\ &\quad + c_{i,L+2}(RL(W'_{\widetilde{t}}, r_{\mathrm{invF},i,L+2}) - RL(W_{\widetilde{t}}, r_{\mathrm{invF},i,L+2})) \\ &\quad + \sum_{j=0}^{i-1} c_{i,L+3+j}(RL(T'_{\mathrm{invF},\widetilde{t},j}, r_{\mathrm{T},i,j}) - RL(T_{\mathrm{invF},\widetilde{t},j}, r_{\mathrm{T},i,j})) \\ &= \sum_{j=1}^{L+2+i} c_{i,j}\delta Y_{i,j}.\end{aligned}$$

All $\delta Y_{i,j}$ have been determined already. Thus $\delta T_{\mathrm{invF}\widetilde{t},i}$ is hereby determined and for $i = 2V + 1$ also $\delta F_{\widetilde{t}}$.

Let X_j denote $RL(Q_{\widetilde{t}-L+2+j}, r_{\mathrm{bool},\widetilde{t}-L+2+j})$ and g be the underlying boolean function of $f_{\mathrm{bool},\widetilde{t}}$ as in Section 5.5.2. For $i = 0, \ldots, N-1$, the set

$$U_i = \{((X'_0[i], \ldots), (X_0[i], \ldots)) \mid X'_j[i] = X_j[i] + \Delta X_j[i]\}$$

defines all possible input bit values associated with the output bit $F_t[i]$. The known differential steps restrict this set U_i further to $\widetilde{U}_i$ by allowing only those values for which there is a matching solution of those differential steps. Let

$$V_i = \left\{ g(X'_0[i], \ldots) - g(X_0[i], \ldots) \mid ((X'_0[i], \ldots), (X_0[i], \ldots)) \in \widetilde{U}_i \right\}.$$

Now it remains to verify whether $\delta F_{\widetilde{t}}$ can be achieved by one of the possible BSDRs Z such that $Z[i] \in V_i$ for $i = 0, \ldots, N-1$.

To this end we desire to construct sets

$$\mathcal{Z}_i = \left\{ (k_j)_{j=0}^{i} \;\middle|\; k_j \in V_j \wedge \left(\sum_{j=0}^{i} k_j 2^j \equiv \delta F_{\widetilde{t}} \bmod 2^{i+1} \right) \right\}, \quad i \in \{0, \ldots, N-1\}.$$

It follows that $\mathcal{Z}_0 = \{(k_0) \mid k_0 \in V_0 \wedge (k_0 \equiv \delta F_{\widetilde{t}} \bmod 2)\}$. For $i = 1, \ldots, N-1$, we construct $\mathcal{Z}_i$ as

$$\mathcal{Z}_i = \left\{ (k_j)_{j=0}^{i} \;\middle|\; (k_j)_{j=0}^{i-1} \in \mathcal{Z}_{i-1} \wedge k_i \in V_i \wedge \left(\sum_{j=0}^{i} k_j 2^j \equiv \delta F_{\widetilde{t}} \bmod 2^{i+1} \right) \right\}.$$

If $\mathcal{Z}_i = \emptyset$ then $\delta F_{\widetilde{t}}$ cannot be achieved and other choices above have to be tried. If all choices above have been exhausted then we try a untried combination of $\mathcal{P}_{\mathrm{l}}$ and $\mathcal{P}_{\mathrm{u}}$. Otherwise, $\delta F_{\widetilde{t}}$ is achieved by any of the BSDRs $Z \in \mathcal{Z}_{N-1}$. Choose any $\Delta F_{\widetilde{t}} \in \mathcal{Z}_{N-1}$.

Most of the information for the differential step $t = \widetilde{t}$ is a direct result of the above steps: the values $(\delta Q_j)_{j=\widetilde{t}-L+1}^{\widetilde{t}+1}$, $(\Delta Q_j)_{j \in I_t}$, δW_t and ΔF_t. It remains to determine

the $\delta\widetilde{Y}_{i,j}$ corresponding to differences after rotation of the inputs of $f_{\text{temp},t,i}$. As the proof of Theorem 5.2 is analogous to that of Theorem 5.1, the original inputs of $f_{\text{temp},t,j}$ are used in the $f_{\text{invF},t,i}$ in one of two ways. In the first way they are used as a direct input to a call of $f_{\text{temp},t,j}$ within $f_{\text{invF},t,i}$. In the second way, they form the outcome of some $f_{\text{invQ},t,i}$ that is used together with $f_{\text{invQ},t,i-1}$ to invert $f_{\text{temp},t,j}$.

Thus all differences $\delta\widetilde{Y}_{i,j}$ after rotation of the inputs of $f_{\text{temp},t,i}$ are already determined above. It remains to verify whether there exists a simultaneous solution for $\mathcal{P}_\text{l}$, $\mathcal{P}_\text{u}$ and the differential steps $t = t_c, \ldots, \widetilde{t}$. If there is no such solution then this differential step is of no further interest and we try different choices above.

We can now try the next remaining differential step until $\widetilde{t} = t_c + L - 1$ in which case we have a full valid differential path.

5.6.4 Complexity

The complexity of the above procedure is of great interest, however it depends on many factors which are still undecided. It is common in attacks against hash functions to describe the complexity as the equivalent runtime cost in compression function calls. However, the parameters N, L, K and S, and the initialization, finalization, message expansion and step functions of the compression function remain to be chosen, thereby making a thorough complexity analysis infeasible. Nevertheless, below we comment on several significant factors that contribute to the complexity of the above differential path construction.

Connect search. The connect search tries combinations of lower and upper partial differential paths. The average runtime cost for each combination of lower and upper partial differential paths depends among others on the degrees of freedom. Less freedom implies a lower average runtime cost for each combination, however the average success probability per combination also decreases. The expected number of combinations to try is determined by the average success probability multiplied by the desired number of valid full differential paths. This desired number of valid full differential paths can be one, but often a lot more than one are desired as this leaves additional freedom when implementing a collision attack.

The degrees of freedom can mostly be found in the number of allowed BSDRs ΔQ_i for $i \in \{t_c - L + 2, \ldots, t_c + L\} \setminus \bigcup_{t=t_c}^{t_c+L-1} I_t$ and the average size of the boolean function output bit difference sets $\widetilde{U}_j$. At least two ΔQ_i are not uniquely determined yet: $i = t_c$ and $i = t_c + 1$. On average, a higher weight $\text{NAF}(\delta Q_i)$ results in a larger number of allowed BSDRs ΔQ_i. Therefore a high weight $\text{NAF}(\delta Q_i)$ for these values of i is beneficiary to the connect search, which is in contrast with the desire to obtain differential paths with as few as possible number of bitconditions.

Forward and backward search. The forward and backward search have very similar complexity characteristics. The complexity of the forward and backward search for each step consists of the following factors:

- number of input partial differential paths: by keeping only the R "best" partial differential paths this factor is upper bounded by R. Let R_{f} and R_{b} be this upper bound for the forward and backward search, respectively. It follows that $R_{\mathrm{f}} \cdot R_{\mathrm{b}}$ must be chosen large enough such that the desired number of valid full differential paths in the connect search may be obtained.

- average number of BSDRs $\Delta Q_{\tilde{t}}$ of $\delta Q_{\tilde{t}}$: this is lower bounded by $2^{w(\mathrm{NAF}(\delta Q_{\tilde{t}}))}$. Since low weight BSDRs are preferred, the set of allowed BSDRs $\Delta Q_{\tilde{t}}$ can be bounded for instance by taking only the B lowest weight BSDRs, limiting the maximum weight $w(\Delta Q_{\tilde{t}}) \leq B$, limiting the maximum offset weight $w(\Delta Q_{\tilde{t}}) \leq w(\mathrm{NAF}(\delta Q_{\tilde{t}})) + B$ or any combination thereof.

- average size κ of the possible boolean function output bit difference sets $\widetilde{U}_j$ resulting in a total factor of κ^N.

- size of the set of possible message word differences $\mathcal{W}_{\tilde{t}}$.

5.6.5 Specializations

In Chapters 6 and 7 we provide specializations of the differential cryptanalysis and differential path construction for MD5 and SHA-1. In particular, we use the concept of bitconditions to describe the differential path. These bitconditions also provide a means to efficiently determine the set of possible boolean function output bit differences and the additional bitconditions necessary to enforce the chosen boolean function output bit difference. Moreover, we improve the connect search such that it deals with the to be determined BSDRs ΔQ_i and ΔF_j in a per-digit manner instead of a per-BSDR manner.