Ways to restrict the differential path

ZiJie Xu and Ke Xu
xuzijiewz@gmail.com
xukezp@gmail.com

Abstract: People had developed some attack methods to attack hash function. These methods need to choose some "differential pattern"[Dau05]. We present a way to restrict the collisions that hold the "differential pattern". At the same time, to build a hash function that meet the different needs, we propose a construction.

Key Words: differential path, differential cryptanalysis, hash function, data-depend function

1. Introduction

Magnus Daum[Dao05] had presentation hash function and the attack methods Dobbertin's method[Dob95, Dob96a, Dob96c, Dob97, Dob98a], Chabaud and Joux's method [CJ98] and Wang's method[WYY05]. These attack methods include two parts[Dao05].

One part is choosing "differential pattern" [Dao05]. The work in this part is use modular difference, xor-difference and bitwise deference define all differences in the calculation. Dobbertin's method uses mainly modular divergences, Chabaud and Joux's method uses xor-difference, and Wang's method uses modular divergences, xor-difference and bitwise deference.

In Wang's method, when "differential pattern" is determined, it can build sufficient conditions[EUROCRYPT'05] that hold the "differential pattern". The collision must satisfy the sufficient conditions. In a way, the sufficient conditions astrict the messages that will hold "differential pattern". But the sufficient conditions is for the chain values. In different attack, the messages that conform the "differential pattern" maybe different. If there is some measures can be used to astrict the messages, then it can control the collision that attacker can get. We will present ways to astrict the conditions in section 2 and section 3.

Because the hash function is a map that compress arbitrary long messages to fixed length values[Dao05]. So there always exists collisions, there always exists viable

"differential pattern". M. Schläffer and E. Oswald[1], P.-A. Fouque, G. Leurent, and P. Nguyen[1] had developed an algorithm to construct the differential path. The algorithm will determine the defference at every step operation in order. It can think of the algorithm as a search technology. The complexity of the algorithm depends on the size of difference space. If the calculation has more step operation, it will raise the size of difference space. But it will raise the calculation at the same time. We find a construction. With the construction, it can build hash function that user can regulate the number of the step operations to meet his requirement.

We will expound the ways that can be used restrict the differential path in section 2 and section 3. And then we will expound the construction in section 4.

We will discuss the shift of n-bits word, where $n = 2^l$.

2. Modular Difference And Bitwise Deference

The modular difference is the difference of x, x' that are elements of the ring of integers modulo 2^n as[Dao05]:

$$\Delta^+ x = x - x'$$

And the bitwise deference is bitstrings difference of x, x' that are elements of the vector space F_2^n as[Dao05]:

$$\Delta^\pm x := \Delta^\pm(x, x') := (x_{n-1} - x'_{n-1}, x_{n-2} - x'_{n-2}, ..., x_0 - x'_0) := (\Delta^\pm x_{n-1}, \Delta^\pm x_{n-2}, ..., \Delta^\pm x_0)$$

Where $\Delta^\pm x_i \in \{-1,0,1\}$.

Because there exists:

$$\begin{cases} x = \sum_{i=0}^{n-1} x_i \times 2^i \\ x' = \sum_{i=0}^{n-1} x'_i \times 2^i \end{cases}$$

So there exists:

$$\Delta^+ x = x - x' = \sum_{i=0}^{n-1} x_i \times 2^i - \sum_{i=0}^{n-1} x'_i \times 2^i = \sum_{i=0}^{n-1} (x_i - x'_i) \times 2^i = \sum_{i=0}^{n-1} \Delta^\pm x_i \times 2^i \qquad (2.1)$$

To given $\Delta^+ x$, (2.1) maybe has many roots. That means there maybe are many bitwise differences has same modular difference.

We find a way to strict the number of the bitwise differences that has same modular difference. Let constant $C = \sum_{i=0}^{n/2-1} 2^{i\times 2} = 0x01...01...01$. And let hash function has two variable x and x1. And x and x1 satisfy:

$$x1 = x \wedge \sum_{i=0}^{n/2-1} 2^{i\times 2} \qquad (2.2)$$

Then in the "differential pattern", there will be two differences as follow:

$$\begin{cases} \Delta^+ x = x - x' = \sum_{i=0}^{n-1} \Delta^{\pm} x_i \times 2^i \\ \Delta^+ x1 = x1 - x1' = x \wedge \sum_{i=0}^{n/2-1} 2^{i\times 2} - x' \wedge \sum_{i=0}^{n/2-1} 2^{i\times 2} \end{cases} \tag{2.3}$$

There exists:

$$\begin{aligned} \Delta^+ x1 &= x1 - x1' \\ &= x \wedge C - x' \wedge C \\ &= \sum_{i=0}^{n-1} \Delta^{\pm} x1_i \times 2^i \\ &= \sum_{i=0}^{n/2-1} \Delta^{\pm} x_{2\times i} \times 2^{i\times 2} \end{aligned}$$

There is a proposition about the reration between modular differences ($\Delta^+ x$, $\Delta^+ x1$) and bitwise deference $\Delta^{\pm} x$ as follows:

Proposition 2.1: There is sole one bitwise deference $\Delta^{\pm} x$ has the given modular differences ($\Delta^+ x$, $\Delta^+ x1$) that satisfy (2.3) and $\Delta^+ x \neq 0$.

Proof: It can compute the bitwise difference $\Delta^{\pm} x$ from the given modular differences ($\Delta^+ x$, $\Delta^+ x1$) as follow steps:

1. Let i=0.
2. The bit value of the $(2\times i+1)$-th bit of C is 0, then $\Delta^{\pm} x1_{2\times i+1} = 0$
3. Compute $z = (\Delta^+ x1 \mod 2^{2\times i+2}) - \sum_{j=0}^{2\times i-1} \Delta^{\pm} x1_j \times 2^j$. Then
$$z = \Delta^{\pm} x1_{2\times i+1} \times 2^{2\times i+1} + \Delta^{\pm} x1_{2\times i} \times 2^{2\times i} + 2^{2\times i+2} \mod 2^{2\times i+2}$$
Because $\Delta^{\pm} x1_{2\times i+1} = 0$, then
$$\Delta^{\pm} x1_{2\times i} \times 2^{2\times i} = z + k \times 2^{2\times i+2} \qquad k = 0, \pm 1, \pm 2, \ldots$$
i) If $z = 2^{i\times 2}$, there has $\Delta^{\pm} x1_{2\times i} = z / 2^{i\times 2} = 1$
ii) If $z = 0$, there has $\Delta^{\pm} x1_{2\times i} = z / 2^{i\times 2} = 0$
iii) If $z = 3\times 2^{i\times 2}$, there has $\Delta^{\pm} x1_{2\times i} = (z + (-1)\times 2^{2\times i+2}) / 2^{i\times 2} = -1$
4. If i<n/2, let i=i+1, turn to step 2.
5. Let i=0
6. Compute $z = (\Delta^+ x \mod 2^{2\times i+3}) - \Delta^{\pm} x_{2\times i+2} \times 2^{2\times i+2} - \sum_{j=0}^{i\times 2} \Delta^{\pm} x_j \times 2^j$. Then
$$z = \Delta^{\pm} x_{2\times i+1} \times 2^{2\times i+1} + 2^{2\times i+3} \mod 2^{2\times i+3}$$
Then
$$\Delta^{\pm} x_{2\times i+1} \times 2^{2\times i+1} = z + k \times 2^{2\times i+3} \qquad k = 0, \pm 1, \pm 2, \ldots$$
i) If $z = 2^{i\times 2+1}$, there has $\Delta^{\pm} x_{2\times i+1} = z / 2^{i\times 2+1} = 1$
ii) If $z = 0$, there has $\Delta^{\pm} x_{2\times i+1} = z / 2^{i\times 2+1} = 0$
iii) If $z = 3\times 2^{i\times 2+1}$, there has $\Delta^{\pm} x_{2\times i+1} = (z + (-1)\times 2^{2\times i+3}) / 2^{i\times 2+1} = -1$
7. If i<n/2, let i=i+1, turn to step 6

It will compute out the only one bitwise difference $\Delta^{\pm} x$ with given modular differences ($\Delta^+ x$, $\Delta^+ x1$) by mention steps. So there is only one bitwise deference $\Delta^{\pm} x$ has the given

modular differences ($\Delta^+ x$, $\Delta^+ x1$) that satisfy (2.3) and $\Delta^+ x \neq 0$. □

So it can use (2.2) to make there is only one bitwise difference satisfy some modular differences in the "differential pattern".

3. Data-depend function

In this section, we will expound a way that can restrict the message data conform the "differential pattern".

3.1 A System of Equation

At first, let hash function has two variables x, y1 and y2. And x, y1, y2 satisfy the system of equation as follows:

$$\begin{cases} y1 = x << (n-r) + 2^{n-1-r} \\ y2 = x >> r \end{cases} \qquad 0 \le r \le n-1 \tag{3.1}$$

where r depends on message. (3.1) can be divided, for further analysis, in two cases: $r = r'$ and $r \neq r'$.

3.2 case 1: $r = r'$

To two computes with (x,r) and (x',r'), there is the differences as follow:

$$\begin{cases} \Delta^+ x = x - x' \\ \Delta^+ y1 = x << (n-r) - x'(n-r') \\ \Delta^+ y2 = x >> r - x' >> r' \end{cases} \tag{3.2}$$

There is a proposition as follows:

Proposition 3.1: To given differences $\Delta^+ x, \Delta^+ y1, \Delta^+ y2$, if $\Delta^+ x \neq 0 \quad \Delta r = 0$, (3.2) will has:

$$r = n - \log_2((\Delta^+ y2 \times 2^n + \Delta^+ y1) / \Delta^+ x)$$

Proof: Let:

$$\begin{cases} x := (x_{n-1}, ..., x_0) \\ x' := (x'_{n-1}, ..., x'_0) \\ xl := (x_{n-1}, ..., x_r) \\ xr := (x_{r-1}, ..., x_0) \\ xl' := (x'_{n-1}, ..., x'_r) \\ xr' := (x'_{r-1}, ..., x'_0) \end{cases} \tag{3.3}$$

There will exist:

$$\begin{cases} \Delta^+ xl = (xl - xl') \\ \Delta^+ xr = (xr - xr') \\ \Delta^+ x = \Delta^+ xl \times 2^r + \Delta^+ xr \\ \Delta^+ y1 = \Delta^+ xr \times 2^{n-r} \\ \Delta^+ y2 = \Delta^+ xl \end{cases} \tag{3.4}$$

Then:

$$\begin{aligned} \Delta^+ x \times 2^{n-r} &= \Delta^+ xl \times 2^n + \Delta^+ xr \times 2^{n-r} \\ &= \Delta^+ y2 \times 2^n + \Delta^+ y1 \end{aligned} \tag{3.5}$$

If $\Delta^+ x \neq 0$, there exists:

$$r = n - \log_2((\Delta^+ y2 \times 2^n + \Delta^+ y1) / \Delta^+ x) \qquad \square$$

So in "differential pattern", if there exist the given differences ($\Delta^+ x, \Delta^+ y1, \Delta^+ y2$) that $\Delta^+ x \neq 0 \quad \Delta r = 0$, the parameter r will be fixed and cannot be modified.

3.3 case 2: $r \neq r'$

Let $\Delta r = r - r'$. We use bitwise differences analyses (3.1). Denote x, y1, y2 as follows:

$$\begin{cases} x := (x_{n-1}, ..., x_0) \\ y1 := (y1_{n-1}, ..., y1_0) \\ y2 := (y2_{n-1}, ..., y2_0) \end{cases} \tag{3.6}$$

Then there has:

$$y1_i = \begin{cases} x_{i-(n-r)} & n - r \leq i < n \\ 1 & i = n - 1 - r \\ 0 & i < n - r - 1 \end{cases} \tag{3.7.1.a}$$

$$x_i = y1_{i+(n-r)} \quad 0 \leq i < r \tag{3.7.1.b}$$

$$y2_i = \begin{cases} x_{i+r} & 0 \leq i < n - r \\ 0 & n - r \leq i \end{cases} \tag{3.7.2.a}$$

$$x_i = y2_{i-r} \quad r \leq i < n \tag{3.7.2.b}$$

To two computes with (x, y1,y2,r) and (x' ,y1',y2',r'), there exist differences as follows:

$$\begin{cases} \Delta^{\pm} x_i = x_i - x'_i & 0 \leq i < n \\ \Delta^{\pm} y1_i = y1_i - y1'_i \\ \Delta^{\pm} y2_i = y2_i - y2'_i \end{cases} \tag{3.8}$$

We suppose r>r', then there exists:

$$\begin{cases} \Delta^{\pm} y1_i = 0 & i < n-r-1 \\ \Delta^{\pm} y2_i = 0 & n-r' \le i \end{cases} \tag{3.9}$$

Because r>r', by (3.7.1.a) and (3.7.2.a), there has:

$$\begin{cases} y1_{n-1-r} = 1 \\ y1'_{n-1-r} = 0 \\ \Delta^{\pm} y1_{n-1-r} = y1_{n-1-r} - y1'_{n-1-r} = 1 \end{cases} \tag{3.10}$$

Because $0 \le r' < r \le n-1$, there exists $0 \le n-1-r < n-1$. By (3.9), if i<n-r-1, there exists $\Delta^{\pm} y1_i = 0$. Then to given difference $\Delta^{\pm} y1$, there exist:

$$r = n-1-\min\{\, i \mid \Delta^{\pm} y1_i = 1\} \tag{3.11}$$

Because $\Delta r = r - r'$, there exists $r' = r - \Delta r$. Now r and r' are computed out, it can compute the message from given differences. We will divide x and x' to two parts and compute as follow two sections.

3.3.1 compute $(x_{r-1},..., x_0)$ and $(x'_{r'-1},..., x'_0)$

If we study (3.7.1.a)~(3.9), we will find that $y1_i = x_{i-(n-r)}$ and $y1_i = 0$ $(n-r \le i < n-r')$. It can compute $x_{i-(n-r)}$, then it can compute $x'_{i-(n-r)}$ by $\Delta^{\pm} x_{i-(n-r)} = x_{i-(n-r)} - x'_{i-(n-r)}$, then there exists $y1'_{i+(r-r')} = x'_{i+r-r'-(n-r')} = x'_{i-(n-r)}$ and $y1_{i+(r-r')} = x_{i+r-r'-(n-r)}$ turn to $\Delta^{\pm} y1_{i+(r-r')} = y1_{i+(r-r')} - y1'_{i+(r-r')} = x_{i-(n-r)+r-r'} - x'_{i-(n-r)}$ to compute $x_{i-(n-r)+r-r'}$…. If repeat these steps, it will compute bits $x_{i-(n-r)+k\times(r-r')}$. Then it can compute $(x_{r-1},..., x_0)$ and $(x'_{r'-1},..., x'_0)$ as follows steps:

1. Let j=0.
2. Let i=0.
3. Compute $x_{j+i\times(r-r')} = y1_{j+i\times(r-r')+(n-r)} = \Delta^{\pm} y1_{j+i\times(r-r')+(n-r)} + y1'_{j+i\times(r-r')+(n-r)}$ by (3.7.1.b) and (3.9).
4. Compute $x'_{j+i\times(r-r')} = x_{j+i\times(r-r')} - \Delta^{\pm} x_{j+i\times(r-r')}$ by (3.8), and if $j + i\times(r-r') < r'$, compute $y1'_{j+i\times(r-r')+(n-r')}$ by (3.7.1.a) as follows:

   $y1'_{j+(i+1)\times(r-r')+(n-r)} = y1'_{j+i\times(r-r')+(n-r')} = x'_{j+i\times(r-r')}$
5. Let i=i+1.
6. If $j + i\times(r-r') < r$, turn to step 3.
7. Let j=j+1.
8. If $j < r-r'$, turn to step 2.

From above-mentioned compute steps, it is easy know that there is sole bits $(x_{r-1},..., x_0, x'_{r'-1},..., x'_0)$ will satisfy (3.8).

3.3.2 compute $(x_{n-1},...,x_r)$ and $(x'_{n-1},...,x'_{r'})$

If we study (3.7.1.a)~(3.9), we will find that $y2'_i = x'_{i+r'}$ and $y2_i = 0$ $(n-r < i \le n-r')$. It can compute $x'_{i+r'}$, then it can compute $x_{i+r'}$ by $\Delta^{\pm}x_{i+r'} = x_{i+r'} - x'_{i+r'}$, then there exists $y2_{i-(r-r')} = x_{i+r'}$ and $y2'_{i-(r-r')} = x'_{i-(r-r')+r'} = x'_{i+r'-(r-r')}$ turn to $\Delta^{\pm}y2_{i-(r-r')} = y2_{i-(r-r')} - y2'_{i-(r-r')}$ to compute $x'_{i+r'-k\times(r-r')}$…. If repeat these steps, it will compute bits $x_{i+r'-k\times(r-r')}$. Then it can compute $(x_{n-1},...,x_r)$ and $(x'_{n-1},...,x'_{r'})$ as follows steps:

We will compute $(x_{n-1},...,x_r)$ and $(x'_{n-1},...,x'_{r'})$ as follows steps:

1. Let j=0.
2. Let i=0.
3. Compute $x'_{n-1-j-i\times(r-r')} = y2'_{n-1-j-i\times(r-r')-r'} = y2_{n-1-j-i\times(r-r')-r'} - \Delta^{\pm}y2_{n-1-j-i\times(r-r')-r'}$ by (3.9.2.b) and (3.10).
4. Compute $x_{n-1-j-i\times(r-r')} = x'_{n-1-j-i\times(r-r')} + \Delta^{\pm}x_{n-1-j-i\times(r-r')}$ by (3.10), and if $n-1-j-(i+1)\times(r-r')-r' \ge 0$, compute $y2_{n-1-j-(i+1)\times(r-r')-r'}$ by (3.9.2.a) as follows:
   $$y2_{n-1-j-(i+1)\times(r-r')-r'} = y2_{n-1-j-i\times(r-r')-r} = x_{n-1-j-i\times(r-r')}$$
5. Let i=i+1.
6. If $n-1-j-i\times(r-r')-r' \ge 0$, turn to step 3.
7. Let j=j+1.
8. If $j < r-r'$, turn to step 2.

From above-mentioned compute steps, it is easy know that there is sole bits $(x_{n-1},...,x_r,x'_{n-1},...,x'_{r'})$ will satisfy (3.8).

So there exists proposition as follows:

Proposition 3.2: To given differences $\Delta^{\pm}x, \Delta^{\pm}y1, \Delta^{\pm}y2$, if $\Delta r \ne 0$, there is sole pair (x,x') satisfy (3.8).

Proof: Let r>r', then it can compute $(x_{r-1},...,x_0)$ and $(x'_{r'-1},...,x'_0)$ by the steps given in section 3.3.1. And it can compute the sole $(x_{n-1},...,x_r)$ and $(x'_{n-1},...,x'_{r'})$ by the steps given in section 3.3.2.

So there is only one pair (x,x') satisfy (3.8) □

By proposition 3.2, we will know that if $\Delta r \ne 0$, there is only one collision will hold given differences. So it can modify the input.

In section 3, we discuss a system of equation (3.1) in two cases. And we get proposition 3.1 and proposition 3.2. By proposition 3.1, if the input difference $\Delta^{+}x \ne 0$, the parameter r will be fixed. By proposition 3.2, if the parameter difference does not equal to zero, then there is only collision will conform the given differences.

By the way, the proposition 3.2 use bitwise differences to analyze, and the modular differences is the mainly differences that used to analyses hash function. So the proposition 3.2 maybe not works. But it can use (2.2) to strict the differences.

4. Alterable Iteration Mix Construction (AIMC)

The compression function construction of some hash functions as follows:

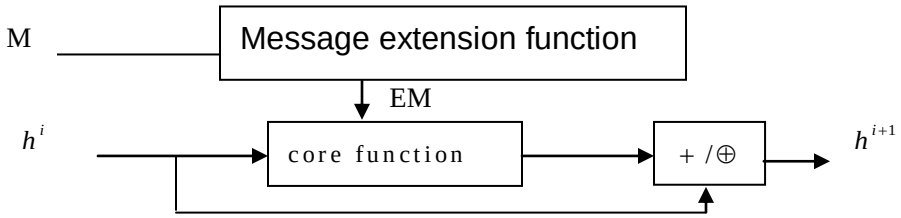

Figure. 1 original compression function construction: M is the message data to be hashed, EM is extended words, h^i and h^{i+1} are hash chain value

With the construction shown in figure 1, it will use simple and high efficiency core function to build compression function. But if the ways that build the viable differential path is found, it is not easy to change the system.

At the same time, hash function is used in different scene, user maybe has different requirement, to meet these requirements, it need hash function are flexible.

M. Schläffer and E. Oswald[0], P.-A. Fouque, G. Leurent, and P. Nguyen[2] had developed an algorithm to build the differential path. Because it need determine all the differences that appear in the differential path. It can treat the algorithm as search technology. The complexity of the algorithm depend on the number of the step operation. So if the hash function has more step operations, it will increaseed complexity of the algorithm.

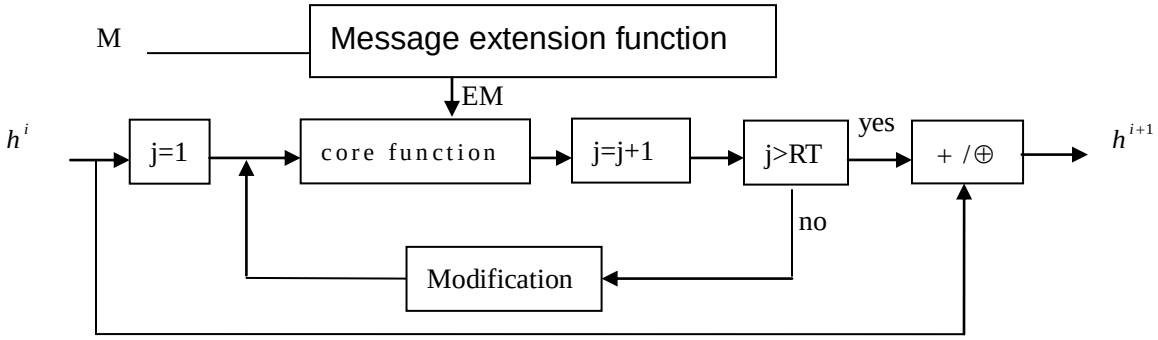

Figure. 2 Alterable Iteration Mix Construction(AIMC): RT is set by user

The direct idea is let user adjust the number of step operations to meet his requirement.

We find a construction Alterable Iteration Mix Construction (AIMC) . The new construction as figure 2:

In construction AIMC, the core functions are put into a circle. To avoid just simple repeat the core functions, it should modify the extended message words, constants or core function every circle.
In construction AIMC, if the parameter RT is turned up, it will lead to:

1. The number of the step operation will be raise. Because the complexity of the algorithm that build the differential path depends on the number of the step operation. So the complexity of the algorithm that build the differential path will be raise.
2. If the hash function is build with some appropriate constructions, when there are more step operations, more bits in message will be fixed.

This will make it harder to build the differential path, and the differential path will fix more bits in message.

5. Conclusions

In this paper, we propose use (2.2) and (3.1) to strict differential path. By proposition 2.1, proposition 3.1 and proposition 3.1, equation (2.2) and (3.1) will strict the differences and the input that conform given differences. In fact, we find another simple nonlinear operation multiplication can be used to strict the variable to given difference. But multiplication needs multiplication operation.

To build a flexible hash function to satify different requirement, we propose construction AIMC that user can change the number of the step operations. We just give out preliminary construction, some details depend on the concrete hash function.

To design economic hash function with (2.2), (3.1) and construction AIMC, it need study core function and some construction to reduce system calculation. To reduce system calculation, it need some way to make there are many bits is fixed in possible lower step operations. Line codes are simple and well researched. It can use the min-Hamming weight of a line code to make there will have many differences in a differential path. We had find a line code when we design DDHA[2]. The min-Hamming Weight of the line code in DDHA[2] is 8. To find out the effect and the way to improve, it need find out some character about the line code. We will work on it.

References:

[EUROCRYPT'05] Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In Cramer [Cra05], pages 19–35.

[Dau05] Magnus Daum. Cryptanalysis of Hash Functions of the MD4-Family. PhD thesis, Ruhr-Universität at Bochum, 2005.

[Dob95] H. Dobbertin. Alf swindles Ann. CryptoBytes, 1(3):5, 1995

[Dob96a] H. Dobbertin. Cryptanalysis of MD4. In Fast Software Encryption – Cambridge Workshop, volume 1039 of LNCS, pages 53–69. Springer-Verlag, 1996.

[Dob96c] H. Dobbertin. The status of MD5 after a recent attack. Crypto-Bytes, 2(2):1–6, 1996

[Dob97] H. Dobbertin. RIPEMD with two-round compress function is not collision-free. Journal of Cryptology, 10:51–68, 1997.

[Dob98a] H. Dobbertin. Cryptanalysis of MD4. Journal of Cryptology, 11:253–274, 1998.

[CJ98] F. Chabaud and A. Joux. Di?erential Collisions in SHA-0. InAdvances in Cryptology – CRYPTO'98, volume 1462 of LNCS, pages 56–71. Springer-Verlag, 1998.

[WYY05] X. Wang, Y. L. Yin, and H. Yu. Collision Search Attacks on SHA1. preprint, February 2005. (http://www.infosec.sdu.edu.cn/paper/sha-attack-note.pdf).

[0] P.-A. Fouque, G. Leurent, and P. Nguyen. Automatic search of differential path in md4. ECRYPT Hash Workshop 2007.
http://events.iaik.tugraz.at/HashWorkshop07/papers/Leurent_MD4.pdf

[1] M. Schläffer and E. Oswald. Searching for differential paths in md4. In M. J. B.Robshaw, editor, FSE, volume 4047 of Lecture Notes in Computer Science, pages 242{261. Springer, 2006

[2] ZiJie Xu and Ke Xu, Data-Depend Hash Algorith, http://eprint.iacr.org/2009/618.