Use Data-depend Function Build Message Expansion Function

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

Abstract: We had found functions can be used to fix bits [2] by given differences. We use these functions build a message expansion function. In the message expansion function, there are some bits include message bits and incremental bits produced from message bits, these bits will be used as parameter of date-depend function. This message expansion function will fixed at lease $5.5 \times n$ bits with given differences, and any message modification will affect at least 8 data-depend function parameter.

Key Word: Message expansion function, Data-depend function, message modification

1. Introduction

When use differential attacks method to attack hash functions, it need build differential path, and then find the collisions that satisfy the differential scheme. In second step, the main methods include message modification [EUROCRYPT'05] and tunnel [1]. If there is some ways reduce the bits can be modified and make any modification will affect many operations that produce message expansion words, it will astrict the methods that base modification. We had found it can use some data-depend function [2] fix variable. We design a message expansion function construction. With the construction and functions, it can build a message expansion function that will fixed lots bits of message by given differences, and any message modification will affect some message expansion words compute.

In section 2, we will review these functions and tools will be use. In section 3, we will expound the message expansion function.

2. Some Functions

In section 2, we will review functions that mentioned at document [2] and other functions will be used to build message expansion function.

We will use module difference and bitwise difference [DAU05] to analyze.

2.1 Astrict Functions

In section 2.1, we will simple review the functions mentioned in document [2].

2.1.1 Astrict Function 1

Module difference and bitwise difference are two main difference used to analyzed hash function. These two differences is not on-one correspondence. Many bitwise differences maybe correspond with same module difference. Then to a sequence of module difference, it can use different sequence of bitwise difference to analyze. To reduce the number of the sequence of bitwise difference that conform same sequence of module difference, it can use function as follow:

$$x1 = x \wedge \sum_{i=0}^{n/2-1} 2^{i\times 2} = x \wedge 0x555.... \tag{2.1}$$

If variable x and x1 satisfy (2.1), there will exist:

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

Because module difference and bitwise difference is not on-one correspondence, some bitwise difference characteristic cannot be used to module difference. By proposition (2.1), (2.1) will make there is only one bitwise difference pair conform given module difference pair. It can use (2.1) to make bitwise difference pair and module difference pair has same characteristic.

2.1.2data-depend equations

In the document [2] , we find a data-depend equations as follow:

$$Y = F2(r,x) = \begin{pmatrix} y1 \\ y2 \end{pmatrix} = \begin{cases} y1 = x << (n-r) + 2^{n-1-r} \\ y2 = x >> r \end{cases} \tag{2.2}$$

The difference equations as follow:

$$\Delta^+ Y = F2(r',x') - F2(r,x) = \begin{pmatrix} \Delta^+ y1 \\ \Delta^+ y2 \end{pmatrix} = \begin{cases} \Delta^+ y1 = (x' << (n-r') + 2^{n-1-r'}) - (x << (n-r) + 2^{n-1-r}) \\ \Delta^+ y2 = (x' >> r') - (x >> r) \end{cases} \qquad (2.2.a)$$

There are two propositions as follow:

Proposition 2.2: *To given differences* $\Delta^+ x, \Delta^+ y1, \Delta^+ y2$*, if* $\Delta^+ x \neq 0 \quad \Delta r = 0$*, (2.2) will has:*

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

Proposition 2.3: *To given differences* $\Delta^{\pm} x, \Delta^{\pm} y1, \Delta^{\pm} y2$*, if* $\Delta r \neq 0$*, there is sole pair (x,x') conform given differences.*

By proposition 2.2, 2.3, if the parameter or input has difference, the variable or parameter of (2.2) will be fixed. In message expansion function in section 3, mostly parameters and variable is message bits. So (2.2) will directly fix message bits.

Proposition 2.3 is based on bitwise difference. This means it cannot use proposition 2.3 on module difference. But because (2.1) can astrict difference, it can use (2.1) to recast (2.2) to make it can use proposition 2.3 on module difference.

2.2 Divide word to 16 cells

We divide a n-bit word to 16 parts, one part is a cell. One cell will have n/16 bits. Then if the u-th bit is the w-th bit of v-th cell, there will exists:

$$u = v \times (n/16) + w \qquad (2.3)$$

The cell will become the data-depend shift parameter.

2.3 Inside Hash Function

In section 3, it is easy to know that when input high weight collision, there will fix many cells. But there are light weight collision will fix few cells. So it needs ways to increase the weight when input light weight collision. Because the height weight collisions and light weight collisions are different, we design a simple inside hash function to filtrate light weight collision.

The inside hash function is fixed input length hash function, because the function of inside hash function is filtrate light weight collision, when input light weight collision, the inside hash function will produce non-zero output. We use follow four function to build inside hash function:

1. $h1(M) = \oplus_{i=0}^{15} m_i$
2. $h2(M) = \oplus_{i=0}^{15} (2^{n/16 \times i} \times (\oplus_{j=0}^{15} (m_i >> n/16 \times j) \wedge (2^{n/16} - 1)))$
3. $h3(M) = \oplus_{i=0}^{15} ROLR^{n/16 \times i}(m_i)$

4. $h4(M)=(\oplus_{i=0}^{7} ROLR^{(15-i)\times n/16}(m_i))\oplus(\oplus_{i=8}^{15} ROLR^{(30-i)\times n/16}(m_i))$

Where ROLR is rotate right (circular right shift) operation. The series of output of the 4 function is the inside hash value. Then there exists:

Proposition 2.4: *if $\triangle m$ is a collision of inside hash function, there exists:*

$$HW(\triangle m)\geqslant 8$$

Proposition 2.5: *if input $\triangle m$ into h1 or h2 or h3 or h4, there exists:*

$$HW(\triangle m)\geqslant HW(h(\triangle m))$$

Where h is h1 or h2 or h3 or h4.

We will not direct proof proposition 2.4 and 2.5 here. We had written a program to find the light weight collision of inside hash function. And there is not a collision that light than 8.

By define of function h1, h2, h3, h4, the output will not weight than input and there exists:

$$(HW(M)-HW(h(M)))\equiv 0 \bmod 2 \tag{2.4}$$

Where h is function h1, h2, h3, h4.

The inside hash function is simple, and the light weight collisions have some characteristics, if it need increase the weight of the collision in some schemes, it can add other function aim at the light weight collisions.

2.4Enhancement Function

The condition of proposition 2.2, 2.3 is variable or parameter has differences. So it can reduce the number of the fixed bits by select light weight collision. To avoid this, we use some enhancement functions to increase the weight as follow:

$$\begin{cases} y=ST(x)=\oplus_{i=0}^{15} ROLR^{r_i}(x) \\ r_i\neq r_j \quad i\neq j \quad (r_i\neq null \quad or \quad r_i\neq null) \end{cases} \tag{2.5}$$

Where x, y are n-bit word. We divide a word to 16 cells, so the parameter r_i is integral multiple of n/16. Then there exists:

$$r_i\in\{null,0,n/16,2\times n/16,...,k\times n/16,...,15\times n/16\} \quad 0\leq i<16 \quad 0\leq k<16$$

Where *null* means there is not the operation. The enhancement functions are built with simple operation for efficiency. Because it need define 16 parameters, and if the parameters are not same value except *null*. It can use a 16-bit word describe an enhancement function. If there exists the operation $\text{ROLR}^{i\times n/16}(x)$, the i-th bit of the

16-bit is 1. Then there are 2^{16} possible enhancement function, we can select the enhancement function that we need by check and compare all possible enhancement functions.

In some scheme, one enhancement function will not satisfy the requirement, it will need several enhancement functions. If it needs J enhancement functions, then these enhancement functions should make the follow formula to achieve maximum:

$$CHW\left(\bigcup_{j=0}^{J} y_j\right) = CHW\left(\bigcup_{j=0}^{J} \left(\oplus_{i=0}^{15} ROLR^{r_{i,j}}(x)\right)\right) \tag{2.6}$$

Where CHW(x) is the number of non-zero cells. There will be $J\times 16$ parameters. The possible enhancement function combination is $2^{J\times 16}$. It is hard to select the enhancement function combination by check and compare all possible enhancement function combination s. So it can use greedy method to select the enhancement functions.

When we selected enhancement function, we find that there are many enhancement functions make (2.6) have same value, so we use some other target to select enhancement function in the source code https://github.com/xuzijiewz/LCFOREF.

2.5 Simplify analysis

The message block has 16words, it is 16×32(resp.16×64) bits. Some time, it is hard to analyses for huge input space. Inside hash function and enhancement function has follow characteristics:

1. Enhancement function and inside hash function use xor and shift operation.
2. The parameter of shift left operation is integral multiple of n/16.

These two characteristics make it can use 16-bit word study enhancement function and inside hash function. More discuss about these measures is in appendix A.

3. Message expansion function

In section 3, we will expound and analysis message expansions function. Our focuses are the minimum fixed cells number and the effect of message modification

Let HW(x) is hamming weight of x.

3.1Construction of Message Expansion Function

The construction of message expansion functions as follow:

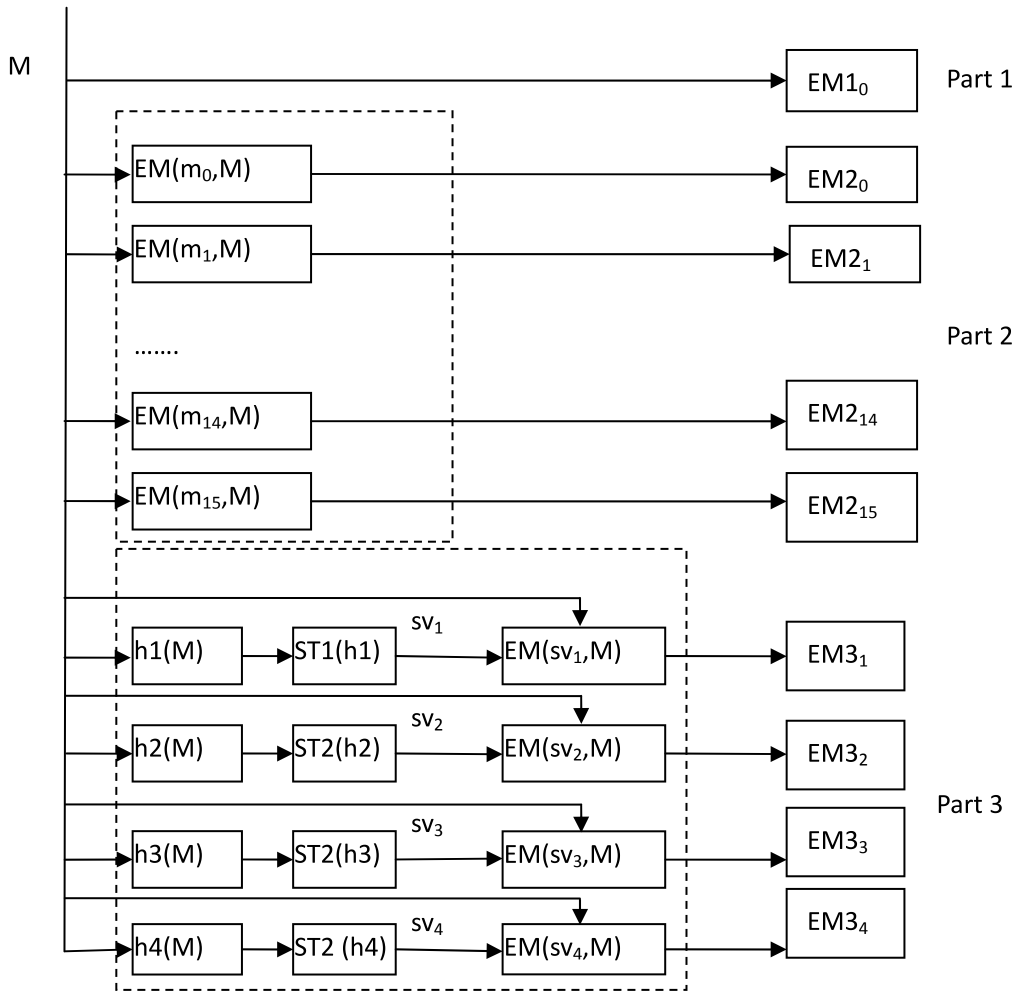

Fig 1 the construction of message expansion function

Where ST1, ST2 are enhancement function. We will expound function ST1, ST2, EM later. The message expansion includes three parts:

1. Part 1includes EM1$_0$. EM1$_0$ include 16 words, it equal to message.
2. Part 2 includes EM2$_0$, … , EM2$_i$,…,EM2$_{15}$. EM2$_0$, … , EM2$_i$,…,EM2$_{15}$ are output of function EM, the massage word bit will direct become data-depend shift parameters.
3. Part 3 includes EM3$_1$, EM3$_2$, EM$_3$, EM3$_4$. EM3$_1$, EM3$_2$, EM$_3$, EM3$_4$ are output of function EM, but one input is output of enhancement function ST1 or ST2. The message will be input into inside hash function, and input inside hash value into enhancement function, then the output of enhancement function will become data-depend shift parameters.

3.2 Function EM(m,M)

Function EM as follow:

$$Y(m,M)=\begin{pmatrix} y_0 \\ \dots \\ y_i \\ \dots \\ y_{15} \end{pmatrix}=\begin{cases} y_0 = F2(m\%2^{n/16}, m_0) \\ \dots \\ y_i = F2((m >> (i \times n/16))\%2^{n/16}, m_i) \\ \dots \\ y_{15} = F2(m >> (15 \times n/16))\%2^{n/16}, m_{15}) \end{cases}$$

The function EM(m,M) takes as input 2 values:

1. A word m that will be parameter of function F2. Function F2 is data-depend equations (2.2)
2. The message block M.

Function EM(m,M) takes as output 1 values:

1. 16 outputs of function F2 $y_0,\dots,y_{15}$, very output has 2 words.

By define of function EM, i-th cell of m will be used as data-depend shift parameter with i-th message word. By proposition 2.2 and 2.3, if i-th cell or i-th message word includes difference, i-th word or i-th cell will be fixed.

3.3 Part 1

The part 1 of message expansion word include $EM1_0$, $EM1_0$ include 16 words as follow:

$$EM1_0 = (m_0,\dots,m_i,\dots,m_{15})$$

$EM1_0$ include 16 words, these 16 words are equal to message words. Then in any differential path, part 1 will produce the difference of message. The message difference will be input difference of some function.

3.4 Part 2

The part 2 of message expansion word include $EM2_0,\dots,EM2_i,\dots,EM2_{15}$, $EM2_i$ is produced by input message block and a word that will be of shift operation parameter:

$$EM2_i = EM(m_i, M) \qquad 0 \le i < 16$$

The message block has 16 words, and every word will be parameter of shift left operation once.

Now let us discuss the minimums cells that are fixed if input differences into part 2. For predigest discuss, let us suppose:

1. If input differences is $\Delta m_0,...,\Delta m_i,...,\Delta m_{15}$. Let the number of the non-zero difference is a as follow:
$$a = \sum_{i=0}^{15} \{1 \mid \Delta m_i \neq 0\}$$
2. Let there are 16 16-bit words $w_0,\dots,w_i,\dots,w_{15}$. If j-th cell of Δm_i is not zero, the j-th bit of w_i is 1.

In part 2, the bits will be fixed in two cases:

I. The condition of proposition 2.3 is satisfied.
By section 3.2, if i-th cell of Δm_i, is non-zero difference, i-th word will be fixed. So it can find out the words that had been fixed from $w = w_0 \mid ... \mid w_i \mid ... \mid w_{15}$. If i-th bit of w is 1, then i-th word is fixed. we suppose b=HW(w).

II. The condition of proposition 2.2 is satisfied.
By $EM(\Delta m_i, \Delta M)$, if Δm_i is non-zero difference, the i-th cell of $\Delta m_0,...,\Delta m_i,...,\Delta m_{15}$ are fixed. Because there are b words are fixed in case I, so the number of other fixed cells will be $(16-b)\times a$.

So, the minimum fixed cells will be:
$$b \times 16 + a \times (16-b) = 16 \times (a+b) - a \times b \tag{3.1}$$
Le the collision has p cells are non-zero difference, because there are b words has non-zero difference, and every has no more than a cells are non-zero difference, so there will exists:
$$p \le a \times b$$
Because there exists a，b<16, to given p, a, there exists:
$$\begin{aligned} & 16 \times (a+b) - ab \\ & = 16 \times a + b \times (16-a) \\ & \ge 16 \times a + p/a \times (16-a) \\ & = 16 \times (a + p/a) - p \end{aligned}$$
To given p, there exists:
$$16 \times (a + p/a) - p \ge 32 \times \sqrt{p} - p \tag{3.2}$$
When $a = \sqrt{p}$ and $b = p/a$, the equality hold. Because when $0 \le p \le 256$, (3.2) is increasing function, if there is more non-zero cell differences, the minimum fixed cells will be increased.

3.5 Part 3

By (3.2), if input height weight difference collision, there will be lots fixed cells. And if input light weight difference collision, it can avoid fix many cells. To fix enough cells, we append part 3 to message expansion. The function of part 3 is to increase the number of fixed cell when input light weight difference collision.

In part 3, the message will be inputted an inside fixed length hash function, then input the inside hash value to enhancement functions. Then the output of enhancement functions will be use produce message expansion words.

The inside fixed length hash function has not light weight difference collision, if input light weight difference collision, the inside fixed length hash function will produce non-zero inside hash value differences.

The inside hash value differences will be inputted into enhancement functions. The enhancement functions are used to increase the weight to fix more cells.

By the construction of part 3, it can increase the number of fixed cell when input light weight difference collision.

We will discuss inside hash function and enhancement function in section 3.5.1 and 3.5.2.

3.5.1 Inside Hash Function

The inside hash value include four words produced by four function h1, h2, h3, h4. Function h1, h2, h3, h4 have been describe in section 2.3. By proposition 2.4, inside hash function has not collision light than 8.

By (2.4), if the weight of input difference is odd, then there exists:

$$(HW(h(M))-1)\equiv 0 \bmod 2$$

$$HW(h(M))>0$$

So the weight of collision of inside will be even. Then by the result of source code https://github.com/xuzijiewz/IHFC, there exists:

HW(M)	8	10	12
Minimum fixed cells	⩾88	⩾112	⩾112

Fig 2. Minimum fixed cells of collision

3.5.2 Enhancement Function

The enhancement function is some function like (2.5), and there are two enhancement function st1, st2.

The function of enhancement function is increase the weight when input light weight differences. At the same time, the minimum weight of inside hash function is 8 and the output of function h1, h2, h3, h4 light than input difference. So we suppose the enhancement function will enhance the input differences that light than 8.

3.5.2.1 Enhancement Function st1

The input of enhancement function st1 is output of function h1. In section 3.4, in order to know which word is fixed, we use 16 16-bit words $w_0,\ldots,w_i,\ldots,w_{15}$ to simplify analysis. And if i-th word is fixed, i-th bit of $w = w_0 | \ldots | w_i | \ldots | w_{15}$ is 1. The function h1 is $h1(M) = \oplus_{i=0}^{15} m_i$. Because to $w = w_0 | \ldots | w_i | \ldots | w_{15}$ and $w1 = \oplus_{i=0}^{15} w_i$, there exists:

$$\begin{aligned} & w | w1 \\ & = w_0 | \ldots | w_i | \ldots | w_{15} | (w_0 \oplus \ldots \oplus w_i \oplus \ldots \oplus w_{15}) \\ & = w_0 | \ldots | w_i | \ldots | w_{15} \\ & = w \end{aligned} \tag{3.3}$$

(3.3) means if the i-th bit of w1 is 1, the i-th bit of w is 1. So if the i-th cell of h1 is non-zero difference, m_i is fixed in part 2.

So when we select enhancement function st1 for the input x that has the follow character:

1. If i-th cell of the input is non-zero difference, m_i is fixed.
2. HW(x)<8

Then we run program https://github.com/xuzijiewz/LCFOREF. And select the follow enhancement function for st1:

$$ST1(x) = ROTR^{12\times n/16}(x) \oplus ROTR^{8\times n/16}(x) \oplus ROTR^{4\times n/16}(x) \oplus ROTR^{2\times n/16}(x) \oplus ROTR^{n/16}(x)$$

And there exists:

Hw(x)	1	2	3	4	5	6	7
Hw(st1(x))	5	⩾6	⩾5	⩾4	⩾3	⩾4	⩾3
Lightest collision: Hw(st1(x))+Hw(x)	6	⩾8	⩾8	⩾8	⩾8	⩾10	⩾10
Minimum fixed words: Hw(st1(x)|x)	6	⩾6	⩾7	⩾6	⩾6	⩾6	⩾7

Fig 3. Performance of function st1

By fig 3, it is easy to know that there exists:

1. if h1(△M)≠0, then there are at least 6 words is fixed.
2. if h1(△M)>1, there are at least 8 differences in △M and h1(△M). This means if modify message, it will change at least 8 function F2 parameters.

3.5.2.2 Enhancement Function st2

The output of function h2, h3, h4 is inputted into enhancement function st2. By (2.4), if the weight of input difference is odd, function h1 will produce odd weight output. So we suppose the odd weight collision will fix enough words by $EM3_1$. So when we select enhancement function st2 for the input x that has the follow character:

1. HW(x)≡0 mod 2

2. HW(x)<8

Then we run program https://github.com/xuzijiewz/LCFOREF. And select the follow enhancement function for st2:

$ST2(x) = ROTR^{8\times n/16}(x) \oplus ROTR^{6\times n/16}(x) \oplus ROTR^{4\times n/16}(x) \oplus ROTR^{n/16}(x) \oplus x$

And there exists:

Hw(x)	1	2	3	4	5	6	7
Hw(st2(x))		⩾6		⩾4		⩾4	
Hw(st2(x))+Hw(x)		⩾8		⩾8		⩾10	

Fig 4. Performance of function st2

Let eh is function h2 or h3 or h4. By fig 4, it is easy to know that there exists:

1. if eh(△M)≠0, then there are at least 4 words is fixed.
2. if eh(△M)>1, there are at least 8 difference in △M and eh(△M). This means if modify message, it will change at least 8 function F2 parameters.

3.6 Performance of Message Expansion Function

In section 3.6, we will discuss the minimum fixed cell and how many message expansion word compute will be affected when modify message.

3.6.1 The Minimum Fixed Cells

When input differences that weight p into message expansion function, by (3.2), there will be at least $32\times\sqrt{p}-p$ cells are fixed in part 2, and if the inside hash value is non-zero, there may be some cells will be fixed. By proposition 2.4, the collision of inside hash function will not light than 8. So we will continue discuss into two cases: the difference light than 8 and other differences.

3.6.1.1 the difference light than 8

We will continue discuss in two cases: h1(△M)=0 and h1(△M)≠0

3.6.1.1.1 case 1: h1(△M)≠0

In fig 1, the output of h1 is inputted into function st1. By proposition 2.5, there exists h1(△M)<8, so by fig 3, there exists:

$$Hw(st1(x)|x) \geqslant 6$$

Because one word has 16 cells, the minimum fixed cells will be:

$$6\times16=96 \tag{3.4}$$

3.6.1.1.2 case 2: h1(△M)=0

By proposition 2.4, if the inputted difference light than 8, the inside hash value will be non-zero. And h1(△M)=0, so there exists:

$$h2(\triangle M)\neq 0 \text{ or } h3(\triangle M)\neq 0 \text{ or } h4(\triangle M)\neq 0$$

By proposition 2.5, there exists :

$$HW(h2(\triangle M))<8 \text{ or } HW(h3(\triangle M)) <8 \text{ or } HW(h4(\triangle M)) <8$$

By (2.4) and h1(△M)=0, there exists HW(△M)≡0 mod 2. So there exists:

$$HW(h2(\triangle M))\equiv 0 \bmod 2$$
$$HW(h3(\triangle M))\equiv 0 \bmod 2$$
$$HW(h4(\triangle M))\equiv 0 \bmod 2$$

If h1(△M)=0, there will has at least two word has non-zero difference, otherwise there will exists △M=0 or h1(△M)≠0.

So in part 2 there will be at least 2×16=16 cells will be fixed.

If there are q words are fixed in part 3, the minimum fixed cells will be:

$$q\times 16+2\times(16-q)= q\times 14+32$$

By fig 4, there exists q≥4. So the minimum fixed cells will be:

$$4\times 14+32=88 \tag{3.5}$$

3.6.1.1 the difference no light than 8

By (3.1), if input p weight difference, there is a words are non-zero difference and every word has no more than b non-zero cells, then in part 2, there exists:

$$p\leq a\times b \tag{3.6}$$

the minimum fixed cells will be:

$$\begin{aligned}&b\times 16+a\times(16-b)\\ &\geq b\times 16+p/b\times(16-b)\\ &=b\times 16+p\times(16/b-1)\\ &\geq b\times 16+8\times(16/b-1)\end{aligned} \tag{3.7}$$

The derivate of (3.7) is $16-128/(b\times b)$ (3.8)

So if there exists:

$$\begin{aligned}&16-128/(b\times b)=0\\ &b=2.8\end{aligned}$$

(3.7) is increasing function, so when b in 2 and 4, (3.7) will take the minimum. We will continue discuss in two follow cases.

Case 1: b is 2 or 4

When b is 2 or 4, there exists:

$$\begin{aligned} & b\times 16+a\times(16-b) \\ & \geq 2\times 16+8\times(16/2-1) \\ & =88 \end{aligned} \tag{3.9.1}$$

$$\begin{aligned} & b\times 16+a\times(16-b) \\ & \geq 4\times 16+8\times(16/4-1) \\ & =88 \end{aligned} \tag{3.9.2}$$

Case 2: b is 3

When b is 3, by (3.6), there exists: $a \geqslant 8/b = 2.6$ (3.10)

If $HW(a) \equiv 1 \bmod 2$, then there exists: $HW(h1(\triangle M)) \equiv 1 \bmod 2$. By fig 3, the minimum fixed cells will be:

$$6 \times 16 = 96 \tag{3.11}$$

If $HW(a) \equiv 0 \bmod 2$, then by (3.9), there exists: $a \geqslant 4$

Then there exists: $b \times 16 + a \times (16-b) \geqslant 3 \times 16 + 4 \times (16-3) = 48 + 52 = 90$ (3.12)

By (3.4), (3.5), (3.9.1), (3.9.2), (3.11), (3.12), the minimum fixed cells will be 88 cells.

3.6.2 The least parameters message modification change

In a differential analysis, the message modification will change the original collision to conform more differences in a differential scheme. It is equivalent to append some difference to the original collision. Then the new append differences will change the compute of some message expansion words.

If modify less than 8 cells of message, by proposition 2.4, the inside hash value will be changed. By fig 3 and fig 4, if there are more one cells of input are modified, there are more than 8 cells in message and the output of enhancement functions are changed. Then there will be parameters of at least 8 functions EM are changed in part 2 and part 3. If there is only one cell in message is modified, by the define of inside hash functions, fig 3 and fig 4, there are $1+5\times 4=21$ cells are changed in part 2 and part 3. So the message modification will change at least 8 function F2 parameters.

When input any collision, by the discuss in section 3.6.1, there will be at least 88 cells will be fixed, that is $5.5\times n$ bits, n is the bit-length of word. At the same time, by discuss in section 3.6.2, any message modification will change at least 8 function F2 parameters.

3.7 Astrict the bitwise difference conform given module difference

Proposition 2.3 is based on bitwise difference, there may be many bitwise difference sequences conform a given module difference sequence. It can use (2.1) ascrit bitwise differences of a bitwise difference sequences to reduce bitwise difference sequences conform given module difference sequence. It need change equation (2.2) as follow:

$$Y = F2(r,x) = \begin{pmatrix} y1 \\ y2 \end{pmatrix} = \begin{cases} y1 = x << (n-r) + 2^{n-1-r} \\ y2 = x >> r \\ y3 = y1 \wedge \sum_{i=0}^{n/2-1} 2^{i\times 2} \\ y4 = y2 \wedge \sum_{i=0}^{n/2-1} 2^{i\times 2} \end{cases} \qquad (3.13)$$

If equation (2.2) is changed to (3.13), by proposition 2.1, there is only one ($\Delta^{\pm}y1, \Delta^{\pm}y2$) conform given difference ($\Delta^{+}y1, \Delta^{+}y2, \Delta^{+}y3, \Delta^{+}y4$). By proposition 2.3, if $\Delta r \neq 0$, there is only collision conform given ($\Delta^{\pm}x, \Delta^{\pm}y1, \Delta^{\pm}y2$). So if $\Delta r \neq 0$, there is only collision conform given difference ($\Delta^{+}y1, \Delta^{+}y2, \Delta^{+}y3, \Delta^{+}y4, \Delta^{+}x$).
Because the message words will be repeat inputted into function EM, so to astrict the bitwise difference of message, it can change part 1 as follow:

$$EM1_0 = (m_0, \ldots, m_i, \ldots, m_{15}, ml_0, \ldots, ml_i, \ldots, ml_{15})$$
$$ml_i = m_i \wedge \sum_{j=0}^{n/2-1} 2^{j\times 2} \qquad 0 \le i \le 15$$

Then the difference of input x of (3.13) will be astrict.

In section 3, we had expounded the message expansion function. By the analysis in section 3.6, to given bitwise differences of message expansion words, there are at least 88 cells are fixed. At the same time, any modification will change at least 8 function F2 parameters. In section 3.7, we give out a way to use (2.1) to reduce the bitwise differences that conform given module differences.

4. Conclusion

Here we design a message expansion function that will fix some message bits by given differences. The message expansion function is build with data-depend shift operations (2.2), inside hash function and enhancement functions. The construction and functions make achieve the aim that fix at least 5.5×n bits. At the same time, the construction make any modification will affect at least 8×2 message words compute.

By discuss in section 3.6, it is known that the inside hash function and enhancement functions decide the minimum fixed bits and the minimum effect data-depend shift parameters. So it can increase the fixed bits and effect parameters by augment inside hash function and enhancement functions.

The message expansion words include 656 words. To build a quick hash function, it needs corresponding core function.

Reference:

[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.

[1]Vlastimil Klima. Tunnels in hash functions: MD5 collisions within a minute. (IACR Cryptology ePrint Archive: Report 2006/105 http://eprint.iacr.org/2006/105), 2006.

[2]ZiJie Xu and Ke Xu, Ways to restrict the differential path, http://eprint.iacr.org/2011/307

Appendix A: Simplify analysis

In appendix A, we will discuss how to simplify the analysis for inside hash function and enhancement function.

At first, inside hash function and enhancement function has follow characteristics:

1. Enhancement function and inside hash function use xor and shift operation.
2. The parameter of shift operation is integral multiple of n/16.

And we note i-th bit of word x as x^i.

1. Functions Form

The enhancement function and inside hash function can use follow function form describe:

$$f = F(x_0,..., x_i,...) = (\oplus_{j=0}^{J1-1} \oplus_{i=0}^{I1_j-1} SHR^{r1_{i,j}\times n/16}(x_j)) \oplus (\oplus_{j=0}^{J2-1} \oplus_{i=0}^{I2_j-1} SHL^{r2_{i,j}\times n/16}(x_j)) \qquad (a.1)$$

Where n/16 is size of cell. Function F has J n-bit word input, the parameter of right shift operation of j input is $r1_{i,j}$ ×n/16, the parameter of left shift operation of j input is $r2_{i,j}$ ×n/16.

The message word will be inputted into different function step by step. Because there are xor and shift operation, the complex function is constructed in follow way:

$$z = SHR^r(x \oplus y) \qquad z = SHL^r(x \oplus y)$$

Where x, y are output of some kind function of (a.1). There exists:

$$\begin{aligned} z &= SHR^r(x \oplus y) \\ &= SHR^r(x) \oplus SHR^r(y) \end{aligned}$$

SHL operation resembles. So it can transform the complex function to some function as (a.1).

2. Compute the Bits of Output

If the parameter of shift right operation is r, the u-th bit will be moved to (u-r)%n. If the parameter of shift left operation is r, the u-th bit will be moved to (u+r)%n. Because xor operation will not create carry bit, so it can compute u-th bit of output of function f as follow:

$$f^u = (\oplus_{j=0}^{J-1} \oplus_{i=0}^{I_j-1} x_j^{u+r1_{i,j}\times n/16}) \oplus (\oplus_{j=0}^{J-1} \oplus_{i=0}^{I_j-1} x_j^{u-r2_{i,j}\times n/16})$$

By (2.3), if u1-th bit is w1-th bit of v1-th cell, there exists:

$$\begin{cases} u1 = v1 \times (n/16) + w1 \\ u1\%(n/16) = w1 \end{cases}$$

If parameter of i-th shift operation of j-th variable is $r_{i,j} \times n/16$, the u2-th bit of x_j will be moved to u1, then there exists:

$$\begin{aligned} u2 &= u1 \pm r_{i,j} \times n/16 \\ &= v1 \times n/16 \pm r_{i,j} \times n/16 + w1 \end{aligned}$$

Then:

$$u2\%(n/16) = w1 \qquad (a.2)$$

So the bits used to compute u1-th bit of output will satisfy (a.2). Because w1 is the place of bit in a cell, so it can divide bits into independent groups by the place in a cell.

If bits are divided into different groups by the place in a cell, each group has one and only one bit in a cell. By proposition 2.2 and 2.3, if parameter or variable has non-zero difference, the variable or parameter will be fixed. And if a cell has a non-zero difference bit, the cell will be non-zero difference. So the numbers of non-zero cell difference will no less than any group difference weight. When one bit group has differences, and other bit group has same differences or no difference, the numbers of non-zero cell difference will be same as the bit group that has non-zero differences.

3. Use 16-bit Word

It can divide bits into independent groups by the place in a cell. Then number groups by the order of the bit in the cell. Then it can change (a.1) for w-th group as follow:

I. select the w-th bit of every cell of a word, construct a new 16-bit word with the selected bits by the order of the cell. If the u-th bit is w-th bit of v-th cell of word x, then the u-th bit will be v-th bit of new 16-bit word. And there exists:

$$v=（u-w）/（n/16） \qquad (a.3)$$

II. change (a.1) as follow:

$$f = F(x_0, \ldots, x_i, \ldots) = (\oplus_{j=0}^{J-1} \oplus_{i=0}^{I_j-1} SHR16^{r1_{i,j}}(x_j)) \oplus (\oplus_{j=0}^{J-1} \oplus_{i=0}^{I_j-1} SHL16^{r2_{i,j}}(x_j)) \qquad (a.4)$$

Where $r1_{i,j}$ and $r2_{i,j}$ is same as (a.1), SHR16 is shift right operation of 16-bit word, SHL16 is shift left operation of 16-bit word.

Let y2 is one of 16-bit words that divide n-bit word y1, $x2_j$ is one of 16-bit words that divide n-bit word $x1_j$.

Then to the v-th bit of y2, there exists:

$$y2^v = y1^u = (\oplus_{j=0}^{J-1} \oplus_{i=0}^{I_j-1} x1_j^{u+r1_{i,j}\times n/16}) \oplus (\oplus_{j=0}^{J-1} \oplus_{i=0}^{I_j-1} x1_j^{u-r2_{i,j}\times n/16})$$

By (a.3), the $(u \pm r_{i,j} \times n/16 + n)\%n$-th bit of word $x1_j$ will be placed the place of word $x2_j$ as follow:

$$\begin{aligned}&(u \mp r_{i,j} \times n/16 - w)/(n/16)\\ &= (v \times n/16 + w \mp r_{i,j} \times n/16 - w)/(n/16)\\ &= v \mp r_{i,j}\end{aligned}$$

There exists:

$$\begin{aligned}&y2^v = y1^u = (\oplus_{j=0}^{J-1} \oplus_{i=0}^{I_j-1} x1_j^{u+r1_{i,j}\times n/16}) \oplus (\oplus_{j=0}^{J-1} \oplus_{i=0}^{I_j-1} x1_j^{u-r2_{i,j}\times n/16})\\ &= (\oplus_{j=0}^{J-1} \oplus_{i=0}^{I_j-1} x2_j^{v+r1_{i,j}}) \oplus (\oplus_{j=0}^{J-1} \oplus_{i=0}^{I_j-1} x2_j^{v-r2_{i,j}}) \qquad (a.5)\end{aligned}$$

By (a.4), it can compute the v-th bit of output as follow:

$$y2^v = (\oplus_{j=0}^{J-1} \oplus_{i=0}^{I_j-1} x2_j^{v+r1_{i,j}}) \oplus (\oplus_{j=0}^{J-1} \oplus_{i=0}^{I_j-1} x2_j^{v-r2_{i,j}}) \qquad (a.6)$$

(a.5) is same with (a.6). So I and II will not change the original compute. In the above discuss, it can find that (a.5) and (a.6) can be used for every group.

So it can use I and II to simplify the analysis of inside hash function and enhancement function. We had use these measures in source code https://github.com/xuzijiewz/LCFOREF and https://github.com/xuzijiewz/IHFC.