A New and Collision-resistance Hash Function DIHA2

Xigen Yao
Wuxi Hengqi Electromechanical Device Co.Ltd.China
email : dihuo377@163.com

Abstract The new hash function DIHA2 (Dynamic Input Hash Algorithm)is with the structure of Merkle-Damgard and is based on 64-bit computing.It operates each 1024-bit block and outputts a 256-bit hash-value. For a 64-bit sub-block $X[j](0 \leq j \leq 15)$ of each step, DIHA2 gets a dynamic mapping value of TLU (table look up,The table was 256-Byte only)and add it to operation of variables a, b, c, d,so as to eliminate the differential effect.At the same time DIHA2 sets 3 assistant register variables r_1, r_2, r_3 to store the mapping value and resume loading 3 steps later, so as to be interleaving. DIHA2 therefore obtained strong avalanche effect than the others and can resist the sharp and serious attack of differential.
keywords: hash function , dynamic mapping, avalanche , differential

1 Introduction:

Hash function is very important in information security and data integrity. Since 2005-2006 , the primary hash functions have been proved nolonger safe.We can see many recent attacks on collision resistance of existing hash functions [BC04,RO05,Kli05,WLF+05,WY05,BCJ+05,WYY05b,WYY05a] , it is that the most effectual attacks use differential method to find collisions.

Now SHA2, which we depends on comes from MD4 and have the same structure(the Merkle-Damgard[Dam89,Mer89]construction hash functions). It is now hot necessitous pressing and so difficult to creat a new and secure hash function. DIHA2 was a new hash function which based on the Merkle-Damgard construction.By using dynamic mapping input,DIHA2 is able to resist those who are sharp and serious differential attacks on collision resistance .

The new hash function DIHA2(Dynamic Input Hash Algorithm) was 64-bit computing.It operated each 1024-bit block for 16 steps with 4 rounds,and outputted a 256-bit hash-value.It added 2 settings into the computing:

* dynamic mapping input of each sub-block

* 3 assistant register variables r_1, r_2, r_3.

In DIHA2 ,the dynamic mapping value of addition was with 2^{64} types in a step .It was inputted to computing and feeded back .That gave DIHA2 the effects of stronger avalanche and eliminating the fixed differential. DIHA2 set 3 assistant register variables r_1, r_2, r_3 , which would resume loading the mapping value 3 steps later so as to be interleaving.This was helpful for one-way effect.

There were more register variables in DIHA2. It is well known that a mainstream chip had 16 64-bit general registers already. Even though the time each step cost was as 3 times as MD5 , DIHA2 operated 1024 bits of a block with 4 rounds,so the whole speed of DIHA2 would not be too slow.

This paper is organized as follows: Section 2,is the explanation of DIHA2 algorithm,Section3 is the explanation of attack on MD5 by WANG's differential method,Section4,is the explanation of how DIHA2 overcome the differential attack.Finally, Section 5 is the summarizing of the paper.

2 Dynamic Input Hash Algorithm:

Input : a message x with the length L bits, $L < 2^{128}$
Output: a 256-bit message digest:$H(x)$

2.1 Pretreatment 1

Build a fixed array p[256] , Take 256 different single bytes as basic units.Put Array p[256] into buffer.See the table of the Array p[256].

2.2 Pretreatment 2

Set 4 64-bit register variables a, b, c, d as working variables , Set 3 64-bit register variables r_1, r_2, r_3 as assistant variables.

2.3 Pretreatment 3

(as the same of SHA512) [1]
Append padding bits:

The message is padded so that its length L_p congruent to 896 modulo1024 [$L_p \equiv 896(mod 1024)$].Padding is always added,even if the message is already of the desired length .The number of padding bits is in the range of 1 to 1024.The padding consists of single 1-bit followed by the necessary number of 0-bits.

Append length: A block of 128 bits is appended to the message.This block is treated as an unsigned 128-bit integer(most significant byte first)and contains the length of the original message (before the padding). The final length L_f of the message is an integer multiple of 1024 .We can write : $L_f \equiv 1024 \times m$

$1024m \equiv 16m \times 64$

The formative input is made up of 16m 64-bit words: $x_0, x_1, x_2, ..., x_{16m-1}$

the Table of the Array p[256]

S	P[S]	S	P[S]	S	P[S]	S	P[S]	S	P[S]	S	P[S]	S	P[S]	S	P[S]
0	147	32	230	64	108	96	120	128	72	160	249	192	0	224	156
1	227	33	47	65	162	97	200	129	1	161	250	193	56	225	114
2	44	34	111	66	59	98	25	130	12	162	179	194	209	226	65
3	116	35	191	67	131	99	213	131	92	163	196	195	50	227	153
4	188	36	8	68	211	100	64	132	172	164	23	196	35	228	68
5	5	37	80	69	37	101	144	133	199	165	16	197	60	229	126
6	77	38	152	70	117	102	102	134	30	166	104	198	148	230	175
7	149	39	224	71	197	103	177	135	63	167	140	199	236	231	161
8	221	40	41	72	14	104	2	136	146	168	228	200	118	232	95
9	38	41	121	73	94	105	100	137	13	169	53	201	79	233	243
10	110	42	193	74	166	106	252	138	93	170	206	202	167	234	133
11	182	43	10	75	246	107	26	139	173	171	98	203	34	235	7
12	254	44	83	76	143	108	106	140	141	172	163	204	122	236	170
13	71	45	155	77	223	109	186	141	229	173	251	205	210	237	137
14	135	46	235	78	48	110	124	142	54	174	88	206	61	238	212
15	207	47	52	79	128	111	21	143	134	175	176	207	232	239	85
16	24	48	132	80	208	112	101	144	214	176	9	208	73	240	15
17	96	49	204	81	33	113	62	145	39	177	89	209	231	241	27
18	168	50	29	82	113	114	142	146	127	178	189	210	76	242	217
19	240	51	109	83	139	115	222	147	58	179	22	211	136	243	187
20	57	52	181	84	219	116	32	148	138	180	42	212	145	244	165
21	129	53	6	85	245	117	112	149	218	181	130	213	241	245	66
22	201	54	78	86	174	118	192	150	4	182	43	214	103	246	45
23	18	55	119	87	105	119	17	151	84	183	159	215	160	247	31
24	90	56	49	88	185	120	97	152	164	184	247	216	248	248	194
25	178	57	82	89	74	121	55	153	244	185	225	217	81	249	19
26	11	58	154	90	226	122	242	154	69	186	220	218	216	250	125
27	99	59	234	91	20	123	67	155	157	187	238	219	205	251	233
28	195	60	51	92	180	124	36	156	237	188	3	220	46	252	190
29	253	61	123	93	150	125	70	157	183	189	91	221	184	253	87
30	86	62	203	94	215	126	151	158	147	190	75	222	202	254	107
31	158	63	28	95	40	127	239	159	169	191	171	223	115	255	198

For example :$p[0] = 147$, $p[8] = 221$

2.4 The Symbols:

$''\leftarrow''$ — assignment or simultaneous assignment

$''\|''$ — cascading

$''$ <<< n $''$ — Rotate Left for n bits

$''$ >>> n$''$ — Rotate Right for n bits

$''+''$ — addition modulo 2^{64}

$''$ $M_{ap}(b, x[j]) \rightarrow r_1$ $''$ — split value $t = (b + x[j])$ to 8-bit byte stream:

$s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8$,(most significant byte first) then cascade the 8-bit byte stream of TLU(table look up) to be a 64-bit word:

$p[s_1] \parallel p[s_2] \parallel p[s_3] \parallel p[s_4] \parallel p[s_5] \parallel p[s_6] \parallel p[s_7] \parallel p[s_8]$ (most significant byte first)

finally assign the 64-bit word value to r_1

The whole process is concise to express with the 64-bit registers ,For example: t and r_1 are taken as Register RBX ,and RDX; BL, DL are the lowest bytes of RBX and RDX ;

$DL \leftarrow p[BL]$; RBX>>> 8; RDX>>> 8 ,Repeat the process for 7 times.

2.5 Initialization Setting

Define 7 64-bit initial chaining values :

$h_1 = 0x6a09e667f3bcc908$; $h_2 = 0xbb67ae8584caa73b$; $h_3 = 0x3c6ef372fe94f82b$

$h_4 = 0xa54ff53a5f1d36f1$;$h_5 = 0x510e537fade682d1$; $h_6 = 0x9b05688c2b3e6c1f$

$h_7 = 0x1f83d9abfb41bd6b$

(These values are come from SHA-512.)

Define 7 64-bit chaining variables and initialize them :

$H_1, H_2, H_3, H_4, H_5, H_6, H_7$

$(H_1, H_2, H_3, H_4, H_5, H_6, H_7) \leftarrow (h_1, h_2, h_3, h_4, h_5, h_6, h_7)$

Define 4 64-bit constants :

$W_1 = 0xcdaa8b436ed9eba1$,$W_2 = 0x6ed9eba18f1bbcdc$,$W_3 = 0x15b49ce581535a99$
$W_4 = 0xf9cf311393b27d54$ *(These values were obtained by taking the first sixty-four bits of the fractional parts of the square roots of the prime numbers.)*

2.6 Operation :

For i from 0 to m-1 ,(The formative input is made up of 16m 64-bit words: $x_0, x_1, x_2, ..., x_i, ..., x_{16m-1}$.) Copy the ith block of 16 64-bit words to Buffer: $X[j] \leftarrow x_{16i+j}$, $0 \le j \le 15$, Operate 16 steps with 4 rounds before the chaining value updated.

Initialize variables :

$(a, b, c, d) \leftarrow (H_1, H_2, H_3, H_4)$
$(r_1, r_2, r_3) \leftarrow (H_5, H_6, H_7)$

The each step of the 4 rounds:for j from 0 to 15 , for n from 1 to 4:
1) $a \leftarrow (a + r_1)$; $t \leftarrow (b + X[j])$

2) $M_{ap}(b, X[j]) \rightarrow r_1$
3) $t \leftarrow (a + \phi_n(r_1, c, d) + W_n + b <<< 1) <<< 5$
4) $(a, b, c, d) \leftarrow (d + r_1 <<< j, t <<< 30, b, c)$;
$t \leftarrow r_1$;
$(r_1, r_2, r_3) \leftarrow (r_2, r_3, t >>> 12)$
Where ϕ_n ,are the boolean functions from MD5:

$\phi_1(X, Y, Z) = (XandY)or((notX)and(Z))$
$\phi_2(X, Y, Z) = XandZor(Yand(notZ)$
$\phi_3(X, Y, Z) = XxorYxorZ$
$\phi_4(X, Y, Z) = Yxor(Xor(not)Z)$

After finished the 4th round, Update the chaining values
$(H_1, H_2, H_3, H_4) \leftarrow (H_1 + a, H_2 + b, H_3 + c, H_4 + d)$
$(H_5, H_6, H_7) \leftarrow (H_5 + r_1, H_6 + r_2, H_7 + r_3)$

The final result $H(x)$:
$(H_1 + H_5) \parallel H_2 \parallel (H_3 + H_6) \parallel (H_4 + H_7)$ (most significant byte first)

2.7 The Step Function

In each step (Step j),The first pass :$a = a + r_1$ the previous value of r_1 was added into the computing,

then the second pass :$M_{ap}(b, X[j]) \rightarrow r_1$, r_1 was updated by the mapping and was added into the computing also.The mapping value we mark it as Q_j

Finally,the fourth pass : $t \leftarrow r_1$;$(r_1, r_2, r_3) \leftarrow (r_2, r_3, t >>> 12)$; r_1 was assigned to r_3 , it would be taken as the ″previous value of r_1 ″ of Step $(j + 3)$, and this made the computing interleaving .
For $1 \leq i \leq 64$,$1 \leq n \leq 4$,the step function can be described as follows:
$b_i = (a_{i-1} + Q_{i-3} >>> 12 + \phi_n(Q_i, c_{i-1}, d_{i-1}) + W_n + (b_{i-1}) <<< 1) <<< 35$
$a_i = d_{i-1} + Q_i <<< j$
$c_i = b_{i-1}$
$d_i = c_{i-1}$
$Q_i = r_1 = M_{ap}(b_{i-1}, m_i)$,where $m_{(i-1)mod16} = x[j]$.
if $1 \leq i \leq 16$, then $n = 1$;
if $17 \leq i \leq 32$, then $n = 2$;
if $33 \leq i \leq 48$, then $n = 3$;
if $49 \leq i \leq 64$, then $n = 4$
The values of Q_0 ,Q_{-1} ,Q_{-2} ,are separately equal to the input values of the chaining variables H_7, H_6, H_5.

And for a hash function of $H(x)$ with the M-D structure ,the compression

function f , (m blocks of message x,$(0 \leq i < m)$)
CV_i = Chaining variable ,
IV_0 = Initial Value ,x_i = the ith block
$CV_0 = IV_0$
$CV_i = f(CV_{i-1}, x_i)$
$H(x) = CV_m$ The hash code of length in DIHA2 is 256.

3 MD5 And The Attack of WANG'S

MD5 was the most widely used cryptographic hash function. It was designed as an improvement version of MD4.Now, its weaknesses has been found,In 2005 WANG present a new powerful attack that can efficiently find a collision of MD5 with differential attack.As the attacks on other functions *are* similar,we observe *the attack* on MD5 only.

The following *are* selected and summarized from WANG's paper [2]:

To describe the compression function for MD5. For each 512-bit block M_i of the padded message M, divide M_i into 32-bit words, $M_i = (m_0, m_1, ..., m_{15})$. The compression algorithm for M_i has four rounds, and each round has 16 operations. Four successive step operations are as follows:

$a = b + ((a + \phi_i(b, c, d) + w_i + t_i) \lll s_i)$
$d = a + ((d + \phi_{i+1}(a, b, c) + w_{i+1} + t_{i+1}) \lll s_{i+1})$
$c = d + ((c + \phi_{i+2}(d, a, b) + w_{i+2} + t_{i+2}) \lll s_{i+2})$
$b = c + ((b + \phi_{i+3}(c, d, a) + w_{i+3} + t_{i+3}) \lll s_{i+3})$

Where the operation + means ADD modulo 2^{32} . $t_{i+j} and s_{i+j} (j = 0, 1, 2, 3)$ are step-dependent constants. w_{i+j} is a message word.$\lll s_{i+j}$ is circularly left-shift by s_{i+j}bit positions.

The differential definition is a kind of precise differential which uses the difference in term of integer modular subtraction. The differential definition is a kind of precise differential which uses the difference in term of integer modular subtraction. The difference for two parameters X and X' is defined as $\Delta X = X' - X$.any two messages M and M' with l-bit multiples, $M = (M_0, M_1, ..., M_{k-1})$, a full differential for a hash function is defined as follows:

$\triangle H_0 \xrightarrow{(M_0, M_0')} \triangle H_1 \xrightarrow{(M_1, M_1')} \triangle H_2 \xrightarrow{(M_2, M_2')} .....\triangle H_{k-1} \xrightarrow{(M_{k-1}, M_{k-1}')} \triangle H$

Consider a message with 2 blocks: where $\triangle H_0$ is the initial value difference which equals to zero. $\triangle H$ is the output difference for the two messages. $\triangle H_i = \triangle IV_i$ is the output difference for the i-th iteration, and also is the initial difference for the next iteration.

It is clear that if $\triangle H = 0$, there is a collision for $M$and M' . We call the differential that produces a collision a collision differential.

1. $M = (m_0, m_1, ..., m_{15})$and $M' = (m_0', m_1', ..., m_{15}')$ represent two 512-bit messages. $\triangle M = (\triangle m_0, \triangle m_1, ..., \triangle m_{15})$ denotes the difference of two message

blocks. That is, $\triangle m_i = \triangle m_i' - \triangle m_i$ is the i-th word difference.

Select a collision differential with two iterations as follows:
$\triangle H_0 \xrightarrow{(M_0, M_0')} \triangle H_1 \xrightarrow{(M_1, M_1')} \triangle H = 0$
Where
$\triangle M_0 = \triangle M_0' - \triangle M_0 = (0,0,0,0,2^{31},0,0,0,0,0,0,2^{15},0,0,2^{31},0)$
$\triangle M_1 = \triangle M_1' - \triangle M_1 = (0,0,0,0,2^{31},0,0,0,0,0,0,-2^{15},0,0,2^{31},0)$
$\triangle H_1 = (2^{31}, 2^{31}+2^{25}, 2^{31}+2^{25}, 2^{31}+2^{25})$
Sufficient Conditions for the Characteristics to Hold

In what follows, we describe how to derive a set of sufficient conditions that guarantee the differential characteristic in Step 8 of MD5 (Table 3 of WANG's paper) to hold Other conditions can be derived similarly. The differential characteristic in Step 8 of MD5 is:

$(\Delta c_2, \Delta d_2, \Delta a_2, \Delta b_1) \rightarrow \Delta b2$. Each chaining variable satisfies one of the following equations. $b_1' = b_1$
$a_2' = a_2[7, ..., 22, -23]$
$d_2' = d_2[-7, 24, 32]$
$c_2' = c_2[7, 8, 9, 10, 11, -12, -24, -25, -26, 27, 28, 29, 30, 31, 32, 1, 2, 3, 4, 5, -6]$
$b_2' = b_2[1, 16, -17, 18, 19, 20, -21, -24]$
According to the operations in the 8-th step, we have
$b_2 = c_2 + ((b_1 + F(c_2, d_2, a_2) + m_7 + t_7) \lll 22$
$b_2' = c_2' + ((b_1 + F(c_2', d_2', a_2') + m_7' + t_7) \lll 22$
$\phi_7 = F(c_2, d_2, a_2) = (c_2 \bigwedge d_2) \bigvee (\neg c_2 \bigwedge a_2)$

In the above operations,c_2 occurs twice in the right hand side of the equation. In order to distinguish the two, let c_2^F denote the c_2 inside F,and c_2^F denote thec_2 outside F. The derivation is based on the following two facts:

1. Since $\Delta b_1 = 0$ and $\Delta m7 = 0$, we know that $\Delta b_2 = \Delta c_2^{NF} + (\Delta_7 \lll 22)$
2. Fix one or two of the variables in F so that F is reduced to a single variable. Guarantee all the differential characteristics in the collision differential to hold:

By the similar method, we can derive a set of sufficient conditions (Table 4 and Table 6 of WANG's paper) which guarantee all the differential characteristics in the collision differential to hold. then ,
1 Repeat the following steps until a first block is found.
(a) Select a random message M_0.
(b) Modify M_0 by the message modification techniques described in the previous subsection.
(c) Then, M_0 and $M_0' = M_0 + \triangle M_0$ produce the first iteration differential.
$\Delta M_0 \rightarrow (\Delta H_1, \Delta M_1)$
2. Repeat the following steps until a collision is found (a) Select a random message M_1.

(b) Modify M_1 by the message modification techniques described in the previous subsection.
(c) Then, $M_1$and$M_1 + \Delta M_1$ generate the second iteration differential
$(\Delta H_1, \Delta M_1) \rightarrow \Delta H = 0$

4 To Overcome The Differential Attack

We know ,the differential attack based on the techniques :
1 To find and built a set of conditions equation for differential .
2 The techniques of modification: Select messages and modify them so as to keep and hold the differential characteristics.

4.1 The Avalanche In DIHA2

In DIHA2,where the $M_{ap}(b, X[j]) \rightarrow r_1$, P[256]. P[256]is a mapping which is corresponding to any probability array of bits.Let's see the avalanche of one-bit difference in DIHA2.

The tiny difference of one bite will *be sent* to the dynamic mapping

$M_{ap}(b, X[i]) \rightarrow r_1$ $(0 \leq i \leq 15)$,and enlarge irregularly to 8-bit extent,then,*the Variable b is replaced by the mapping value (which we gained is marked as* $Q_i = r_1$*) in the Boolean function ,and compution is nonlinearity further*,the difference is spread much more and then the Variable b:
$b_i = (a_{i-1} + Q_{i-3} >>> 12 + \phi_n(Q_i, c_{i-1}, d_{i-1}) + W_n + b_{i-1} <<< 1) <<< 35$

b_i is gained and will be feeded back ,it will *act* on the next dynamic mapping immediatly ... This process is *a* consecutive avalanche,we call this the "first avalanche".The internal state $Q_i = r_1$ will be reinserted 3 step later, and this will cause the "second avalanche ".

4.2 The Differential

We now discuss the attack in DIHA2, first, regard all the words ,variables as single-bytes , so as to simplify the analysis.

For DIHA2,the difference of input $\Delta m_i = m'_i - m_i$ cause the intermediate value $Q_i = r_1$ changed ,we write it*:* $\Delta Q_i = Q'_i - Q_i$

$\Delta m_i \rightarrow \Delta Q_i = Q'_i - Q_i$,
and then ,for more *simplifying*, let $b_{i-1} = 0$:

$Q'_i = M_{ap}(b_{i-1}, m'_i) = M_{ap}(0, m'_i) = P[m'_i]$

As the same , $Q_i = P[m_i]$,so,

$\Delta Q_i = Q'_i - Q_i = P[m'_i] - P[m_i]$

If the difference of input $\Delta m_i = m'_i - m_i$ keep a fixed value N_0,e.g.let $N_0 = 3$,we can see the differenceΔQ_i:

e.g. , when $m_i' = 18$, $m_i = 15$, $\Delta m_i = 18 - 15 = 3$;
$\Delta Q_i = Q_i' - Q_i = P[m_i'] - P[m_i] = P[18] - P[15]$,we can look up to the table to see the value of TLU:
$P[18] = 168$, $P[15] = 207$,so $\Delta Q_i = 168 - 207 = -39$
and change the $m_i' = 18$, $m_i = 15$ optionally,
let $m_i' = 70$, $m_i = 67$
$\Delta Q_i = P[70] - P[67] = 117 - 131 = -14$; and more
.....,

Then,we can know ,that the difference of input$\Delta m_i = m_i' - m_i$ can't cause a regular output of the intermediate valueΔQ_i , for any determinate difference value of input $\Delta m_i = m_i' - m_i$,ΔQ_i can be any possible value.

b_{i-1} is not be zero always. For $Q_i = M_{ap}(b_{i-1}, m_i) = P[b_{i-1} + m_i]$,and
$b_i = (a_{i-1} + Q_{i-3} >>> 12 + \phi_n(Q_i, c_{i-1}, d_{i-1}) + W_n + b_{i-1} <<< 1) <<< 35$

in fact, The no regular value of ΔQ_i was added into the Boolean function,this can't cause a regular value of Δb_i .
$\Delta b_i = b_i' - b_i = (\phi_n(Q_i', c_{i-1}, d_{i-1})) <<< 35 - (\phi_n(Q_i, c_{i-1}, d_{i-1})) <<< 35$

i.e.,the difference of outputΔb_i isn't effectually depend on the difference of inputΔm_i, we can't build a set of conditions equation for the differential of sub-block and not even *for* a message condition [3].

The first prerequisite of a differential attack ,is a set of sufficient conditions for differential characteristics can be found .

Since it is no effectual relational for difference of input and difference of output, to find a set of sufficient conditions for the differential characteristics is invalid work.

For a assured values of output ,the input of $M = (m_0, m_1, ..., m_{15})$ are limited. If to be simple,to eliminate the dynamic mapping effect ,let Q_i be a constantC_N,which $Q_i = P[b_{i-1} + m_i] = C_N$, $b_{i-1} + m_i = C_N^{-1}$, then the $M = (m_0, m_1, ..., m_{15})$ is limited,for:

If m_0 is determined ($a_0, b_0, c_0, d_0, Q_0, Q_{-1}, Q_{-2}$ are known),
then, $b_0 + m_1 = C_N^{-1} \rightarrow m_1$,m_1 is gotten;
$b_1 = (a_0 + Q_{-2} >>> 12 + \phi_1(Q_1, c_0, d_0) + W_1 + b_0 <<< 1) <<< 35$,$b_1$is gotten;
and $b_1 + m_2 = C_N^{-1} \rightarrow m_2$,$m_2$ is gotten;
.....,
the each sub-block of $m_i = (m_0, m_1, ..., m_{15})$ can be determined.

So,in this case,we needn't discuss how to keep the differential characteristics.

Although each assured value of the input determined the value of the output,it is not regular.

4.3 About The Message Modification

The technology of message modification of WANF's meets with difficulty in DIHA2:

If modify a sub-block m_i to m_{ic} so as to achieve a objective,then the mapping $Q_i$of m_i and the mapping Q_{ic} of m_{ic} are different,they will be reinserted 3steps later. To eliminate this change ,need modify the value of m_{i+3} again , and this cause the next modification and so on,.....(if not ,there will be "the second avalanche"),till the last sub-block ,this can't last.

Since the differential characteristics for the conditions in DIHA2 are insignificance ,and because of the strong avalanche ,to find and to hold the differential Characteristics for 4 rounds is unpractical.By the way, Since the strong avalanche and the dynamic mapping in DIHA2, any message modification leads to unexpected change ,even though with low Hamming weight messages.

5 Discussion And Summary

If Q_i is regarded as a part of the ith input word ,then the ith input m_i is regarded as $Q_i + m_i$,Compare with SHA :the input of SHA is expanded from 16 to 80 by the fixed way with Boolean function [4] ,and the DIHA2 each input expanded with the internal state value .

Now,the serious attack ,such as the second Preimage attacks,is based on the achievement of recent attacks on collision resistance of existing hash functions[5].

DIHA2 took the reference form MD5,SHA,etc.,used a table only 256 bytes, obtained the 3properties(Preimage -resistance;Second Preimage -resistance and Collision-resistance). By using dynamic mapping of TLU , inserting and reinserting intermediate value, achieves stronger avalanche to resist various attacks. Anyways,for any other new hash function which with different structure from Merkle-Damgard,it is very difficult to prove it's security is higher than those of actual applications. The Boolean functions DIHA2 used are from MD5,and the other Boolean functions are also useful,but this is not decisive here.

6 References

[1] See also *Cryptography and Network Security : Principles and Practices , Fourth Edition*, William Stallings, Published– PEARSON EDUCATION ASIA LIMITED and Publishing House of Electronics Industry , 2005

[2] See also *How to Break MD5 and Other Hash Functions*, Xiaoyun Wang and Hongbo Yu. http://www.paper.edu.cn

[3] See also *Finding Collisions in the Full SHA-1*,Xiaoyun Wang,YiqunLisaYin, and Hongbo Yu.http://www.paper.edu.cn

[4] See also *Cryptography and Network Security : Principles and Practices , Fourth Edition*, William Stallings,Published– PEARSON EDUCATION ASIA LIMITED and Publishing House of Electronics Industry , 2005

[5] See also *Second Preimage Attacks on Dithered Hash Functions* ,Elena Andreeva1, Charles Bouillaguet, Pierre-Alain Fouque, Jonathan J. Hoch, John Kelsey, Adi Shamir, and Sebastien Zimmer