2007

How to Find the Sufficient Collision Conditions for Haval-128 Pass 3 by Backward Analysis

Pairat Thorncharoensri
University of Wollongong

Tianbing Xia
University of Wollongong, txia@uow.edu.au

Yi Mu
University of Wollongong, ymu@uow.edu.au

How to Find the Sufficient Collision Conditions for Haval-128 Pass 3 by Backward Analysis

Abstract

Wang et al. recently found several collisions in some hash functions, such as MD4, MD5, Haval-128 and RIPEMD. These findings have significantly changed our views about the security of existing hash functions. Unfortunately, al- though it is easy for us to verify the correctness of the collisions published by Wang et al., the sufficient condi- tions for collisions are not clear. In this paper, we present our methodology for constructing the sufficient conditions of collision tables by using Haval-128 Pass 3 as an ex- ample. We propose a backward analysis method of com- pression functions for constructing the sufficient condition table and the differential characteristic table. We also ex- pose the weaknesses of Haval-128 which may be applied to other hash functions.

Disciplines

Physical Sciences and Mathematics

Publication Details

Thorncharoensri, P., Xia, T. & Mu, Y. (2007). How to Find the Sufficient Collision Conditions for Haval-128 Pass 3 by Backward Analysis. International Journal of Network Security, 4 (2), 138-148.

How to Find the Sufficient Collision Conditions for Haval-128 Pass 3 by Backward Analysis

Pairat Thorncharoensri, Tianbing Xia and Yi Mu

(Corresponding author: Pairat Thorncharoensri)

School of IT and Computer Science University of Wollongong
Wollongong, NSW 2522, Australia. (Email: {pt78, txia, ymu}@uow.edu.au)

(Received Oct. 11, 2005; revised and accepted Dec. 6, 2005)

Abstract

Wang *et al.* recently found several collisions in some hash functions, such as MD4, MD5, Haval-128 and RIPEMD. These findings have significantly changed our views about the security of existing hash functions. Unfortunately, although it is easy for us to verify the correctness of the collisions published by Wang *et al.*, the sufficient conditions for collisions are not clear. In this paper, we present our methodology for constructing the sufficient conditions of collision tables by using Haval-128 Pass 3 as an example. We propose a backward analysis method of compression functions for constructing the sufficient condition table and the differential characteristic table. We also expose the weaknesses of Haval-128 which may be applied to other hash functions.

Keywords: Collision in hash functions, cryptography, and hash function

1 Introduction

Hash functions, such as MD5 and SHA1, are widely used in computer and network security. The security of those hash functions was not a major concern until the collisions in MD4, MD5, Haval-128 and RIPEMD were recently reported in the rump session of CRYPTO'04 by Wang *et al.*[4]. Soon after, in [1], the methodology of Wang et al. [4] was explained by analyzing the outputs of [4], where they found a set of conditions and differential behaviors for successfully searching the collision in each step of Haval-128 from the first round to the last round.

Recently, Wang *et al.* revealed more about their findings of the collisions in MD4, MD5 and RIPEMD and reported more results of the collisions in SHA-0 and SHA-1 [5, 6, 7, 8]. They provided a set of sufficient conditions of collisions and differential characteristics for MD4, MD5 and RIPEMD and applied them to single and multi-message modifications. However, they did not reveal the main methodology of constructing the sufficient conditions for searching collisions.

It is also noticed that the recent results of a MD5 collision analysis in [2, 3] show a greater performance in searching the collisions than the original one by Wang *et al.*. The results show that a collision for MD5 can be found on a notebook computer in a few hours. They provided their results with some new conditions and utilized a multi-message modification method to fulfill the condition in the first message and the second message to obtain higher probability in successfully finding a collision. Nevertheless, they provided only the results and the conditions of finding collisions but did not show how to choose the suitable bits and how to construct the sufficient condition for searching the collision table.

In this paper, we for the first time reveal the secret methodology of Wang *et al.*'s findings and show how to find a set of sufficient conditions and differential characteristics for collisions. We propose a backward analysis method of compression functions to construct the sufficient condition table and differential characteristic table. We utilize a raw result from [4] to produce the sufficient condition for the searching collision table of Haval-128 pass 3. As we believe that Haval-128 is easier to break, we choose it as the entry point for finding weak points of hash functions and revealing Wang *et al.*'s methodology of constructing the sufficient condition and the collision table. These weak points lead some information for us to break other hash functions.

This paper is organized as follows. In Section 2, we describe the notations used in this paper. In Section 3, we explain the compression of Haval related to our scheme. In Section 4, we present the methodology used for our analysis and construct a set of sufficient collision conditions and differential characteristics. In Section 5, we provide some examples and explain how to construct the condition table. In the last section, we conclude our paper and describe the future work.

2 Notations

The notations used in this paper are defined in the following.

- M_1 and M_2 are two 1024-bit messages that are expected to produce the same hash value.
- $E_{(M,i)}$ is an expansion function w.r.t the input message M and step i.
- W_i, $W1_{(i)}$, $W2_{(i)}$, or $W_{(i,j)}$ is the input of a 32-bit word of the compression function. It is the output of message expansion $E_{(M,i)}$, where i denotes a step position, for $i = 0, 1, \cdots, 95$ and j denotes a bit position for $j = 0, 1 \cdots, 31$.
- Y_i, $Y1_{(i)}$, or $Y2_{(i)}$ is the output of a 32-bit word of the compression function, where i denotes a step position, for $i = 0, 1, \cdots, 95$ and j denotes a bit position for $j = 0, 1, \cdots, 31$. Note that W and Y are referred to as general terms in the compression function structure and $W1, W2, Y1$, and $Y2$ are referred to as the terms in the result.
- $Y_{-1}, \cdots, Y_{-7}$ is the initial value (IV) or the output of the compression function of the previous message.
- Z or $Z_{(i,j)}$ denotes a carry bit in each step, where i denotes the step position and j denotes the bit position for $j = 0, 1, \cdots, 31$.
- P, $Pp_{(i)}$, $P_{(i,j)}$, or $Pm_{(i,j)}$ denotes a non-linear term, where p denotes a pass number, m denotes the message index, i denotes the step position and j denotes the bit position.
- K or $K_{(p,i)}$ denotes a constant bit, where p denotes a pass number and i denotes the step position.
- BDI or the bit-difference of inputs is the position of the difference of two input messages at the specific word and bit. The value of BDI represents the different between $W1$ and $W2$ which are either 1 or -1. For example, the inputs at step I are $E(M_1, i) = W1_{(i)} =$ 0x1001 and $E(M_2, i) = W2_{(i)} =$ 0x1000. The BDI$_{(i)}$ bit is bit 0 and value is 1.
- BDO or the bit-difference of outputs is the position of the different output bits in the specific step of two input messages and bits. The value of BDO represents the difference between $Y1$ and $Y2$ which are either 1 or -1. For example, the outputs at step i are $Y1_i = H_i(W1_i) =$ 0x1000 and $Y2_i = H_i(W2_i) =$ 0x1001. The BDO$_{(i)}$ bit is bit 0 and the value is -1.
- MSD or the modulo-subtraction difference of outputs is the output of the deference between $Y1_{(i)}$ and $Y2_{(i)}$, computed with modulus 2^{32}. For example, $Y1_{(0)} - Y2_{(0)} = 2^{10}$ or MSD(0) is at bit 10. It can be negative when $Y2_{(i)} > Y1_{(i)}$ and it can also be more than one bit such as MSD$(i) = 2^{22} + 2^{21} =$ bit 22 and bit 21.
- $\boxplus$ denotes an addition on two 32-bit words under modulus 2^{32}.
- $\oplus$ denotes an exclusive or.
- $\wedge$ denotes a logical and.
- ROT(X, j) denotes a rotation by j bits of a 32-bit word X.

3 Compression Function of Haval

The full description of Haval is given in [9]. In this section, we only describe the compression function which is the core of the algorithm.

Each pass of Haval contains 4 rounds and each round of Haval contains 8 steps. Therefore, there are totally 96 for Haval-128 (Pass 3). Three Boolean functions for the nonlinear terms of Pass 1 to 3 are:

$$\begin{aligned}
&F_1(x_6, x_5, x_4, x_3, x_2, x_1, x_0)\\
&\quad = x_1 \wedge x_4 \oplus x_2 \wedge x_5 \oplus x_3 \wedge x_6 \oplus x_0 \wedge x_1 \oplus x_0\\
&F_2(x_6, x_5, x_4, x_3, x_2, x_1, x_0)\\
&\quad = x_1 \wedge x_2 \wedge x_3 \oplus x_2 \wedge x_4 \wedge x_5 \oplus x_1 \wedge x_2 \oplus x_1\\
&\qquad \wedge x_4 \oplus x_2 \wedge x_6 \oplus x_3 \wedge x_5 \oplus x_4 \wedge x_5 \oplus x_0\\
&\qquad \wedge x_2 \oplus x_0\\
&F_3(x_6, x_5, x_4, x_3, x_2, x_1, x_0)\\
&\quad = x_1 \wedge x_2 \wedge x_3 \oplus x_1 \wedge x_4 \oplus x_2 \wedge x_5 \oplus x_3 \wedge x_6\\
&\qquad \oplus x_0 \wedge x_3 \oplus x_0
\end{aligned}$$

The non-linear terms for pass 1 to 3 are:

$$\begin{aligned}
P_1(i) &= F_1(Y_{i-7}, Y_{i-6}, Y_{i-5}, Y_{i-4}, Y_{i-3}, Y_{i-2}, Y_{i-1});\\
&\quad \text{for } i = 0, \cdots, 31,\\
P_2(i) &= F_2(Y_{i-7}, Y_{i-6}, Y_{i-5}, Y_{i-4}, Y_{i-3}, Y_{i-2}, Y_{i-1});\\
&\quad \text{for } i = 32, \cdots, 63,\\
P_3(i) &= F_3(Y_{i-7}, Y_{i-6}, Y_{i-5}, Y_{i-4}, Y_{i-3}, Y_{i-2}, Y_{i-1});\\
&\quad \text{for } i = 64, \cdots, 95.
\end{aligned}$$

The output in each step is:

$$Y_i = ROT(P, 7) \boxplus ROT(Y_{(i-8)}, 11) \boxplus W_i \boxplus K.$$

4 4B Methodology

There are four main methods used in the analysis and the construction of the sufficient condition for searching the collision table. In this section, we will describe all the methods and examples. Although some contents in the first three methods are similar to those used in [2, 3, 5, 6], the most of our results are obtained from our observation in the experiment.

Table 1: Example of Single-bit modification to produce multi-BDO bits(1)

	M_1	M_2	M_1	M_2	M_1	M_2	M_1	M_2	M_1	M_2
Terms	6,3	6,3	6,4	6,4	6,5	6,5	6,6	6,6	6,7	6,7
carry bits				1		1		1	1	2
$Y_{(i-8)}$	0	0	0	0	0	0	1	1	1	1
P (non-linear term)	0	1	1	1	1	1	1	1	1	1
$W_{(input)}$	1	1	0	0	0	0	1	1	1	1
K	0	0	0	0	0	0	0	0	0	0
output($Y_{(i)}$)	1	0	1	0	1	0	1	0	0	1

4.1 Bit-differential Control

Bit-differential control is a method for the generation of the BDO at the necessary position. The differential characteristic in each round can be performed by this method. There are two sub-methods that are used to create a single BDO bit and multiple BDO bits. The purpose of this method is to use the existing BDO or BDI bit to generate the BDO bit and to balance with BDO or BDI in other rounds. This method is utilized to control the differential characteristic in the right position and define the differential characteristic in each round by applying other 3B methods.

4.1.1 Single-bit Modification for Generating One BDO Bit

This method uses a previous BDI bit or BDO bit from a previous step to pass the BDO value and bit to the current step. For some step, it requires one BDO bit for the differential characteristic at that step where it can be used to pass a value to other steps or balance a BDI bit or a BDO bit in other steps. The BDO bit can be a consequent output of a BDI bit or a BDO bit from a liner term or a non-linear term. For example, at round 20 and bit 21,

$$\begin{aligned} Y_{(20,21)} &= W_{(20,21)} \longrightarrow \text{a message input term.} \\ &\boxplus Y_{(12,0)} \longrightarrow \text{ROT}(Y_{(i-8)}, 11) \text{ term.} \\ &\boxplus [\, Y_{(15,28)} \wedge Y_{(17,28)} \oplus Y_{(14,28)} \wedge Y_{(18,28)} \\ &\quad \oplus Y_{(13,28)} \wedge Y_{(19,28)} \oplus Y_{(16,28)} \wedge Y_{(17,28)} \\ &\quad \oplus Y_{(15,28)} \,] \longrightarrow \text{the set of XOR terms} \\ &\qquad\qquad \text{is a nonlinear term } (P) \\ &\boxplus K \longrightarrow \text{these terms are constant } (K) \end{aligned}$$

If $Y_{(20,31)}$ dominates the BDO bit, then either $Y_{(12,0)}$ or one of Y terms in P is BDO. If $Y_{(20,31)}$ dominates BDI bit, then $W_{(20,31)}$ is a BDI bit. The equation above is split into two groups: the nonlinear term and the group of additional modulus 2^{32} terms. The nonlinear term is also one of the additional modulus 2^{32} terms but we review it separately because it reveals the secret for balancing or controlling the differential characteristic or the BDO bit in every step. In addition, the equation above shows that the distance between the previous BDO bit and the current BDO bit are 8 steps. This discloses one point of the constructing differential characteristics and it is used to pass the value of BDO or BDI from one round to next round (8 steps). It can be found from the differential characteristics in Table 8 in Appendix A.

4.1.2 Single-bit Modification to Produce Multi-BDO Bits

This method is used to produce several continuing bits or non-continuing bits of BDO while MSD appears only one bit or two bit positions. It considers only the terms used in modulation. For example, Table 1 and Table 2 show the terms and values of each bit from $Y_{(6,3)}$ to $Y_{(6,31)}$. Assume that we want $Y_{(6,3)}$ to $Y_{(6,7)}$ to dominate BDO and we have only one BDO bit from the non-linear term at $Y_{(6,3)}$. Therefore, we adjust other terms to change the carry bit value to only 1, move the BDO bit spreading from $Y_{(6,3)}$ through $Y_{(6,7)}$, and control the carry bit at $Y_{(6,7)}$ for M_1 equal to 1 and M_2 equal to 2. The above condition will make the BDO stop moving forward through other bits and we need to keep the difference of carry bits of M_2 and M_1 to 2 from $Y_{(6,8)}$ to $Y_{(6,31)}$. The carry bit will be dropped at $Y_{(6,31)}$. This technique has been used on [6].

The purpose of the above two methods is to generate the useful BDO bit for the next few steps or next rounds. Besides, the result of each bit will spread out though other bits in other rounds. However, they can be easily balanced with the different bit that spreads out in the next two methods.

4.2 Balance a Consequence of BDO Bit or BDI Bit in a Nonlinear Term

There are two methods to balance a consequence of BDO bit or BDI bit in a non-liner term. They are used in the different situation. The first method is used when BDO bit(s) is (are) a partial multiple term ("and" term) and no single BDO bit term in the nonlinear term. The other method is used when there are more than one BDO bit in the nonlinear term.

Table 2: Example of Single-bit modification to produce multi-BDO bits(2)

	M_1	M_2	M_1	M_2		M_1	M_2
Terms	6,8	6,8	6,9	6,9	to	6,31	6,31
carry bits		2		2			2
$Y_{(i-8)}$	1	1	1	1		1	1
P (non-linear term)	1	1	1	1		1	1
$W_{(input)}$	0	0	1	1		0	0
K	0	0	0	0		0	0
output($Y_{(i)}$)	0	0	1	1		0	0

4.2.1 Balance BDO Bit by Balancing One or the Entire Terms that Multiple with BDO Bit

This method is used at the output bit (Y) which is not necessary to be a BDO bit. The output bit does not contain a BDO bit or a BDI bit of a single linear term.

Example 1. *The nonlinear terms may contain*

$$Y_{(0,10)} \wedge Y_{(0,17)} \oplus Y_{(1,10)} \wedge Y_{(2,10)}.$$

The output bit is not required to be a BDO bit for the output of the next 8 steps. To correct the output for the above term, we must set $Y_{(0,17)} = 0$ *when* $Y_{(0,10)}$ *is a BDO bit.*

Example 2. *The nonlinear terms may contain*

$$Y_{(0,10)} \wedge Y_{(1,17)} \oplus Y_{(0,10)} \wedge Y_{(0,17)} \oplus Y_{(1,10)} \wedge Y_{(2,10)}.$$

The condition to correct the output of the above terms is $Y_{(0,17)}$ *and* $Y_{(1,17)}$ *must be the same to balance* $Y_{(0,10)} \wedge Y_{(1,17)} \oplus Y_{(0,10)} \wedge Y_{(0,17)}$*. The output of these two terms is* 0 *in the XOR process.*

4.2.2 Balance BDO Bit by Other BDO Bits

This method is used when there are two terms or more of BDO bits at the same output word. We can simply adjust some bits form 0 to 1 to make those different bits balanced.

For example, $Y_{(0,10)}$ is a BDO bit and the non-linear term is $Y_{(0,10)} \oplus Y_{(1,10)} \wedge Y_{(6,3)} \oplus Y_{(1,10)}$. To correct the output of the above term, we set $Y_{(1,10)} = 1$ to balance $Y_{(0,10)}$ and $Y_{(6,3)}$. If there are more than 2 BDO bits in non-linear term, such as $Y_{(0,10)} \oplus Y_{(1,10)} \wedge Y_{(6,3)} \oplus Y_{(0,10)} \wedge Y_{(0,17)}$, then we set $Y_{(1,10)} = 1$ or $Y_{(0,17)} = 1$ to balance $Y_{(0,10)}$ and $Y_{(6,3)}$.

4.3 Balance a Consequent BDO Bit or BDI Bit in a Linear Term of Additional Modulo 2^{32} Process

There are three methods to correct the output from BDO bit or BDI bit in the linear term of the additional modulus 2^{32} process. They are used in different situations. The first method is used in the situation that a sequence of BDO bits appear in the output such as $Y_{(6,3)}$ to $Y_{(6,7)}$. The second method is used when the position of the bit occurs at the BDO bit is near bit 31 that is easy to eliminate the carry bit. Finally, it is used when there are two or more BDO bits or BDI bits in the same output bit. This method is frequently used in Pass 2 and Pass 3 which avoid to occur at the BDO bit in the most steps in Pass 2 and Pass 3.

4.3.1 Move a BDO Bit Forward to Balance with Other BDO Bits

Assume there are two BDO bits at the same step and the BDO value of those bits are opposite each other. For example in Step i, $Y1_i$ is 0x1A1 and $Y2_i$ is 0xA9. The BDO of $Y1_i$ and $Y2_i$ appear at bit 4 and bit 9 whose values are 1 and -1, respectively. In order to balance $Y1_i$ such that it equals $Y2_i$. We need to perform the BDO at bit 4 to move the carry bit to bit 9. This method can be used when there are two or more terms of BDO or BDI in the additional modulo 2^{32} process in which one may be from the non-linear term and the other is from the linear term. For example, assume that $Y_{(0,10)}$ is a BDO bit in the non-linear term and $W_{(8,31)}$ is a BDI bit in the linear term (the message input term). At bit 4, the non-linear term may contain $Y_{(0,10)} \oplus Y_{(1,10)} \wedge Y_{(0,10)} \oplus Y_{(1,10)} \wedge Y_{(2,10)}$. The requirement for this term is that the output of this term must be equal to the BDO value 1. Therefore, this $P_{(i,4)}$ (non-linear term) contains two of $Y_{(0,10)}$ and only $Y_{(1,10)} = 0$ will make that the BDO value of this term becomes 1. Therefore, $P_{(i,4)}$ of this term and $W_{(8,31)}$ will produce a carry bit in output at bit 5. If we want to force the different bit in output to move forward to bit 9, we need at least 2 carry bits in $Y_{(i,4)}$to $Y_{(i,8)}$ to move BDO bit to balance other BDO bits at bit 9 (it may be from $P_{(I,9)}$). In Table 3, an example has been shown in detail.

4.3.2 Balance One BDO Bit by Moving the BDO Bit Forward to The Last Carry Bit

This method can be used when there are two terms that contain two or more terms of BDO or BDI in the additional modulus 2^{32} process. It is the same as the previous

Table 3: Example of moving a BDO bit forward to balance with other BDO bits

	M_1	M_2	M_1	M_2	M_1	M_2	M_1	M_2	M_1	M_2	M_1	M_2
Terms	i,4	i,4	i,5	i,5	i,6	i,6	i,7	i,7	i,8	i,8	i,9	i,9
carry bits				2		2		2		2		1
$Y_{(i-8)}$	1	1	1	1	0	0	1	1	0	0	1	1
P (non-linear term)	0	1	1	1	0	0	1	1	0	0	1	0
$W_{(input)}$	0	1	1	1	1	1	1	1	0	0	1	1
K	1	1	0	0	1	1	0	0	0	0	0	0
output($Y_{(i)}$)	0	0	1	1	0	0	1	1	0	0	1	1

Table 4: Example of balancing one BDO bit by moving the BDO bit forward to the last carry bit

	M_1	M_2	M_1	M_2	M_1	M_2	M_1	M_2	M_1	M_2
Terms	i,27	i,27	i,28	i,28	i,29	i,29	i,30	i,30	i,31	i,31
carry bits				2		2		2		2
$Y_{(i-8)}$	0	1	1	1	0	0	1	1	1	1
P (non-linear term)	0	1	1	1	0	0	1	1	1	1
$W_{(input)}$	1	1	1	1	1	1	1	1	0	0
K	1	1	0	0	1	1	0	0	0	0
output($Y_{(i)}$)	0	0	1	1	0	0	1	1	0	0

method but there is only one BDO bit in output Y_i. It is better that their positions are near bit 31 so that dropping the BDO bit with a carry bit is easy. The example on Table 4 assumes that $Y_{(I,27)}$ has two terms of BDO bits.

4.3.3 Balance Two Terms of BDO or BDI in The Additional Modulus 2^{32} Process

This method has a similar requirement to that in the above method but only the non-linear term can simply balance the BDO bits without producing a carry bit. Moreover, this method can be applied to more than one BDO bits at the same Y_i but all BDO bits must meet the requirement. This method uses a simple way to change the condition in the non-linear term to balance with other BDO bits in other terms, but it requests at least 1 BDO term and one non-BDO bit term in the non-linear term to adjust the BDO value opposite to other BDO bits or BDI bits. For example, assume that the BDI value is 1 at bit 27 and $P_{(I,27)}$ is $Y_{(0,10)} \oplus Y_{(1,10)} \wedge Y_{(0,10)} \oplus Y_{(0,10)} \wedge Y_{(2,10)} \oplus Y_{(3,10)}$. The BDO value at $P_{(I,27)}$ will be -1 and the even value of $Y_{(0,10)}$ is 1 when both of $Y_{(0,10)}$ and $Y_{(2,10)}$ are either 1 or 0. The result of $P1_{(I,27)}$ will be 0 and $P2_{(I,27)}$ will be 1. Moreover, If two of BDO bits or BDI bits in the additional modulus 2^{32} process are not a non-linear term but opposite to the BDO or BDI value, it can also be used to balance this output. The result is given in Table 5 where $Y_{(I,27)}$ balances the output bit with the non-linear term and $Y_{(I,28)}$ balances the output bit with the BDO bit and the BDI bit of the linear term.

Table 5: Example of balancing two terms of BDO or BDI in the additional modulus 2^{32} process

	M_1	M_2	M_1	M_2
Terms	i,27	i,27	i,28	i,28
carry bits			1	1
$Y_{(i-8)}$	1	1	1	0
P (non-linear term)	0	1	1	1
$W_{(input)}$	1	0	0	1
K	1	1	0	0
output($Y_{(i)}$)	0	0	1	1

4.4 Backward Analysis

The previous methods described in this paper will be applied to this analysis for constructing the sufficient condition of searching a collision table. The analysis used to find the collision of the hash function in this paper is called the modulation differential backward analysis. This method is an analysis from the last round by preventing the BDI bits from influencing other bits in other rounds and moving the analysis from last round back to the first round. It is also used to define which bit should be BDO bits in next 8 steps starting from the last round and also which bit should be BDI bits from the relationship among BDI bits. The main concept of backward analysis in this paper is to assign BDO bits and BDI bits on the appropriate step and construct the condition to hold the highest probability of a local collision. This can be explained in the following along with an example in Table 6.

Table 6: Example of backward analysis

Step	Pass	Input (W)	BDI /BDO	Comment
0	1	$W_{(0)}$	$BDI_{(0,10)}$	
1	1			
2	1			
...	1			
31	1			
32	2		$BDO_{(32,21)}$	
...	2			
40	2	$W_{(0)}$	$BDI_{(0,10)}$	BDI bit of step is balance with $BDO_{(32,21)}$ from step 32
32				
...	2			
63	2			
64	3			
...	3			
85	3	$W_{(0)}$	$BDI_{(0,10)}$	the $BDO_{(85,10)}$ is a consequent output for next 8 steps. This step is the first step to analyze.
86	3	$W_{(18)}$	$BDI_{(18,3)}$	$BDI_{(18,3)}$ use to balance with $BDO_{(85,10)}$ which result is from $BDI_{(0,10)}$
...	3			
96	3			

4.4.1 Choosing the First BDI Bit

One BDI bit has to be decided for initial backward analysis. Choosing the BDI bit is a challenge for this method. From the experiment of Haval, we found that many of BDI bits can be chosen; however, the probability of a successful collision of each chosen BDI bit varies. Therefore, this paper uses the BDI bit designed in [4] as the priority BDI bit. The example of the first BDI bit showed in Table 6 is from the input word $W_{(0)}$.

4.4.2 Controlling the Influence of BDI in Forward and Backward 8 Steps

In the step that the output possesses a BDI bit or a BDO bit, the 3B methods are used to control the difference of output and to construct the sufficient conditions for next 8 steps. Therefore, the outputs in next 8 steps will remain the same when two input messages are different. This requirement applies to every pass, every round and every step by starting from the first BDI bit at the last Pass. Therefore, step 85 in Table 6 is the first step starting the backward analysis. Furthermore, the first BDI bit is used to allocate a BDO bit in previous 8 steps, for example, in Table 6, the BDO bit from step 32 is required to balance with $BDI_{(0,10)}$ in step 40. The first BDI bit, nevertheless, is also used to assign other BDI bits when it needs to. For example, in step 86, in Table 6, a BDI bit is designed to balance $BDO_{(85,10)}$ (output from $BDI_{(0,10)}$) from step 85.

4.4.3 Backward Analysis

The analysis step is moving backwards starting from the last round to the first round. The backward analysis is used to force the output at the last round (the last 8 steps) has a collision without concerning the difference of the output in the previous rounds. Besides, in pass three, we use BDI bits to balance themselves, as in Table 6 where $BDI_{(18,3)}$ in Step 86 balances with the output related to $BDI_{(0,10)}$ in step 85. It is also in pass two which it is used $BD0_{(32,21)}$ to balance with $BDI_{(0,10)}$ in step 40. All steps in pass two and pass three, hence, result in some local collisions.

In Addition, the analysis result has shown that there are two weaknesses in the most hash functions. Firstly, the change of the message is not spread out through every round as it can be seen in the experiment section. Secondly, the balance between the non-linear terms and linear terms is important to distribute the changed bit of the input to the entire process. The analysis result of Haval algorithm shows that only the bit containing one linear term of the changed bit of the input affects other bits.

5 Experiment

5.1 Experiment Process

The experiment process starts with the structure of the compression function for every step from step 0 to step 95 of Haval-128 (Pass 3). We then apply the backward anal-

Table 7: Example of experiment process

Round (Y)	Bit	M_1	M_2	BDO value	MSD	relationship with BDO in
40	10	1	0	1		$W_{(0,10)}$(BDI)
36	31	0	1	-1		$W_{(11,31)}$(BDI)
35	3	0	1	-1		$W_{(18,3)}$ (BDI)
32	21	1	0	1	2^21	$Y_{(40,10)}$
28	10	0	1	-1	-2^10	$Y_{(35,3)}$,$Y_{(36,31)}$
24	3	1	0	1		
24	2	0	1	-1		
24	1	0	1	-1		
24	0	0	1	-1	2^0	$Y_{(32,21)}$
20	21	0	1	-1	-2^21	$Y_{(28,10)}$
16	11	1	0	1	2^11	$Y_{(24,0)}$
12	0	0	1	-1	-2^0	$Y_{(20,21)}$
8	21	1	0	1		
8	22	1	0	1	2^21+2^22	$Y_{(16,11)}$
6	3	1	0	1		
6	4	1	0	1		
6	5	1	0	1		
6	6	1	0	1		
6	7	0	1	-1	-2^3	$Y_{(12,0)}$

ysis along with other methods mentioned in the preceding section. The experiment utilizes the result of Wang *et al.* [4] to construct the sufficient condition for searching a collision table. Four bits dominate as BDI bits: $W_{(0,10)}$, $W_{(0,11)}$, $W_{(11,31)}$, and $W_{(18,3)}$, for example, in the experiment, starting from bit 10 in step 85 (the message input is $W_{(0,10)}$), which is the chosen bit to dominate as BDI bit. There are two ways to make a collision in the last round. The first one is to balance this bit output and the second is to leave this bit to dominate the BDO bit and balance the other bit. We first follow the result from [6] so that we leave this to dominate BDO bit and balance all the consequent results from this bit. The interesting results for the next 8 steps are in steps 86, 87 and 93. The following shows the part of the compression function terms at bit 3 in step 86, bit 3 in step 87 and bit 31 in step 93, and reasons why and how to balance these three bits.

$Y_{(85,10)}$ dominates BDO bit.

$Y_{(86,3)}$ contains $W_{(18,3)}$.

$P_{(86,3)}$ contains $Y_{(82,10)} \wedge Y_{(85,10)} \oplus Y_{(85,10)}$.

$P_{(87,3)}$ contains $Y_{(82,10)} \wedge Y_{(85,10)}$.

The two non-linear terms in steps 86 and 87 are related. To balance $Y_{(86,3)}$, $Y_{(82,10)}$ must be 1 but $P_{(87,3)}$ will need other BDO bits to balance. On the other hand, to balance $Y_{(87,3)}$, $Y_{(82,10)}$ must be 0. Therefore, this result shows the reason why Wang *et al.* chose $W_{(18,3)}$ to dominate as the BDI bit.

$Y_{(93,31)}$ contains $W_{(11,31)} \boxplus YR_{(85,10)}$.

To balance $Y_{(93,31)}$, $W_{(11,31)}$ should be a BDI bit with value 1, where it is used to balance $W_{(0,10)}$. This result clearly shows the reason of dominated $W_{(11,31)}$ to BDI.

The experiment is carried out in backwards to construct the entire sufficient condition for searching the collision table. Table 7 shows a part of the relationship between each BDO and BDI in the related steps.

5.2 Additional Explanation of 4B Method in The Experiment

Table 7 explains how to apply the other 3B methods (except the backward analysis) to obtaining the sufficient condition for searching the collision table. The bit-differential control method is used to propagate a BDO bit to the bit that should be a BDO bit. For example, from $Y_{(6,3)}$ to $Y_{(6,7)}$, the BDO bit from $Y_{(0,10)}$ actually only appears in $Y_{(6,3)}$. However, it is required that $Y_{(6,7)}$ must be a BDO bit to make $Y_{(12,0)}$ dominate BDO bit too. Therefore, we use this method to make BDO bit appear from $Y_{(6,3)}$ to $Y_{(6,7)}$. This is a trick to reuse its own BDI bit ($Y_{(0,10)}$) to balance itself or other BDI bits in other steps which can be seen clearly from the full relationship table in Appendix A. The method of balancing different bit in non-liner term and the method of balancing a different bit in linear term are used to eliminate the BDO bit from the other bit that does not have to be a BDO bit. For example, the result of $Y_{(6,3)}$ to $Y_{(6,7)}$ should be spread to $Y_{(12,0)}$ to $Y_{(12,4)}$, but the BDO bit only appears in $Y_{(12,0)}$ because $Y_{(12,1)}$ to $Y_{(12,4)}$ was eliminated with these two methods. Moreover, these two methods are the

Table 8: The differential characteristics for Haval–128 when i=11

Step	Pass	bit diffs	message change (BDI)	MDO	BDO bit
0	1	400	bit change at $2^{10}+2^{11}$	: -2^{10}	[-10]
6	1	8		: -2^{3}	[3,4,5,6,-7]
7	1	10000000		:-2^{28}	[28,29,30,-31]
8	1	600000		: $2^{22}+2^{21}$	[22,21]
10	1	4000		: 2^{14}	[14]
11	1	0	bit change at 2^{31}		
12	1	1		: -2^{0}	[-0]
15	1	20000		:-2^{17}	[-17]
16	1	800		: 2^{11}	[11]
18	1	0	bit change at 2^{3}		
20	1	200000		: -2^{21}	[-21]
23	1	40		: -2^{6}	[-6]
24	1	1		: 2^{0}	[-0,-1,-2,3]
28	1	400		: -2^{10}	[-10]
32	2	200000		: 2^{21}	[21]
35	2	0	bit change at 2^{3}		
36	2	0	bit change at 2^{31}		
40	2	0	bit change at $2^{10}+2^{11}$		
85	3	400	bit change at $2^{10}+2^{11}$	:-2^{10}	[-10]
86	3	0	bit change at 2^{3}		
93	3	0	bit change at 2^{31}		

Table 9: The differential characteristics for Haval–128 when i=10

Step	Pass	bit diffs	message change (BDI)	MDO	BDO bit
0	1	200	bit change at $2^{9}+2^{10}$	: -2^{9}	[-9,]
6	1	4		:-2^{2}	[2,3,4,5,-6]
7	1	8000000		:-2^{27}	[27,28,29,-30]
8	1	300000		: $2^{20}+2^{21}$	[20,21]
10	1	2000		: 2^{13}	[13]
11	1	0	bit change at 2^{30}		
12	1	80000000		: -2^{31}	[-31]
15	1	10000		:-2^{16}	[-16]
16	1	C00		: $2^{10}+2^{11}$	[10,11]
18	1	0	bit change at 2^{2}		
20	1	100000		: -2^{20}	[-20]
23	1	20		: -2^{5}	[-5]
24	1	80000001		: $2^{31}+2^{0}$	[-0,-1,2,31]
28	1	200		: -2^{9}	[-9]
32	2	100000		: 2^{20}	[20]
35	2	0	bit change at 2^{2}		
36	2	0	bit change at 2^{30}		
40	2	0	bit change at $2^{9}+2^{10}$		
85	3	400	bit change at $2^{9}+2^{10}$	:-2^{10}	[-9]
86	3	0	bit change at 2^{2}		
93	3	0	bit change at 2^{30}		

Table 10: A set of sufficient conditions for differential characteristics of Haval–128 when i=11

Y_0	$Y_{(0,3)} = 1$, $Y_{(0,4)} = 0$, $Y_{(0,5)} = 0$, $Y_{(0,6)} = 0$, $Y_{(0,7)} = 0$
Y_1	$Y_{(1,10)} = 0$, $Y_{(1,28)} = 1$, $Y_{(1,29)} = 1$, $Y_{(1,30)} = 0$, $Y_{(1,31)} = 0$
Y_2	$Y_{(2,10)} = 1$, $Y_{(2,21)} = 0$, $Y_{(2,22)} = 0$, $Y_{(2,3)} = 0$, $Y_{(2,4)} = 0$, $Y_{(2,5)} = 0$, $Y_{(2,6)} = 1$, $Y_{(2,7)} = 0$
Y_3	$Y_{(3,28)} = 0$, $Y_{(3,29)} = 0$, $Y_{(3,30)} = 0$, $Y_{(3,31)} = 0$
Y_4	$Y_{(4,10)} = 1$, $Y_{(4,21)} = 1$, $Y_{(4,22)} = 0$, $Y_{(4,3)} = Y_{(5,3)}$, $Y_{(4,4)} = Y_{(5,4)}$, $Y_{(4,5)} = Y_{(5,5)}$, $Y_{(4,6)} = Y_{(5,6)}$, $Y_{(4,7)} = Y_{(5,7)}$
Y_5	$Y_{(5,28)} = Y_{(6,28)}$, $Y_{(5,29)} = Y_{(6,29)}$, $Y_{(5,3)} = Y_{(4,3)}$, $Y_{(5,30)} = Y_{(6,30)}$, $Y_{(5,31)} = Y_{(6,30)}$, $Y_{(5,4)} = Y_{(4,4)}$, $Y_{(5,5)} = Y_{(4,5)}$, $Y_{(5,6)} = Y_{(4,6)}$, $Y_{(5,7)} = Y_{(4,7)}$
Y_6	$Y_{(6,0)} = 0$, $Y_{(6,10)} = 0$, $Y_{(6,21)} = Y_{(7,21)}$, $Y_{(6,22)} = Y_{(7,22)}$, $Y_{(6,28)} = Y_{(5,28)}$, $Y_{(6,29)} = Y_{(5,29)}$, $Y_{(6,30)} = Y_{(5,30)}$, $Y_{(6,31)} = Y_{(5,30)}$
Y_7	$Y_{(7,21)} = Y_{(6,21)}$, $Y_{(7,22)} = Y_{(6,22)}$, $Y_{(7,3)} = 0$, $Y_{(7,4)} = 0$, $Y_{(7,5)} = 0$, $Y_{(7,6)} = 0$, $Y_{(7,7)} = 0$
Y_8	$Y_{(8,0)} = 0$, $Y_{(8,28)} = 0$, $Y_{(8,29)} = 0$, $Y_{(8,3)} = 1$, $Y_{(8,30)} = 0$, $Y_{(8,31)} = 0$, $Y_{(8,4)} = 1$, $Y_{(8,5)} = 1$, $Y_{(8,6)} = 0$, $Y_{(8,7)} = 1$
Y_9	$Y_{(9,17)} = 1$, $Y_{(9,21)} = 0$, $Y_{(9,22)} = 0$, $Y_{(9,28)} = 1$, $Y_{(9,29)} = 1$, $Y_{(9,30)} = 1$, $Y_{(9,31)} = 1$
Y_{10}	$Y_{(10,0)} = Y_{(11,0)}$, $Y_{(10,11)} = 0$, $Y_{(10,21)} = 1$, $Y_{(10,22)} = 1$, $Y_{(10,3)} = 0$, $Y_{(10,4)} = 0$, $Y_{(10,5)} = 0$, $Y_{(10,6)} = 0$, $Y_{(10,7)} = 1$
Y_{11}	$Y_{(11,0)} = Y_{(10,0)}$, $Y_{(11,17)} = 0$, $Y_{(11,28)} = 0$, $Y_{(11,29)} = 0$, $Y_{(11,30)} = 0$, $Y_{(11,31)} = 0$
Y_{12}	$Y_{(12,11)} = 0$, $Y_{(12,21)} = 0$, $Y_{(12,22)} = 0$, $Y_{(12,3)} = 0$, $Y_{(12,4)} = 0$, $Y_{(12,5)} = 0$, $Y_{(12,6)} = 0$, $Y_{(12,7)} = 0$
Y_{13}	$Y_{(13,0)} = 0$, $Y_{(13,17)} = Y_{(14,17)}$, $Y_{(13,28)} = 0$, $Y_{(13,29)} = 0$, $Y_{(13,30)} = 0$, $Y_{(13,31)} = 1$
Y_{14}	$Y_{(14,0)} = 1$, $Y_{(14,11)} = Y_{(15,11)}$, $Y_{(14,17)} = Y_{(13,17)}$, $Y_{(14,21)} = 0$, $Y_{(14,21)} = 0$, $Y_{(14,22)} = 0$
Y_{15}	$Y_{(15,11)} = Y_{(14,11)}$
Y_{16}	$Y_{(16,0)} = 0$, $Y_{(16,17)} = 0$, $Y_{(16,21)} = 0$
Y_{17}	$Y_{(17,11)} = 0$, $Y_{(17,17)} = 1$, $Y_{(17,6)} = 0$
Y_{18}	$Y_{(18,0)} = 0$, $Y_{(18,0)} = 0$, $Y_{(18,1)} = 0$, $Y_{(18,11)} = 1$, $Y_{(18,2)} = 0$, $Y_{(18,21)} = Y_{(19,21)}$, $Y_{(18,3)} = 0$
Y_{19}	$Y_{(19,17)} = 0$, $Y_{(19,21)} = Y_{(18,21)}$, $Y_{(19,6)} = 0$
Y_{20}	$Y_{(20,0)} = 0$, $Y_{(20,1)} = 0$, $Y_{(20,11)} = 0$, $Y_{(20,2)} = 0$, $Y_{(20,3)} = 0$
Y_{21}	$Y_{(21,17)} = 0$, $Y_{(21,21)} = 0$, $Y_{(21,6)} = Y_{(22,6)}$
Y_{22}	$Y_{(22,0)} = Y_{(23,0)}$, $Y_{(22,1)} = Y_{(23,1)}$, $Y_{(22,10)} = 0$, $Y_{(22,11)} = 0$, $Y_{(22,2)} = Y_{(23,2)}$, $Y_{(22,21)} = 1$, $Y_{(22,3)} = Y_{(23,3)}$, $Y_{(22,6)} = Y_{(21,6)}$
Y_{23}	$Y_{(23,0)} = Y_{(22,0)}$, $Y_{(23,1)} = Y_{(22,1)}$, $Y_{(23,2)} = Y_{(22,2)}$, $Y_{(23,3)} = Y_{(22,3)}$
Y_{24}	$Y_{(24,10)} = 0$, $Y_{(24,21)} = 0$, $Y_{(24,6)} = 0$
Y_{25}	$Y_{(25,0)} = 0$, $Y_{(25,1)} = 0$, $Y_{(25,2)} = 0$, $Y_{(25,3)} = 0$, $Y_{(25,6)} = 1$
Y_{26}	$Y_{(26,0)} = 1$, $Y_{(26,1)} = 1$, $Y_{(26,10)} = 0$, $Y_{(26,2)} = 1$, $Y_{(26,21)} = 0$, $Y_{(26,3)} = 1$
Y_{27}	$Y_{(27,10)} = 0$, $Y_{(27,21)} = 0$, $Y_{(27,6)} = 0$
Y_{28}	$Y_{(28,0)} = 0$, $Y_{(28,1)} = 0$, $Y_{(28,2)} = 0$, $Y_{(28,21)} = 1$, $Y_{(28,3)} = 0$
Y_{29}	$Y_{(29,10)} = 0$, $Y_{(29,21)} = 1$, $Y_{(29,6)} = 0$
Y_{30}	$Y_{(30,0)} = 0$, $Y_{(30,1)} = 0$, $Y_{(30,10)} = 0$, $Y_{(30,2)} = 1$, $Y_{(30,21)} = 0$, $Y_{(30,3)} = 0$
Y_{31}	$Y_{(31,10)} = 1$, $Y_{(31,21)} = 0$
Y_{32}	$Y_{(32,10)} != Y_{(33,10)}$
Y_{33}	$Y_{(33,10)} != Y_{(32,10)}$, $Y_{(33,21)} = 1$
Y_{34}	$Y_{(34,21)} = 0$
Y_{35}	$Y_{(35,21)} = Y_{(36,21)}$
Y_{36}	$Y_{(36,21)} = Y_{(35,21)}$
Y_{37}	$Y_{(37,21)} = 0$
Y_{82}	$Y_{(82,10)} = 0$
Y_{83}	$Y_{(83,10)} = 1$
Y_{84}	$Y_{(84,10)} = 0$
Y_{86}	$Y_{(86,10)} = 1$
Y_{87}	$Y_{(87,10)} = 0$
Y_{88}	$Y_{(88,10)} = 0$

Table 11: A set of sufficient conditions for differential characteristics of Haval–128 when i=10

Y_0	$Y_{(0,2)} = 1$, $Y_{(0,3)} = 0$, $Y_{(0,4)} = 0$, $Y_{(0,5)} = 0$, $Y_{(0,6)} = 0$
Y_1	$Y_{(1,9)} = 0$, $Y_{(1,27)} = 1$, $Y_{(1,28)} = 1$, $Y_{(1,29)} = 0$, $Y_{(1,30)} = 0$
Y_2	$Y_{(2,9)} = 1$, $Y_{(2,20)} = 0$, $Y_{(2,21)} = 0$, $Y_{(2,2)} = 0$, $Y_{(2,3)} = 0$, $Y_{(2,4)} = 0$, $Y_{(2,5)} = 1$, $Y_{(2,6)} = 0$
Y_3	$Y_{(3,27)} = 0$, $Y_{(3,28)} = 0$, $Y_{(3,29)} = 0$, $Y_{(3,30)} = 0$
Y_4	$Y_{(4,9)} = 1$, $Y_{(4,20)} = 1$, $Y_{(4,21)} = 0$, $Y_{(4,2)} = Y_{(5,2)}$, $Y_{(4,3)} = Y_{(5,3)}$, $Y_{(4,4)} = Y_{(5,4)}$, $Y_{(4,5)} = Y_{(5,5)}$, $Y_{(4,6)} = Y_{(5,6)}$
Y_5	$Y_{(5,27)} = Y_{(6,27)}$, $Y_{(5,28)} = Y_{(6,28)}$, $Y_{(5,2)} = Y_{(4,2)}$, $Y_{(5,29)} = Y_{(6,29)}$, $Y_{(5,30)} = Y_{(6,29)}$, $Y_{(5,3)} = Y_{(4,3)}$, $Y_{(5,4)} = Y_{(4,4)}$, $Y_{(5,5)} = Y_{(4,5)}$, $Y_{(5,6)} = Y_{(4,6)}$
Y_6	$Y_{(6,31)} = 0$, $Y_{(6,9)} = 0$, $Y_{(6,20)} = Y_{(7,20)}$, $Y_{(6,21)} = Y_{(7,21)}$, $Y_{(6,27)} = Y_{(5,27)}$, $Y_{(6,28)} = Y_{(5,28)}$, $Y_{(6,29)} = Y_{(5,29)}$, $Y_{(6,30)} = Y_{(5,29)}$
Y_7	$Y_{(7,20)} = Y_{(6,20)}$, $Y_{(7,21)} = Y_{(6,21)}$, $Y_{(7,2)} = 0$, $Y_{(7,3)} = 0$, $Y_{(7,4)} = 0$, $Y_{(7,5)} = 0$, $Y_{(7,6)} = 0$
Y_8	$Y_{(8,31)} = 0$, $Y_{(8,27)} = 0$, $Y_{(8,28)} = 0$, $Y_{(8,2)} = 1$, $Y_{(8,29)} = 0$, $Y_{(8,30)} = 0$, $Y_{(8,3)} = 1$, $Y_{(8,4)} = 1$, $Y_{(8,5)} = 0$, $Y_{(8,6)} = 1$
Y_9	$Y_{(9,16)} = 1$, $Y_{(9,20)} = 0$, $Y_{(9,21)} = 0$, $Y_{(9,27)} = 1$, $Y_{(9,28)} = 1$, $Y_{(9,29)} = 1$, $Y_{(9,30)} = 1$
Y_{10}	$Y_{(10,31)} = Y_{(11,31)}$, $Y_{(10,10)} = 0$, $Y_{(10,11)} = 0$, $Y_{(10,20)} = 1$, $Y_{(10,21)} = 1$, $Y_{(10,2)} = 0$, $Y_{(10,3)} = 0$, $Y_{(10,4)} = 0$, $Y_{(10,5)} = 0$, $Y_{(10,6)} = 1$
Y_{11}	$Y_{(11,31)} = Y_{(10,31)}$, $Y_{(11,16)} = 0$, $Y_{(11,27)} = 0$, $Y_{(11,28)} = 0$, $Y_{(11,29)} = 0$, $Y_{(11,30)} = 0$
Y_{12}	$Y_{(12,10)} = 0$, $Y_{(12,11)} = 0$, $Y_{(12,20)} = 0$, $Y_{(12,21)} = 0$, $Y_{(12,2)} = 0$, $Y_{(12,3)} = 0$, $Y_{(12,4)} = 0$, $Y_{(12,5)} = 0$, $Y_{(12,6)} = 0$
Y_{13}	$Y_{(13,31)} = 0$, $Y_{(13,16)} = Y_{(14,16)}$, $Y_{(13,27)} = 0$, $Y_{(13,28)} = 0$, $Y_{(13,29)} = 0$, $Y_{(13,30)} = 1$
Y_{14}	$Y_{(14,31)} = 1$, $Y_{(14,10)} = Y_{(15,10)}$, $Y_{(14,11)} = Y_{(15,11)}$, $Y_{(14,16)} = Y_{(13,16)}$, $Y_{(14,20)} = 0$, $Y_{(14,20)} = 0$, $Y_{(14,21)} = 0$
Y_{15}	$Y_{(15,10)} = Y_{(14,10)}$, $Y_{(15,11)} = Y_{(14,11)}$
Y_{16}	$Y_{(16,31)} = 0$, $Y_{(16,16)} = 0$, $Y_{(16,20)} = 0$
Y_{17}	$Y_{(17,10)} = 0$, $Y_{(17,11)} = 0$, $Y_{(17,16)} = 1$, $Y_{(17,5)} = 0$
Y_{18}	$Y_{(18,31)} = 0$, $Y_{(18,0)} = 0$, $Y_{(18,10)} = 1$, $Y_{(18,11)} = 1$, $Y_{(18,1)} = 0$, $Y_{(18,20)} = Y_{(19,20)}$, $Y_{(18,2)} = 0$
Y_{19}	$Y_{(19,16)} = 0$, $Y_{(19,20)} = Y_{(18,20)}$, $Y_{(19,5)} = 0$
Y_{20}	$Y_{(20,31)} = 0$, $Y_{(20,0)} = 0$, $Y_{(20,10)} = 0$, $Y_{(20,11)} = 0$, $Y_{(20,1)} = 0$, $Y_{(20,2)} = 0$
Y_{21}	$Y_{(21,16)} = 0$, $Y_{(21,20)} = 0$, $Y_{(21,5)} = Y_{(22,5)}$
Y_{22}	$Y_{(22,31)} = Y_{(23,31)}$, $Y_{(22,0)} = Y_{(23,0)}$, $Y_{(22,9)} = 0$, $Y_{(22,10)} = 0$, $Y_{(22,11)} = 0$, $Y_{(22,1)} = Y_{(23,1)}$, $Y_{(22,20)} = 1$, $Y_{(22,2)} = Y_{(23,2)}$, $Y_{(22,5)} = Y_{(21,5)}$
Y_{23}	$Y_{(23,31)} = Y_{(22,31)}$, $Y_{(23,0)} = Y_{(22,0)}$, $Y_{(23,1)} = Y_{(22,1)}$, $Y_{(23,2)} = Y_{(22,2)}$
Y_{24}	$Y_{(24,9)} = 0$, $Y_{(24,20)} = 0$, $Y_{(24,5)} = 0$
Y_{25}	$Y_{(25,31)} = 0$, $Y_{(25,0)} = 0$, $Y_{(25,1)} = 0$, $Y_{(25,2)} = 0$, $Y_{(25,5)} = 1$
Y_{26}	$Y_{(26,31)} = 1$, $Y_{(26,0)} = 1$, $Y_{(26,9)} = 0$, $Y_{(26,1)} = 1$, $Y_{(26,20)} = 0$, $Y_{(26,2)} = 1$
Y_{27}	$Y_{(27,9)} = 0$, $Y_{(27,20)} = 0$, $Y_{(27,5)} = 0$
Y_{28}	$Y_{(28,31)} = 0$, $Y_{(28,0)} = 0$, $Y_{(28,1)} = 0$, $Y_{(28,20)} = 1$, $Y_{(28,2)} = 0$
Y_{29}	$Y_{(29,9)} = 0$, $Y_{(29,20)} = 1$, $Y_{(29,5)} = 0$
Y_{30}	$Y_{(30,31)} = 0$, $Y_{(30,0)} = 0$, $Y_{(30,9)} = 0$, $Y_{(30,1)} = 1$, $Y_{(30,20)} = 0$, $Y_{(30,2)} = 0$
Y_{31}	$Y_{(31,9)} = 1$, $Y_{(31,20)} = 0$
Y_{32}	$Y_{(32,9)}$!= $Y_{(33,9)}$
Y_{33}	$Y_{(33,9)}$!= $Y_{(32,9)}$, $Y_{(33,20)} = 1$
Y_{34}	$Y_{(34,20)} = 0$
Y_{35}	$Y_{(35,20)} = Y_{(36,20)}$
Y_{36}	$Y_{(36,20)} = Y_{(35,20)}$
Y_{37}	$Y_{(37,20)} = 0$
Y_{82}	$Y_{(82,9)} = 0$
Y_{83}	$Y_{(83,9)} = 1$
Y_{84}	$Y_{(84,9)} = 0$
Y_{86}	$Y_{(86,9)} = 1$
Y_{87}	$Y_{(87,9)} = 0$
Y_{88}	$Y_{(88,9)} = 0$

main methods to define the condition in each related step.

6 Conclusion

The experiment showed how to utilize the backward analysis to construct the differential condition for finding a collision in Haval-128 Pass 3. We will show in a consequent paper that our method can be applied to most hash functions. The key of this analysis is that the differential characteristic of BDO and BDI must not affect the 2nd and 3rd Passes. Moreover, from the experiment result, we found that the nonlinear function affects significantly to the compression function. The highly-nonlinear function does not mean that the compression function is also complex and secure. Our future study will be concentrated on the XOR and additional modulus that are clearly the weak points of the most of hash functions.

Tables given in Appendix A shows a set of sufficient conditions for the success of searching collisions and differential characteristics of Haval when $i = 11$ and $i = 10$. The method used in [4] for Haval is:

$$M' = M + \triangle C, \triangle C = (2^{i-1}, 2^{i-12}, 2^{i-8})$$

at word position $0, 11, 18$ and $i = 0, 1, 2, ..., 31$.

References

[1] P. Hawkes, M. Paddon, and G. G. Rose, *Musings on the Wang et al. MD5 Collision*, IACR Eprint Archive, Report 2004/264, Oct. 2004.

[2] V. Klima, *Finding MD5 Collisions a Toy For a Notebook*, IACR Eprint Archive, Report 2005/075, Mar. 2005.

[3] V. Klima, *Finding MD5 Collisions on a Notebook PC Using Multi-message Modifications*, IACR Eprint Archive, Report 2005/102, Apr. 2005.

[4] X. Y. Wang, D. G. Feng, X. J. Lai, and H. B. Yu, *Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD*, Rump session of CRYPTO'04 and IACR Eprint Archive, Report 2004/199, Aug. 2004.

[5] X. Y. Wang, D. G. Feng, X. J. Lai, and H. B. Yu, "Cryptanalysis of the hash functions MD4 and RIPEMD," *Advances in Cryptology-Eurocrypt'05*, LNCS 3494, pp. 1-18, R. Cramer, Ed., Springer-Verlag, May 2005.

[6] X. Y. Wang, and H. B. Yu, "How to break MD5 and other hash functions," *Advances in Cryptology-Eurocrypt'05*, LNCS 3494, pp. 19-35, R. Cramer, Ed., Springer-Verlag, May 2005.

[7] X. Y. Wang, Y. L. Yin, and H. B. Yu, "Efficient collision search attacks on SHA-0," *Advances in Cryptology-Crypto'05*, LNCS 3621, pp. 1-16, V. Shoup, Ed., Springer-Verlag, Aug. 2005.

[8] X. Y. Wang, Y. L. Yin, and H. B. Yu, "Finding collisions in the full SHA-1," *Advances in Cryptology-Crypto'05*, LNCS 3621, pp.17-36, V. Shoup, Ed., Springer-Verlag, Aug. 2005.

[9] Y. L. Zheng, J. Pieprzyk, and J. Seberry, "HAVAL A one-way hashing algorithm with variable length of output," *Advances in Cryptology-Auscrypt'92*, LNCS 718, pp. 83-104, J. Seberry and Y. Zheng, Eds., Springer-Verlag, 1993.

Appendix A

The differential characteristics and collision tables obtained in our experiment are listed in Tables 8 – 11.

Pairat Thorncharoensri is doing a Master degree in computer science by research at Wollongong University. He received a master of computer science by coursework and a master of internet technology by coursework from Wollongong University in 2004 and 2003 respectively. He received a bachelor of electrical engineering from King Mongkut's Institute of Technology North Bangkok, Thailand. His current research interests include network security, computer security, and cryptography.

Tianbing Xia received his PhD from the University of Wollongong in 2001. He currently is a senior lecturer in the School of Information Technology and Computer Science, University of Wollongong. His research interests include Boolean functions, Hadamard matrix and orthogonal designs, and computer security. He is a member of ACM.

Yi Mu received his PhD from the Australian National University in 1994. He was a lecturer in the School of Computing and IT at the University of Western Sydney and a senior lecturer in the Department of Computing at Macquarie University. He currently is an associate professor in the Information Technology and Computer Science, University of Wollongong. His current research interests include network security, computer security, and cryptography. Yi Mu has published more than 100 research papers in International conferences and journals. He has served as a program committee member in a number of international conferences and is a member of the Editorial Boards of several international journals. He is a senior member of the IEEE, and a member of the IACR.