How To Find Weak Input Differences For MD5 Collision Attacks

[1,2]Tao Xie, [1]Dengguo Feng

[1]State Key Lab of Information Security, Chinese Academy of Sciences, Beijing, China
[2]The Center for Soft-Computing and Cryptology, NUDT, Changsha,China
(hamishxie@vip.sina.com)

Abstract. Since the first feasible collision differential was given for MD5 in 2004 by Wang et al, a lot of work has been concentrated on how to improve it, but the researches on how to select weak input differences for MD5 collision attack are only sporadically scattered in literature. This paper focuses on a reasonable selection of weak input differences for MD5 collision attack, tries to answer some questions such as, what techniques can be applied satisfying bit conditions? which step in the second round can be the latest to apply a search on free bits without violating previously satisfied conditions? what is the optimal characterization of feasible collision differential propagation for MD5, by which we can find more weak input differences? is there any collision differentials that exceed Wang et al's by some practical criteria? In this paper, a divide-and-conquer strategy is introduced with an optimal scheme of grouping the 64 steps of operation into five stages of independent condition fulfillment, and a feasible collision differential propagation is optimally characterized as a guide to select those 1–3-bit weak input differences, with their computational costs estimated. As a result, hundreds of thousands of weak input differences have been found, quite a number of which are superior to Wang et al's, for example, a 2-bit collision differential is able to find a collision within 2^{10} MD5 compressions, a 1-MSB differential collision attack on MD5 is developed with a time complexity of $2^{20.96}$ MD5 compressions, and a practical 1-block collision attack on MD5 is found to be possible. This paper also provides a rich resource of colliding messages with different weak input differences, therefore much greatly increase the probability of finding a second MD5 pre-image for an arbitrarily given message.

Key words: MD5, Differential Collision Attack, Divide-and-Conquer, Weak Input Differences, Time Complexity Estimation.

1 Introduction

Since Wang et al gave a first practical differential collision attack especially on MD5 in 2004 [1] and 2005 [2, 3], a research surge on hash cryptanalysis has appeared. The cryptanalysis on hash functions focuses mainly on three aspects, firstly a fundamental work is to seek a complete understanding of how this approach works, secondly a reasonable direction is to improve the original collision attack, and thirdly but not finally comes up a question whether it can be generalized, for examples, to attack MD5's second pre-image and pre-image, or even HMAC and NMAC etc.

Basically, Wang et al's differential collision attack [2, 3] is a hybrid differential cryptanalysis which takes advantages of both modular difference and XOR difference. In general, three steps are involved in a differential collision attack. The first step is to find a feasible collision differential, which is called weak input difference in this paper, the second step is to construct a feasible differential characteristic or path that leads to a collision, the final step is to design a specific algorithm to find colliding message pairs efficiently. In the final step, these colliding pairs are searched to satisfy all bit conditions that guarantee a differential collision along the differential characteristic. In the past four years, almost all researches focus on the last two steps, and a great number of papers have appeared in variety of conferences or workshops. Since this is not a survey paper, it is inappropriate to review all the work on the second and third steps. However, several notable work can not be ignored. Firstly, a breakthrough in message modification called tunneling technique is made by V. Klima [4], it can be used to provide more freedom to fulfill conditions in the second round, and this results in an improved algorithm capable of generating a collision within one minute on a desktop PC, exponentially accelerating collision search. Klima's tunneling technique is used by Marcs Steven [5] to further reduce the computational cost to $2^{24.8}$ MD5 compressions, such that a collision can be found in several seconds on a desktop PC. In Eurocrypt2007, secondly, a practical and nearly meaningful pair of colliding X.509 certificates for two different distinguished name was found with the same MD5 digest by Marcs Steven [6], this work make the method of producing MD5 collisions a really shattering attack on practical protocol. Especially, Marcs Steven et al [5, 6] and C. De Cannire et al [7–9] contributed respectively to half-automatically designing and optimizing a MD5 and SHA-1 differential characteristic, which could save a large part of human work and have resulted in better collision attacks on MD5 and SHA-1 [9]. Thirdly, by using an *if-then-else* programming structure, two different Postscript files were created with the same MD5 digest but to result in different texts when screening [10], and this attack was extended to other file formats in [11]. Fourthly but not finally, theoretical attacks on the pre-images and second pre-images of MDx hash functions have been made in [12–16], and similar attacks on HMAC and NMAC with several hash functions have also been proposed in [17–19]. However, all the work above especially on MD5 are based on Wang et al's original collision and approach, no new collision differentials have been found to be better than Wang et al's attack [5, 20, 21]. This makes it necessary to get to the bottom of how the weak input differences are selected, how collision differentials are evaluated, and what input difference can be the best collision differential in terms of practically considered criteria. A good solution to these problems will answer questions, such as "Is Wang et al's collision differential the best selection for MD5?", "How fast do we actually make collision attack on MD5?","Can a second pre-image be practically made given an arbitrary message?","Can MD5 collision be applied for attacks on practical security protocol?" and so on.

This paper mainly focuses on an optimal characterization of the feasible collision differential propagation for MD5, tries to find as many as possible weak input differences that conform to the optimal characterization, and to make estimations on the computational costs of obtaining a collision for each weak input difference. In this way, better collision differentials than Wang et al's can be found for MD5. Deep analysis on the correlation between state variables by message words in the first two rounds, shows that a divide-and-conquer strategy can be applied to optimally separate all 64 steps of operation into five groups consecutively, with the final group of steps starting from step a_7. Using the direct and indirect message modifications as well as the tunneling technique, bit conditions in each group can be fulfilled without violating previously satisfied conditions. As a result, the final group of steps becomes the critical one to fundamentally define the computational complexity of generating a collision. This imposes a limit on the number of bit conditions in the final group of steps, such that the differential collision attack is more efficient than the birthday paradox attack. Together with the differential propagation properties implicit in each round, this limit further shapes the differential characteristic for the last two rounds, prescribes an optimal characterization of feasible collision differential propagation, and finally leaves clues on finding weak input differences.

This paper is organized as follows: In Section 2, a divide-and-conquer strategy is described with four condition fulfilling techniques, including the direct message modification, the indirect message modification, the tunneling technique and the grouping scheme, the grouping scheme is an application of the divide-and-conquer strategy in MD5 collision attack, resulting in a general procedure for collision searching. In Section 3, an optimal characterization of feasible collision differential propagation is carried out by the birthday collision attack's limiting on the number of bit conditions in the final group of steps, together with the different difference-incurred bit conditions and the differential propagation properties implicit in each round. In Section 4, all 1–3-bit weak input differences are selected to conform with the optimal differential propagation characterization, including 41 1-bit weak input differences, all possible combinations of two 1-bit weak input differences and four 3-bit weak input differences. In Section 5, using four weak input differences as examples, a general method is proposed to estimate the computational cost for each weak input difference, and some criteria are proposed and described to evaluate weak input differences, then using the optimal grouping scheme, an example of weak input differences is chosen to develop a new differential collision attack on MD5, being able to generate a collision within a second on a common desktop PC. In Section 6, a summary is made with some conclusions drawn.

2 A Divide-and-Conquer Strategy For Collision Searching

Dedicated hash functions like MDx family are actually discrete functions which iterate many steps of nonlinear operations, thus considered usually as complex nonlinear functions. Traditionally, a divide-and-conquer strategy is obviously a primary way to solve complex problems that will incur a currently-unachievable computational effort if no reduction is made.

– **The Divide-and-Conquer Axiom:** *If a problem P with computational complexity $C_{original}$ can be divided into k sub-problems $P_i (1 \leq i \leq k)$, and each sub-problem P_i can be solved independently with computational complexity C_i but without destroying other sub-problems' solutions, then the computational complexity to solve the original problem reduces to $C_{divide-and-conquer} = \sum_{n=1}^{k} C_i$ instead of $C_{original} = \prod_{i=1}^{k} C_i$.*

By the basic fact, that a multiplicative computational complexity of the original problem can be transformed into an additive accumulation of all sub-problem's computational complexities, if the original problem can be massively divided and independently conquered, we can give a theorem similar to the divide-and-conquer axiom. Since the conditions in the first round and some steps like a_5 and b_5 in the second round can be satisfied by message modifications, they are usually ignored in the calculation of computational complexity. In the grouping theory, only probabilistically satisfiable conditions are considered.

– **The Grouping Theorem For MDx Differential Collision Attack:** *Assume p bit conditions in a differential characteristic be only probabilistically satisfiable by a random or brute-force search, and the search procedure can be divided into groups of condition satisfying, namely G_1, G_2, $\ldots$ and G_k, and $\sum_{i=1}^{k} |G_i| = p$. Let $G_{max} = max\{G_1, G_2, \ldots, G_k\}$ be the group with the most conditions to be only probabilistically satisfiable. If all groups of bit conditions are sequentially fulfilled, i.e. G_i is fulfilled before G_{i+1}, and the bit conditions in G_1 to G_i are not violated by satisfying the bit conditions of G_{i+1} and so on. Then, the computational complexity of satisfying the p bit conditions, will be reduced to an additive accumulation of the computational complexities for all groups instead, and the group G_{max} will be characteristic of the total computational complexity, provided that there exist enough freedom (free bits) to be searched for each group.*

Usually, the largest group of only probabilistically satisfiable conditions G_{max} is the final one which includes all bit conditions starting from a step in the second round. Hence, a feasible collision differential characteristic should be constructed such that the number of bit conditions in the final group is currently within the computational feasibility. Therefore, a fundamental problem emerges from behind the divide-and-conquer theorem, that how many groups can we divide a collision differential characteristic independently, what is the optimal separation of MD5 steps, and which step in the second round can be the latest as the beginning step in the final group? A good answer to these questions will lay a theoretical foundation for an optimal characterization of feasible collision differential propagation.

To achieve a feasible differential collision attack on MD5, three steps as described in Section 1 are sequentially involved, and we wish this sequence had not misguided researchers in this field. Actually, however, when you begin to find your feasible collision differentials, a reverse contemplation on differential collision attack will benefit more than sequentially following Wang et al's approach. Since a differential collision attack is feasible only if it is more efficient than the birthday attack, and the possible number of bit conditions and their arrangement along the differential characteristic are completely determined by condition satisfying techniques, we have to make clear that, how many condition satisfying techniques exist, how and to what extent these techniques help satisfy bit conditions.

For the remainder of this article we follow the notation of [22], and restate it whenever needed. For MD5 function, the 64 steps of state variables are sequentially denoted as a_i, d_i, c_i and b_i, where $0 \le i \le 16$ with $i = 0$ defining the four initial state variables a_0, d_0, c_0 and b_0. Particularly, the state variables a_i, d_i, c_i and b_i are also used to denote the corresponding steps of updating operation. Note that the practical sequence order (a_i, d_i, c_i and b_i) of updating state variables is different from the natural order (a_i, b_i, c_i and d_i). Bit index in a word starts from 0 (LSB) to 31 (MSB) unless being specially restated. The four auxiliary bitwise functions are sequentially defined as $F(x, y, z) = (x \wedge y) \vee (\bar{x} \wedge z)$, $G(x, y, z) = (x \wedge z) \vee (y \wedge \bar{z})$, $H(x, y, z) = x \oplus y \oplus z$ and $I(x, y, z) = y \oplus (x \vee \bar{z})$.

For the sake of comprehension and integrity, in Appendix A, a concise introduction is given for MD5 function.

2.1 Direct Message Modification

The step operation equation can be used to calculate a message word such that the corresponding state variable is satisfied with its bit conditions, this is called the direct message modification. Direct message modification is usually used in the first round and occasionally applied in the second round for bit conditions' satisfaction. For example, a state variable a_{i+1} can be randomly assigned but with all its bit conditions prescribed, which can be maintained by calculating a corresponding message word with a transformed step equation. Direct message modification was early used by Boer and Bosselaers [23] in their pseudo collision attack on MD5, and massively applied in Wang et al's [2, 3] differential attack on MD5 for direcltly satisfying all bit conditions in the first round. Direct message modification has become a fundamental technique in differnetial attack on MDx hash functions.

2.2 Indirect Message Modification

The message words used to satisfy the next state variable in the second round will unavoidably change the corresponding state variables or even break some previously satisfied conditions in the first round, this feedback of change from the second round to the first round by a common message word frustrates the direct message modification in the second round. An improvement was made firstly by F. Chabaud and A. Jaux in their differential collision attack on Sha-0, in which the next multiple messages were used to absorb the change of the current state variable due to the direct message modification in the second round, such that the feedback of change from the second round will be blocked from propagating into the second round [24]. This technique of one message word's change having to be complemented by multiple consecutive message words' modifications, is called an indirect message modification, which is virtually a multiple message modification. Indirect message modification can be forward message modification and backward message modification. For an example of indirect forward message modification, if the message word m_1 is recalculated to satisfy the bit conditions in state variable a_5, the corresponding state variable d_1 in the first round must be changed. In order to restrict the change only in d_1, direct modification on next four message words must be implemented to keep the next four state variables unchanged, respectively c_1, b_1, a_2 and d_2. For an example of indirect backward message modification, if the message word m_{11} is recalculated to satisfy the bit conditions in c_5, a corresponding modification on b_2 must be made so that b_3 will keep unchanged, consequently the next three consecutive message modifications must be implemented to maintain the state variables a_3, d_3 and c_3.

This multiple message modification has been improved and consummated further by V. Klima [4], Marc Stevens [5] and Tao Xie [21, 22], it is currently the basic modification technique in the differential collision attack on MDx hash functions. As for MD5, the number of state variables that can be indirectly satisfied by multi-message modification, is determined by the permutation of message words in the second round, the number of free state variables and their arrangement in the first round. However, the state variables a_5 and b_5 can usually be satisfied by this multi-message modification, provided that the first two state variables a_1 and d_1 are free. Generally speaking, a multi-message modification can be applied to satisfy a state variable in the second round, only when the corresponding state variable in the first round is free, or both state variables in the first two rounds can be satisfied with the same message word and inconsistent bit conditions do not occur.

2.3 Tunneling Technique

Tunneling technique is firstly proposed by V. Klima [4] to improve Wang et al's differential collision attack efficiency, it takes advantage of the selective function in the first round, to add in more freedom in collision searching than the multi-message modification, thus it is a more advanced technique for message modification and bit condition fulfillment.

Assume four consecutive state variable bit $a_{i,j}$, $d_{i,j}$, $c_{i,j}$ and $b_{i,j}$ in the first round of MD5. If we set $c_{i,j} = 0$, $b_{i,j} = 1$ and make changes on $d_{i,j}$, then the selective function $F(c_{i,j}, d_{i,j}, a_{i,j}) = (c_{i,j} \wedge d_{i,j}) \vee (\bar{c}_{i,j} \wedge a_{i,j}) = a_{i,j}$ and $F(b_{i,j}, c_{i,j}, d_{i,j}) = (b_{i,j} \wedge c_{i,j}) \vee (\bar{b}_{i,j} \wedge d_{i,j}) = 0$, consequently the change in $d_{i,j}$ has no effect on the next two state variables b_i and a_{i+1}, where $1 \leq i \leq 4$, $0 \leq j \leq 31$. This property of the selective function is called change absorption, and a free bit $d_{i,j}$ with the next two bits' specification $c_{i,j} = 0$ and $b_{i,j} = 1$ constitute a tunneling bit. More precisely, the number of tunneling bits in three consecutive steps in the first round is defined as the strength of the tunnel.

Theoretically, this tunneling technique can be applied without violating previously satisfied conditions in all the steps from a_5 to d_7 except c_6, in which m_{15} is involved, but it is more suitable to be applied in a_6, d_6, b_6, a_7 and d_7, since the state variables a_5, d_5, c_5 and b_5 can be satisfied both by the multi-message modification and the tunneling technique, while the state variables a_6, d_6, b_6, a_7 and d_7 can only be satisfied by the tunneling technique. Unfortunately, applying the tunneling technique in any step after step d_7 will unavoidably break previously satisfied conditions. For a feasible collision differential characteristic, only b_6, a_7 and d_7 are practically suitable for tunneling technique, therefore d_7 become the latest step to apply the tunneling technique.

2.4 Grouping Scheme—An Application Of The Divide-and-Conquer Strategy

The above mentioned three message modifications constitute different levels of bit condition fulfilling techniques, which define the basic grouping scheme together with the permutation of message words in the second round. Theoretically, starting from step a_5 to d_7 , each step or several consecutive steps can be grouped themselves, if there exist enough freedom (free bits) in the group. Practically, however, a scheme of properly grouping steps will benefit while too finely or coarsely grouping will ultimately abate attack efficiency, since freedom is not always enough to satisfy bit conditions. A criterion on optimally grouping the MD5 steps is, to let the message word that is involved in the beginning step of each group be as near as possible the front of the first round or just be the steps with enough tunneling bits, so that there are enough freedom to fulfill all bit conditions in the group, since free steps are always near the front. In general, we give an optimal grouping scheme as follows with respect to some considerations, such as more comprehensive and more instructive for the construction of feasible collision differential characteristics.

- **Group-1:** The steps from a_1 to b_4, that is the first round, constitute the first group. These state variables can be satisfied by the direct message modification.

- **Group-2:** a_5, d_5 and c_5 constitute the second group. The bit conditions in d_5 and c_5 can be satisfied by searching the free bits in a_5, while a_5 can be satisfied by the multi-message modification. If too many bit conditions exist in this group, more finely grouping of steps will benefit.
  - ***Group-2.1:*** a_5 can individually constitute a group or a group with d_5, a_5 or both a_5 and d_5 can be satisfied by the multi-message modification plus a random search on the free bits in a_5.
  - ***Group-2.2:*** d_5 can individually constitute a group or a group with c_5, d_5 or both d_5 and c_5 can be satisfied by the multi-message modification plus a random search on the free bits in d_5.
  - ***Group-2.3:*** c_5 constitutes a group, and c_5 can be satisfied by the multi-message modification.

- **Group-3:** b_5, a_6, d_6 and c_6 constitute the third group. The bit conditions in a_6, d_6 and c_6 can be satisfied by searching the free bits in b_5, while b_5 is satisfied by the multi-message modification. If too many bit conditions exist in this group, more finely grouping of steps will benefit.

- *Group-3.1:* b_5 and a_6 constitute a group which can be satisfied by multi-message modification.
- *Group-3.2:* a_6 and d_6 constitute a group which can be satisfied by multi-message modification.
- *Group-3.3:* d_6 and c_6 constitute a group which can be satisfied by multi-message modification.

– **Group-4:** All steps from b_6 or a_7 to b_{16} can be put in the fourth group. All bit conditions in these steps can be satisfied by the tunneling searching through three consecutive steps b_1, a_2, d_2 and/or a_3, d_3, c_3. If too many bit conditions exist in b_6, then it will benefit more to make b_6 an independent ***Group 4.1*** and all steps after b_6 to be ***Group 4.2***.

- *Group-4.1:* The step b_6 individually constitutes a group, and state variable b_6 can be satisfied by the tunneling searching through three consecutive steps b_1, a_2, d_2.
- *Group-4.2:* The steps from a_7 to b_{16} together constitute the final group, all bit conditions in this group can be satisfied by the tunneling searching through three consecutive steps a_3, d_3, c_3.

– **Group-5:** The steps from d_7 to b_{16} can constitute a final group, and bit conditions in these steps can be satisfied by the tunneling searching through d_4, c_4 and b_4, which is cascaded by another tunneling searching through c_1, d_1 and a_2 to absorb the modification in message word m_6.

By the optimal grouping scheme described above, theoretically d_7 will be the latest step to randomly fulfill the bit conditions behind this step, but practically, the two cascaded tunneling can not be employed as the final group searching, since the number of tunneling bits will be much less than that of bit conditions in the final group, hence the improved efficiency is limited. Therefore, a collision differential characteristic should be constructed so that the section from a_7 to b_{16} will incur bit conditions as small as possible.

Figure 1 is a flowchart to explain how the bit conditions in all steps are satisfied by the optimal grouping scheme. A general procedure for MD5 collision searching will benefit researchers more than a specific algorithm, if it is based on the grouping scheme. Based on Figure 1, a general procedure for MD5 differential collision searching is given as follows:

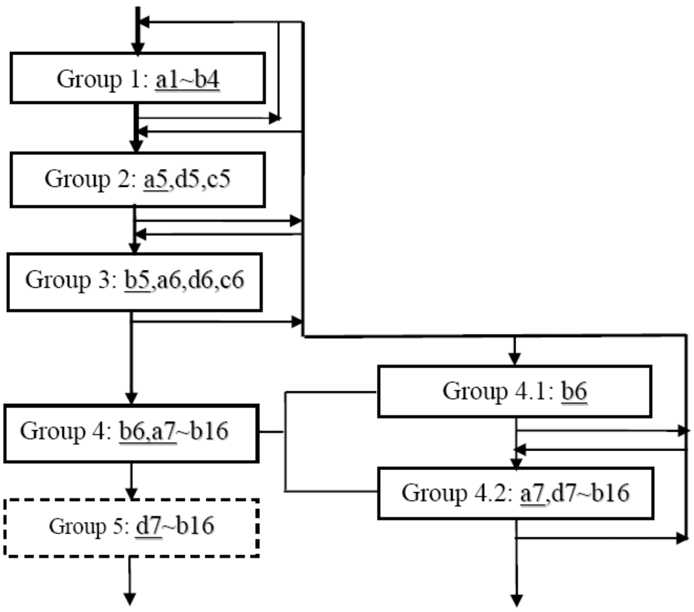

Fig. 1. An Optimal Grouping Scheme (MD5)

– **Group-1's Satisfaction:** To satisfy all steps in the first group by direct message modification.

- **Group-2's Satisfaction:** To satisfy all bit conditions in the second group by the indirect message modification in a random or brute-force search, and keep all bit conditions in the first group satisfied; if not all bit conditions in the second group are satisfied when the search is over, return to **Group-1's Satisfaction**.

- **Group-3's Satisfaction:** To satisfy all bit conditions in the third group by the indirect message modification in a random or brute-force search, and keep all bit conditions in the first two groups satisfied; if not all bit conditions in the third group are satisfied when the search is over, return to **Group-2's Satisfaction**.

- **Group-4.1's Satisfaction:** To satisfy all bit conditions in the fourth group by the tunneling searching technique in a random or brute-force search, and keep all bit conditions in the first three groups satisfied; if not all bit conditions in the fourth group are satisfied when the search is over, return to **Group-3's Satisfaction**.

- **Group-4.2's Satisfaction:** To satisfy all bit conditions in the fifth group by the tunneling searching technique in a random or brute-force search through early stop scheme, and keep all bit conditions in the first four groups satisfied; if not all bit conditions in the fifth group are satisfied when the search is over, return to **Group-4.1's Satisfaction**.

The early stop scheme means, when a bit condition is found to be not satisfied in a step of operation, all the next steps of operation are ignored and the algorithm returns to the beginning of this group's satisfaction and starts a new.

3 Characterization of Weak Differential Propagation

3.1 Bit Conditions Incurred By Bit Difference

For the sake of clarity, we only consider a situation where there is only one bit difference occurring at the same bit in three consecutive steps, as this can be generalized to other situation where there are more than one (two or three) bit differences occurring at the same bit in three consecutive steps if you take other bit differences as fixed bit conditions, and this will reduce the number of bit conditions to be prescribed. Extra conditions due to carries and bit rotations are ignored here, since they do not have any effect on the following analysis.

- **Round 1-2: A bit difference in the first two rounds will incur 4 or 5 bit conditions.** For example, if $b_{7,i} = +1$, that is, there exists a positive flip at the i^{th} bit of b_7, then in three consecutive steps of operation, b_7 will be used respectively as x, y and z in the second auxiliary function $G(x, y, z) = (x \wedge z) \vee (y \wedge \bar{z})$, each leads to a specific bit condition or a prescribed correlation for other two bits. To be more precisely, for the first situation where step operation a_8 is implemented, whether the bitwise function $G(+1, c_{7,i}, d_{7,i}) = (+1 \wedge d_{7,i}) \vee (c_{7,i} \wedge \bar{d}_{7,i})$ produces a positive flip at the i^{th} bit or not, it will depend on the specific assignment on $d_{7,i}$, $d_{7,i} = 1$ produces a positive bit flip, otherwise $c_{7,i}$. For the second situation where step operation d_8 is implemented, whether the bitwise function $G(a_{8,i}, +1, c_{7,i}) = (a_{8,i} \wedge c_{7,i}) \vee (+1 \wedge \bar{c}_{7,i})$ produces a positive flip at the i^{th} bit or not, it will depend on the specific assignment on $c_{7,i}$, $c_{7,i} = 0$ produces a positive bit flip, otherwise $a_{8,i}$. For the third situation where step operation c_8 is implemented, whether the bitwise function $G(d_{8,i}, a_{8,i}, +1) = (d_{8,i} \wedge +1) \vee (a_{8,i} \wedge -1)$ produces a flip at the i^{th} bit or not, it will depend on the specifically prescribed correlation between $a_{8,i}$ and $d_{8,i}$, $d_{8,i} \neq a_{8,i}$ produces a bit flip, otherwise no bit flip. If a positive or negative flip is required, then a specific bit assignment for both $a_{8,i}$ and $d_{8,i}$ is necessary, which will incur two bit conditions.
  As a result, the specific assignments on four bits plus the bit difference itself constitute five bit conditions for a non-MSB difference or four bit conditions for a MSB difference in the second round. Similarly, three situations will happen when a bit difference appears in the first round.

- **Round 3: In the third round, a non-MSB difference will definitely incur 4 bit conditions, while a MSB difference catches no bit conditions.** For example, if $b_{9,i} = \pm 1$ and $i \neq 31$, that is, there exists a flip at the i^{th} bit of b_9, then in three consecutive steps of operation, b_9 will be used respectively as x, y and z in the third auxiliary function $H(x, y, z) = x \oplus y \oplus z$, each leads to a specifically prescribed correlation for the other two bits. No matter what situation (as x, y or z) b_9 is in, the other two bits have only two correlative options that have direct effect on results, one is two equal bits will produce the same flip as $b_{9,i}$, the other one is two different bits will flip the original flip at $b_{9,i}$. Thus, three specifically prescribed correlations on different two bits plus the bit difference itself definitely constitute 4 bit conditions for a non-MSB difference in the third round. Since $2^{31} = -2^{31}\ mod(2^{32})$, no bit conditions will be incurred by a MSB difference in the third round.

- **Round 4: In the fourth round, a non-MSB difference will definitely incur 5 bit conditions, while a MSB difference catches only three bit conditions.** For example, if $b_{13,i} = +1$, that is, there exists a positive flip at the i^{th} bit of b_{13}, then in three consecutive steps of operation, b_{13} will be used respectively as x, y and z in the final auxiliary function $I(x, y, z) = y \oplus (x \vee \bar{z})$, each leads to a specific bit condition or a prescribed correlation for the other two bits. To be more precisely, for the first situation where step operation a_{14} is implemented, whether the bitwise function $I(+1, c_{13,i}, d_{13,i}) = c_{13,i} \oplus (+1 \vee \bar{d}_{13,i})$ produces a flip at the i^{th} bit or not, it will depend on the specific assignment on $d_{13,i}$, $d_{13,i} = 1$ produces a bit flip, otherwise the original flip at $b_{13,i}$ is absorbed. More specifically, when $d_{13,i} = 1$, whether the bitwise function $I(+1, c_{13,i}, 1) = c_{13,i} \oplus (+1 \vee 0)$ produces a positive or negative flip at the i^{th} bit, it will depend on the specific assignment on $c_{13,i}$, $c_{13,i} = 0$ produces a positive bit flip and vice versa. For the second situation where step operation d_{14} is implemented, whether the bitwise function $I(a_{14,i}, +1, c_{13,i}) = +1 \oplus (a_{14,i} \vee \bar{c}_{13,i})$ produces a positive or a negative flip at the i^{th} bit, it will depend on the specific assignment on $a_{14,i}$ and $c_{13,i}$, only $a_{14,i} = 0$ and $c_{13,i} = 1$ produce a positive bit flip, otherwise a negative bit flip.
  For the third situation where step operation c_{14} is implemented, whether the bitwise auxiliary function $I(d_{14,i}, a_{14,i}, +1) = a_{14,i} \oplus (d_{14,i} \vee -1)$ produces a flip at the i^{th} bit or not, it will depend on the specific assignment on d_{14}, $d_{14} = 0$ produces a bit flip, otherwise no bit flip. More specifically, when $d_{14} = 0$, whether the bitwise function $I(0, a_{14,i}, +1) = a_{14,i} \oplus (0 \vee -1)$ produces a positive or negative flip at the i^{th} bit, it will depend on the specific assignment on $a_{14,i}$, $a_{14,i} = 0$ produces a negative bit flip and vice versa.
  Similarly, since $2^{31} = -2^{31}\ mod(2^{32})$, no signed MSB flip needs to be considered, only three specific bit conditions are incurred by a MSB difference in the fourth round. As a result, specific assignments on four bits plus the bit difference itself constitute the five bit conditions for a non-MSB bit difference in the final round.

3.2 Differential Propagation Properties

- **Round 1-2: Differential propagation in the first two rounds can be divergent or convergent.** The selective function $F(x, y, z) = (x \wedge y) \vee (\bar{x} \wedge z)$ and $G(x, y, z) = (x \wedge z) \vee (y \wedge \bar{z})$ can be used to eliminate or reserve unexpected differences by prescribing the corresponding selective bits. In this way, differences can be propagated as expected, sometimes to expand but most of the time to converge. In the second round, for example, the differential propagation should be converged as early as possible to incur as few bit conditions as possible. As a result, usually a local collision or near collision can be obtained and the number of bit conditions incurred in the first two rounds will be minimized.

- **Round 3: Differential propagation in the third round is diffusive except along MSBs.** A non-MSB difference will be drastically diffused in the third round by the XOR function $H(x, y, z) = x \oplus y \oplus z$, while consecutive MSB differences can propagate along the way if not interrupted by input differences. Since consecutive MSB differences catch no bit conditions, this property of step operation in the third round requires within the second round a local collision such that a MSB-differential propagation can be built by input differences in the third round, or a near collision such that it can build a MSB-differential propagation with input differences from the third round. However, a near collision from the second round can also be eliminated by input differences from the third round to build a non-differential propagation. As a result, the number of bit conditions will be minimized. Since odd number (1 or 3) of MSB differences will produce a MSB difference again by the XOR function, and a non-MSB difference will occur in the next step if a MSB difference is produced by the addition modulo 2^{32} before left rotation in the step operation, four consecutive steps of MSB difference or no difference need to be generated before a MSB-(or non-)differential propagation is built. For example, if four consecutive steps of MSB difference are to be produced independently by the input differences in the third round, at least three 1-bit input differences need to be arranged in a proper way. To be more precisely, when a MSB-difference is produced by the first 1-bit input difference in a step, this MSB-difference must be cancelled out in the addition modulo 2^{32} before the left rotation in the next step operation with a second MSB input difference, a MSB difference is then obtained. Thanks to the co-elimination of both MSB differences, these two MSB differences can produce a new MSB difference in the third step without any input difference. However, in the fourth step, a third MSB input difference is needed again to cancel out the MSB difference in the addition modulo 2^{32} before the left rotation, so that a new MSB difference can be obtained. As a result, four MSB differences are consecutively produced and a MSB differential propagation is built without more input differences. In the same way, four consecutive MSB differences can be dissolved by three 1-bit input differences similarly arranged as above in the third round, and a non-differential propagation will be built instead. Nevertheless, 1-bit or 2-bit input difference can build a MSB(or non-)-differential propagation in the third round with some differences derived from the second round.

– **Round 4:Differential propagation in the final round is absorptive and can be within 2 bits.** Thanks to the ONX function $I(x, y, z) = y \oplus (x \vee \bar{z})$, a non-MSB difference can propagate within two consecutive bits, while a MSB difference can propagate along the way to the end, and this can be verified by checking up with the final steps in the differential characteristic of the first block given in [21]. Since five bit conditions will be incurred for a bit difference in the final round, it is better to keep the differential propagate along MSBs, and reduce the non-MSB differential propagation steps as much as possible. In this way, the number of bit conditions incurred in the final round will be minimized. Due to the ONX function, MSB input differences can be absorbed in a MSB-differential propagation, a non-MSB (turning into a MSB by the left rotation) input difference will produce a MSB-differential propagation independently in the final round. If a MSB-differential propagation is to be dissolved, a non-MSB (turning into a MSB by the left rotation) input difference followed by 3-step later a MSB input difference is needed, and this property predicts that a 1-block collision attack on MD5 is possible.

3.3 An Optimal Characterization of Weak Differential Propagation

Since 2-block collision attack on MD5 removes the restriction of no difference by a single block attack in the last four steps, it is easier to construct a 2-block collision attack, where it uses the first block to produce a near collision which can be further canceled out by the second block to generate a real collision. Given an input difference, the objective to design a feasible collision differential characteristic is to converge the differential propagation so that the number of bit conditions especially incurred in the final group of steps will be within the current computational feasibility (compared to the birthday paradox attack). Therefore, an optimal characterization of weak differential propagation will benefit the selection of weak input differences. Based on the differential propagation properties in each round, a weak input difference should be selected such that in the first block there generate four differential propagation sections, denoted respectively as I, II, III and IV in Figure 2. Section I is a non-differential area without input difference, section II is a near or local collision area across the first two rounds, section III is a non-differential area plus MSB-only differential chain starting from the first input difference of the third round if section II is an inner collision, or a MSB-only differential chain or non-differential chain directly derived from section II if section II is an inner near collision. Section IV is a near collision area consisting of only a few of the last consecutive steps, better no more than 4 steps, which are then modulo 2^{32} added with the initial value to produce the chain input

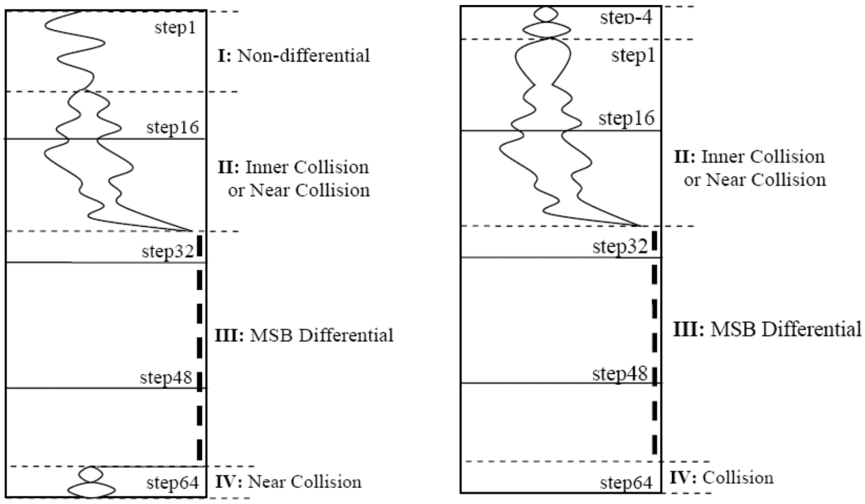

Fig. 2. The Optimal Characterization of Weak Differential Propagation

differences of the second block. Due to the chain input differences, section I does not exist in the second block, the chain input differences propagate all the way to the beginning of section II and constitute a lengthened section II starting from step1, and the section III and IV in the second block correspond respectively to the section III and IV in the first block, except that the differences in the last four consecutive steps are eliminated, i.e. turning a near collision into a full collision. This is illustrated in Figure 2 by the optimal characterization of the weak differential propagation for MD5 collision attack.

4 Selection of Weak Input Differences

Selection of weak input differences as feasible collision differentials is widely regarded as a trial-and-error work that depends on one's intuition, experiences and good luck as well, since Wang et al gave the first 3-bit collision differential. Based on the optimal characterization of weak differential propagation in Section 3.3, 1–3-bit input differences are analyzed in this section, and these input differences that will result in a weak differential characteristic up to the optimal characterization, are listed as follows, though more than 3-bit input differences are not considered in this paper.

4.1 1-Bit Weak Input Differences

As a single 1-bit input difference can not independently build a MSB-differential or non-differential propagation within the third round, three consecutive steps of MSB-difference across the second and third rounds are needed to produce the fourth MSB difference with a MSB input difference in the third round, which can then lead to a MSB-differential propagation, or a step of specific difference is needed with next three steps of no difference to eliminate the next 1-bit input difference, such that a fourth step with no difference is produced and a non-differential propagation is built in the third round. In this way, first of all, some bit input differences can be located in the beginning part of the third round, which include in the message words involved in the first five steps of the third round both the MSBs ($m_{5,31}$, $m_{8,31}$, $m_{11,31}$, $m_{14,31}$ and $m_{4,31}$) and the non-MSBs ($m_{5,10}$, $m_{8,25}$, $m_{11,21}$, $m_{14,16}$ and $m_{4,25}$, all being turned into a MSB by the left rotation in the final round), where $m_{1,31}$ and $m_{1,10}$ are excluded for too few free steps available in the first round. Secondly, any bit in the message word m_{11} is feasible for collision differential if a local collision can be obtained in the four steps b_8, a_9, d_9 and c_9, since it will then result in a non-differential propagation in the third round. Finally, since the message word m_2 is involved respectively in b_{12} and c_{16}, if a local collision can be obtained in the second round, the bit $m_{2,8}$ can then be used to build a MSB-differential propagation

Table 1. The Weak Input Differences For MD5 Collision Attack

Weak Input Difference	Section I	Section II	Section III	Section IV	#
$m_{4,20}, m_{7,31}, m_{13,31}$	1–3	4–27	28–60	61–64	34
$m_{6,8}, m_{9,31}, m_{15,31}$	1–5	6–21	22–58	59–64	30
$m_{2,31}, m_{9,27}, m_{12,31}$	1	2–29	30–63	64–64	24
► $m_{4,31}, m_{11,15}, m_{14,31}$	1–3	4–22	23–61	62–64	29
$m_{2,8}$	1	2–26	27–62	63–64	25
$m_{11,0-30}$	1–11	12–19	20–61	62–64	25
$m_{5,10}, m_{11,21}$	1–4	5–31	32–61–64		38
$*m_{5,31}, m_{8,31}$	1–4	5–30	31–64		34
$*m_{5,10}, m_{11,31}$	1–4	5–31	32–64		43
$m_{2,8}, m_{14,31}$	1–2	3–29	30–62	63-64	10
★$m_{5,10}, m_{10,31}$	1–4	5–33	34–51–64	1-block collision	
$*m_{4,31}$	1–3	4–34	35–64		34
$*m_{5,31}$	1–4	5–29	30–64		24
$*m_{8,31}$	1–7	8–30	31–64		24
$m_{11,31}$	1–10	11–31	32–64		20
$*m_{11,31}$	1–10	11–31	32–64		35
$*m_{14,31}$	1–13	14–32	33–64		35
$*m_{4,25}$	1–3	4–34	35–64		38
$*m_{5,10}$	1–4	5–29	30–64		23
$*m_{8,25}$	1–7	8–30	31–64		27
$*m_{11,21}$	1–10	11–31	32–64		23
$*m_{14,16}$	1–13	14–32	33–64		33

beginning from the last step b_{12} in the third round to the step c_{16} in the final round, and only two non-MSB differences will generate in the final two steps due to the 1-bit input difference $m_{2,8}$. All 1-bit weak input differences are listed in Table 1. In Table 1, the symbol ► denotes Wang et al's 3-bit input difference, ★ denotes the weak input differences that may construct 1-block collision attack on MD5, $*$ denotes those weak input differences which can result in a pseudo collision in the second block (the first block produces a near collision with four words of MSB difference as chain input differences for the second block)[23]. Section I to Section IV in Table 1 correspond to that of the optimal characterization described in subsection 3.3, which are defined by the beginning and ending steps. The final item # denotes the number of bit conditions being incurred in the final group, which is used to estimate the computational cost of obtaining a collision for each weak input difference, and will be described again in subsection 5.1.

4.2 2-Bit Weak Input Differences

As described in subsection 3.2, since four consecutive steps of MSB difference or no difference can not be produced in the third round completely by only two 1-bit input differences, a near collision across the second and third rounds is then needed so that it can be combined with two 1-bit input differences properly selected from the message words involved in the first five steps of the third round, as given in subsection 4.1. Generally, any pairing of two 1-bit input differences from $m_{5,31}$, $m_{8,31}$, $m_{11,0-31}$, $m_{14,31}$, $m_{4,31}$, $m_{5,10}$, $m_{8,25}$, $m_{14,16}$ and $m_{4,25}$, can be used to produce a feasible collision differential characteristic, if four consecutive steps of MSB difference or no difference can be built at the beginning part of the third round or across the second and third rounds, for examples, pairings like $m_{5,10}$ and $m_{11,21}$, $m_{5,31}$ and $m_{8,31}$, $m_{5,10}$ and $m_{11,31}$, each can be a 2-bit weak input difference. In Table 1, these three 2-bit weak input differences are listed as examples.

Particularly, since the bit $m_{2,8}$ can be independently used to build four consecutive steps of MSB difference, any single 1-bit input difference given above can be combined with $m_{2,8}$ to be a 2-bit weak input difference if it results in a local collision at the beginning part of the third round. However, a bit difference both in $m_{2,8}$ and $m_{14,31}$ can produce a feasible collision differential characteristic which does not need a local collision in the third round, it is particularly listed in Table 1 as the possible fastest 2-bit collision differential to be developed, because of its least number of conditions in the final group to be randomly satisfied. In Table 1, it is worth pointing out that the input difference consisting of $m_{5,10}$ and $m_{10,31}$ is likely to build a 1-block collision attack on MD5, since it will result in a very early collision at the step c_{14}.

4.3 3-Bit Weak Input Differences

Since properly arranging three 1-bit input differences can themselves produce four consecutive steps of MSB difference or no difference within the third round, no near collision is needed to be derived from the second round. By the specific requirement of properly arranging three 1-bit input differences and the differential propagation property in the final round, which is described in subsection 3.2, only the corresponding bits of m_2, m_4, m_6, m_9, m_{11}, m_{13} and m_{15}, which are involved respectively in the ending steps of the final round, can be used to produce the beginning MSB difference in the third round. Due to the permutation of input message words in the third round, m_2 and m_{15} cannot be the beginning MSB difference, thus, only the corresponding bits in m_4, m_6, m_9, m_{11} and m_{13} are qualified as the first input difference, namely, $m_{4,20}$, $m_{6,8}$, $m_{9,27}$, $m_{11,15}$ and $m_{13,27}$. Therefore, we have five 3-bit weak input differences, each consists of 3 words, each word has 1-bit difference. More specifically, $m_{4,20}$, $m_{7,31}$ and $m_{13,31}$ constitute the first 3-bit weak input difference, $m_{6,8}$, $m_{9,31}$ and $m_{15,31}$ the second one, $m_{9,27}$, $m_{12,31}$ and $m_{2,31}$ the third one, $m_{11,15}$, $m_{14,31}$ and $m_{4,31}$ the fourth one, $m_{13,27}$, $m_{0,31}$ and $m_{6,31}$ the fifth one. Being required of enough free steps in the first round, the fifth 3-bit weak input difference cannot be a good collision differential. Actually, these four 3-bit input differences are the traditionally weak input differences, for examples, the triplet of $m_{11,15}$, $m_{14,31}$ and $m_{4,31}$ is firstly proposed by Wang et al in 2005 [2], and the triplet of $m_{6,8}$, $m_{9,31}$ and $m_{15,31}$ by Xie et al in 2008 [21]. These four 3-bit weak input differences are also listed in Table 1.

However, there exist other triplets of 1-bit input differences which are not necessarily arranged traditionally as above. For example, any two MSBs in subsection 4.1 can be combined with $m_{2,8}$ to compose a new 3-bit weak input difference if a non-differential propagation is built in the third round, any three MSBs in subsection 4.1 can constitute a new 3-bit weak input difference if a MSB-differential propagation is built in the third round. In particular, these 3-bit weak input differences can be combined with $m_{2,8}$ to further compose 4-bit weak input differences, and some of them may be better than the original 3-bit collision differential in term of computational complexity.

5 Evaluation Criteria Of Weak Input Differences

5.1 Computational Cost Estimation Of Differential Collision Search

Given a weak input difference, usually the computational cost to obtain a full collision is intrinsically determined by the differential characteristic within the final group of steps beginning from b_6, a_7 or d_7, thus an optimal construction of differential propagation from step b_6, a_7 or d_7 to the step involved with the first 1-bit input difference in the third round is critical to the minimization of computational cost. In general, a collision differential characteristic is constructed both by a backward deduction and a forward deduction, these two deductions are to be met in the first round by a trial-and-error method. A backward deduction beginning from the last involved input difference in the third round will produce a critical differential characteristic to b_6, a_7 or d_7 with minimal Hamming weight, which hence incurs minimal bit conditions. Since the differential characteristic deduction (from the third round to the end of the final round) behind this critical section is a trivial one, it can be uniquely determined to be a single section III or a section III plus a section IV defined in Subsection 3.3, incurring minimal bit conditions. This backward deduction of critical differential characteristic to step b_6, a_7 or d_7 plus the trivial forward deduction of differential characteristic to the final step b_{16} will build an intrinsic differential characteristic within the final group of steps for a given weak input difference.

In this subsection, four different weak input differences are selected from Table 1 as examples to illustrate how to deduce the critical differential section. Table 2 is used to estimate the computational cost

Table 2. The Optimal Backward Deduction of Four Critical Differential Characteristic Sections

t	q_t	w_t	s_t	$\triangle^+m_8=2^{31}$		$\triangle^+m_5=2^{10}$		$\triangle^+m_5=2^{31}$ $\triangle^+m_8=2^{31}$		$\triangle^+m_2=2^{31}$ $\triangle^+m_9=2^{27}$ $\triangle^+m_{12}=2^{31}$	
				$\triangle^{\pm}q_t$	#	$\triangle^{\pm}q_t$	#	$\triangle^{\pm}q_t$	#	$\triangle^{\pm}q_t$	#
22	d_6	m_{10}	9	[ ]		[24,28]		[13,-17]		[-17]	
23	c_6	m_{15}	14	[3]		[-19]		[3,-8]		[ ]	
24	b_6	m_4	20	[29,30]	2	[ ]	4	[-29,31]	6	[11]	3
25	a_7	m_9	5	[ ]	2	[-5]	3	[26]	5	[26]	1
26	d_7	m_{14}	9	[ ]	3	[1]	1	[22]	4	[ ]	2
27	c_7	m_3	14	[17]	1	[ ]	2	[17]	3	[ ]	1
28	b_7	m_8	20	[ ]	0	[ ]	2	[ ]	2	[*31]	0
29	a_8	m_{13}	5	[ ]	2	[-10]	1	[*31]	1	[ ]	0
30	d_8	m_2	9	[ ]	0	[ ]	0	[ ]	1	[ ]	1
31	c_8	m_7	14	[*31]	0	[ ]	1	[*31]	0	[ ]	0
32	b_8	m_{12}	20	[*31]	0	[ ]	0	[*31]	0	[ ]	0
33	a_9	m_5	4	[*31]	0	[ ]	0	[*31]	0	[ ]	0
34	d_9	m_8	11	[*31]	0	[ ]	0	[*31]	0	[ ]	0
35–64				19		14		18			20

of probabilistically satisfying all conditions in the final group of steps. These four weak input differences include respectively $\triangle^+m_8 = 2^{31}$(1-MSB input difference), $\triangle^+m_5 = 2^{10}$(1-bit non-MSB input difference), $\triangle^+m_5 = 2^{31}$ plus $\triangle^+m_8 = 2^{31}$(2-MSB input difference), $\triangle^+m_2 = 2^{31}$, $\triangle^+m_9 = 2^{27}$ plus $\triangle^+m_{12} = 2^{31}$(3-bit input difference). In Table 2, for example, backward from d_9 in which m_8 is involved, there are only two possible differential paths deduced conforming to the optimal characterization of feasible collision differential propagation, which are backward deduced to d_6 starting with four consecutive steps of MSB difference or no difference. However, only one of the two differential paths deduced is feasible, since an incorrectly deduced path will result in a number of additional non-MSB differences in the final round and hence incur a considerable number of additional bit conditions. The bit conditions incurred in the four characteristic sections are all derived up to step b_6.

According to Table 2, the first two 1-bit collision differentials respectively with $\triangle^+m_8 = 2^{31}$ and $\triangle^+m_5 = 2^{10}$ can each result in a MD5 collision attack within 2^{27} and 2^{25} MD5 compressions, while the third 2-bit collision differential with $\triangle^+m_5 = 2^{31}$ plus $\triangle^+m_8 = 2^{31}$ will result in a MD5 collision attack within 2^{34} MD5 compressions, which is much inferior to the first two 1-bit collision differentials. In particular, the second block for the first three collision differentials is only a pseudo collision, which was proposed by Boer and Bosselaers [23], thus making it easier to develop a practical collision attack. Although the final collision differential with 3-bit weak input difference $\triangle^+m_2 = 2^{31}$, $\triangle^+m_9 = 2^{27}$ plus $\triangle^+m_{12} = 2^{31}$ can result in a MD5 collision attack within 2^{25} MD5 compressions, it has 3-bit input differences in two blocks, thus making it more difficult to develop a practical collision attack. In Table 2, t denotes the MD5 steps starting from 1, q_t the state variables from a_1 to b_{16}, w_t the message words from m_0 to m_{15}, s_t the number of bits to be left rotated, $\triangle^{\pm}q_t$ the signed differential representation as defined in [22] with bit index starting from 0, # the number of bit conditions incurred in each step.

In the same way, the computational cost for each weak input difference in Table 1 are estimated by their number of conditions in the final group of steps, which are only probabilistically satisfiable and given in the final column as #. It can be easily found in Table 1 that in term of computational complexity, the collision differential composed of $m_{2,8}$ and $m_{14,31}$ has only 10 conditions in the final group to be randomly satisfied, thus it is currently the fastest 2-bit differential collision attack that has been found for MD5. In addition, there are a considerable number of 1-bit weak input differences that are superior to the traditional 3-bit weak input difference ($m_{4,31}$, $m_{11,15}$ plus $m_{14,31}$), such as $m_{11,0-30}$, $m_{5,10}$, $m_{5,31}$, $m_{2,8}$ and $m_{8,31}$, even 3-bit input differences considered, $m_{2,31}$, $m_{9,27}$ plus $m_{12,31}$ is also a better choice. Since the number of bit conditions incurred in the final group for all 1-bit weak input differences in Table 1 is minimized for the first block, these 1-bit weak input differences from $m_{11,0-30}$ will not produce four consecutive MSB differences for the chain input of the second block. As a result, no pseudo collision can be employed in the second block for each 1-bit input difference $m_{11,0-30}$, most likely this will make the computational cost of the second block larger than that of the first one.

5.2 Criteria to Evaluate Weak Input Differences

All weak input differences can be evaluated by the following seven criteria:

1. Whether or not the differential characteristic depends on the initial vector of hash function;
2. The number of blocks constituting a collision differential;
3. The number of free words in the message;
4. The number of bit differences in the message pair;
5. The number of bit conditions which must be satisfied to yield a collision;
6. The number of bit conditions that can only be probabilistically fulfilled in a group;
7. The averaged computational complexity of finding a collision.

Considering a real-world cryptanalytic attack, a differential characteristic which does not rely on the initial vector will obviously be better than those must rely on it, a collision differential which has more free words, less input differences and bit conditions will be more easily used to construct meaningful attacks, a collision differential with less message blocks and probabilistically fulfilled conditions will be more efficient for practical attacks. The less the bit conditions necessary to maintain a collision differential characteristic, the higher the density of colliding messages will be; the less the average computational complexity of finding a collision, the more feasible an attack on practical protocols based on a hash function will be. For the weak input differences given in Table 1, we can make a comparison based on the above criteria. From Table 1, the collision differential due to each of the three 1-bit weak input differences $m_{5,10}$, $m_{5,31}$ and $m_{8,31}$ exceeds those due to other weak input differences, in terms of free message words, bit differences, bit conditions and especially computational cost. If only computational complexity considered, the collision differential due to $m_{2,8}$ and $m_{14,31}$ will be the fastest differential collision attack on MD5.

5.3 An Example:The Fastest 1-MSB Differential Collision Attack On MD5

By the seven evaluation criteria in Subsection 5.2, we choose the weak input difference $\triangle^{+}m_8 = 2^{31}$ to develop a practical 1-MSB collision attack algorithm. Since a signed bit difference in the differential characteristic will incur several bit conditions and the number of bit conditions will fundamentally determine

Table 3. A Collision Example With The MD5 Digest (Underlined Bits With Difference)

M_0	0x68106ac6, 0x2094ed6b, 0xa3ec34eb, 0xf4383dff, 0x157fe4d, 0xeff04e4e, 0x1119f00b, 0x22172e32, 0x$\underline{\text{c}}$55102b0, 0x99355658, 0x97874ee2, 0x2c408161, 0xf55b1a3f, 0x31e6ad3c, 0x6ed9a43b, 0x4116f7b6
M_1	0xec434329, 0xccab7e9a, 0x32b86260, 0x82c53b56, 0xad5ff512, 0xedeab6b5, 0x3e2c15ea, 0x4a564948, 0x292cf96c, 0x684ad345, 0x63cb649d, 0xc2b7e49e, 0xa7cfd089, 0x127c0548, 0xc2906aa4, 0x66e94d25
M_0'	0x68106ac6, 0x2094ed6b, 0xa3ec34eb, 0xf4383dff, 0x157fe4d, 0xeff04e4e, 0x1119f00b, 0x22172e32, 0x$\underline{4}$55102b0, 0x99355658, 0x97874ee2, 0x2c408161, 0xf55b1a3f, 0x31e6ad3c, 0x6ed9a43b, 0x4116f7b6
M_1'	0xec434329, 0xccab7e9a, 0x32b86260, 0x82c53b56, 0xad5ff512, 0xedeab6b5, 0x3e2c15ea, 0x4a564948, 0x292cf96c, 0x684ad345, 0x63cb649d, 0xc2b7e49e, 0xa7cfd089, 0x127c0548, 0xc2906aa4, 0x66e94d25
MD5 Digest	0xa6c8489d, 0xddce2a29, 0x7ae49ec2, 0x7464879f

the feasibility of the corresponding differential collision attack, the objective of designing a collision differential characteristic is to minimize its Hamming weight. We follow the designing rules described in [21,22] to construct 2-block differential characteristics, while the first block characteristic (See Appendix B: Table 4 and its tunneling version Table 5) produces a near collision with only MSB difference in each chain variable differential, the second block characteristic (See Appendix B: Table 6 and its tunneling version Table 7) leads to a full collision through Den Boer and Bosselaer's pseudo collision.
Besides the basic conditions in Table 4–7 that must be satisfied, by Theorems 4 and 5 in [22], some extra conditions must also be satisfied to prevent possible occurrence of some unexpected modular differences due to the rotation operation plus carries or overflow, which may occur when $\sum a_{i+1}^{<<<s_j}$ is executed. Therefore, the set of sufficient conditions includes both the basic conditions and the extra conditions, and thanks to the logic OR operation on extra conditions, most of the extra conditions can be fulfilled with much high probabilities. By Theorems 4 and 5 in [22], no extra conditions are needed for the differential path of the second block, and the extra conditions for the first block are included with the following groups of equations: $\sum b_{3,6-9} = 0$, $\sum a_{5,31} = 0$, $\sum d_{5,29-31} = 0$, $\sum c_{5,15-17} = 1$, $\sum b_{5,28-31} = 1$, $\sum a_{6,24-26} = 1$, $\sum d_{6,17-22} = 0$, $\sum c_{6,22-31} = 0$, $\sum b_{6,9-11} \neq 101$, $\sum a_{7,26} = 1$, $\sum c_{7,3-17} = 0$, $\sum b_{7,29-31} = 1$, $\sum c_{8,17} = 0$.
Except the inequality $\sum b_{6,9-11} \neq 101$ must be held as a whole, other equalities define a logic OR operation on their consecutive bits. For example, $\sum b_{3,6-9} = 0$ requires $\sum b_{3,6} = 0$, or $\sum b_{3,7} = 0$, or $\sum b_{3,8} = 0$, or $\sum b_{3,9} = 0$, which can be satisfied with a probability of $(1 - 0.5^4) = 0.9375$.
Based on the general procedure of collision searching algorithm derived from the optimal grouping scheme for MD5 differential collision attack, a specific collision attack algorithm with a time complexity of $2^{20.96}$ MD5 compressions is developed, which is currently the fastest MD5 collision attack algorithm, being able to generate a collision averagely in 0.45 sec. on a 2.6Ghz Pentium PC. By properly grouping all bit conditions, the multiplicative computational complexities has been particularly transformed into an additive accumulation. As a result, the actual computational complexity is dramatically reduced. The collision searching algorithm has been implemented which is available from the web site `http://www.is.iscas.ac.cn/gnomon`. An example of collision pair $\{M_0, M_1\}$ and $\{M'_0, M'_1\}$ is given in Table 3. For the algorithmic details, please refer to [22].

6 Summary and Conclusion

This paper mainly addresses the problem of how to select weak input differences that can easily lead to MD5 collisions. Firstly, a divide-and-conquer strategy is introduced with an optimal scheme of properly grouping MD5 steps to greatly improve the satisfaction of bit conditions especially in rounds behind the first round, and a five groups of separation of MD5 steps is obtained. Secondly, differential propagation properties are analyzed for each round, and this, together with the optimal grouping scheme, results in an optimal characterization of the weak differential propagation. Thirdly, all 1-bit to 3-bit weak input differences that can produce a feasible collision differential characteristic are thus selected by the optimal characterization. Fourthly, the computational cost is estimated for each weak input difference, which reveals that there exist a great number of weak input differences better than Wang et al's 3-bit collision differential, and a 2-bit collision differential composed of $m_{2,8}$ and $m_{14,31}$ will be the fastest differential collision attack on MD5 that has been found. Finally, a general procedure for MD5 collision searching is proposed, which is a generic implementation of the optimal grouping scheme, and a new example of differential collision attack on MD5 is given with currently the lowest computational complexity of $2^{20.96}$ MD5 compressions.

The result of this paper provides a large resource of colliding messages with different weak input differences, indirectly makes a second pre-image attack for an arbitrarily given message much easier than previously relying only on the 3-bit collision differential. Since this paper, the selection of weak input differences is no more a work relying on one's intuition and experiences commonly regarded as previously, it becomes a reasonable process of deduction, by which people can more easily find new collisions. By the estimation on computational costs of all weak input differences and further considering the second block and other evaluation criteria, only two 1-bit weak input differences in m_5, i.e. $\triangle^+ m_5 = 2^{10}$ and $\triangle^+ m_5 = 2^{31}$, can each be comparable to $\triangle^+ m_8 = 2^{31}$, which is currently the fastest 1-MSB collision attack [22]. Particularly, since a single block collision for MD5 has never been found so far, it remains an open problem to confirm if the 2-bit input difference consisting of $\triangle^+ m_5 = 2^{10}$ and $\triangle^+ m_{10} = 2^{31}$ can be a feasible 1-block collision attack on MD5.

Acknowledgements

Part of this work is supported by MOST of China through the 973 program under contract 2007CB311202, and by National Natural Science of China through the 60473011 project.

References

1. X.Y. Wang, D. G. Feng, X.J. Lai, H.B. Yu. Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD.Rump session of Crypto'04, E-print, 2004.
2. X.Y. Wang, Hongbo Yu. How to Break MD5 and Other Hash Functions. EUROCRYPT 2005, LNCS 3494, pp.19-35, Springer-Verlag, 2005.
3. X.Y Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. Crypt'2005, LNCS 3621, pp17 36.
4. Vlastimil Klima. Tunnels in Hash Functions: MD5 Collisions Within a Minute. Cryptology ePrint Archive, Report 2006/105, 2006. http://eprint.iacr.org/.
5. Marc Stevens. On Collisions for MD5, Master's Thesis, 2007. TU Eindhoven, Faculty of Mathematics and Computer Science, available at `http://www.win.tue.nl/hashclash/`.
6. Marc Stevens, Arjen Lenstra, and Benne de Weger. Chosen-prefix collisions for MD5 and colliding X.509 certificates for different identities. EUROCRYPT 2007, LNCS4515, pp1-22.
7. C. De Cannire and C. Rechberger. Finding SHA-1 Characteristics: General Results and Applications.ASIACRYPT 2006, LNCS4284, pp1-20.
8. C. De Canni'ere, F. Mendel and C. Rechberger. Collisions for 70-step SHA-1: On the Full Cost of Collision Search. Selected Areas in Cryptography - SAC 2007, LNCS4876, pp.56-73.
9. A. Joux and T. Peyrin. Hash Functions and the (Amplified) Boomerang Attack.CRYPTO 2007, LNCS 4622, pp 244-263.
10. Magnus Daum and Stefan Lucks. Hash Collisions (The Poisoned Message Attack) "The Story of Alice and her Boss". Presented at the rump session of Eurocrypt '05.
11. M. Gebhardt, G. Illies, and W. Schindler. A Note on the Practical Value of Single Hash Collisions for Special File Formats. In Jana Dittmann, editor, Sicherheit, volume 77 of LNI, pages 333-344. GI, 2006.
12. G. Leurent.Message Freedom in MD4 and MD5 Collisions: Application to APOP. Fast Software Encryption 2007, LNCS4593, pp 309-328.
13. Y. Sasaki, L. Wang, K. Ohta, and N. Kunihiro. Security of MD5 Challenge and Response: Extension of APOP Password Recovery Attack. CT-RSA 2008, LNCS 4964, pp1-18.
14. E. Andreeva, C. Bouillaguet, P.-A. Fouque, J. J. Hoch, J. Kelsey, A. Shamir, and S. Zimmer. Second Preimage Attacks on Dithered Hash Functions. EUROCRYPT 2008, LNCS 4965, pp 270-288.
15. C. D. Cannire and C. Rechberger. Preimages for Reduced SHA-0 and SHA-1. CRYPTO 2008, LNCS5157, pp179-202.
16. Y. Sasaki, K. Aoki. Finding Preimages in Full MD5 Faster than Exhaustive Search. EUROCRYPT 2009, LNCS , pp. .
17. S. Contini and Y. L. Yin. Forgery and Partial Key-Recovery Attacks on HMAC and NMAC Using Hash Collisions. ASIACYPT2006, LNCS 4284, 2006.
18. Jongsung Kim, Alex Biryukov, Bart Preneel, and Seokhie Hong. On the Security of HMAC and NMAC Based on HAVAL, MD4, MD5, SHA-0 and SHA-1. In Roberto De Prisco and Moti Yung, editors, SCN 2006, LNCS4116, pp 242-256.
19. Xiaoyun Wang, Wei Wang, Hongbo Yu, Haina Zhang, Tao Zhan. Cryptanalysis on HMAC/NMAC-MD5 and MD5-MAC, EUROCRYPT 2009. LNCS , pp. .
20. Y. Sasaki, L. Wang, N. Kunihiro, and K. Ohta. New Message Differences for Collision Attacks on MD4 and MD5. IEICE Transactions,91-A(1):55-63, 2008.
21. Tao Xie, Dengguo Feng, Fanbao Liu. A New Collision Differential For MD5 With Its Full Differential Path. Cryptology ePrint Archive (2008/230), `http://eprint.iacr.org/`.

22. Tao Xie, Fanbao Liu, Dengguo Feng. Could The 1-MSB Input Difference Be The Fastest Collision Attack For MD5? Accepted to be presented at the poster session of EUROCRYPT 2009. Cryptology ePrint Archive (2008/391), http://eprint.iacr.org/.
23. B. den. Boer, A. Bosselaers. Collisions for the compression function of MD5.Advances in Cryptology, Eurocrypt'93 Proceedings, Springer-Verlag, 1994.
24. F. Chabaud and A. Joux. Differential Collisions in SHA-0. CRYPTO'98, LNCS1462, pp.56 71. Springer-Verlag, 1998.

Appendix A: MD5 Function

Practically, a Merkle-Damgard structure-based hash function is iterated by a compression function $y = f(x)$, which compresses l-bit message block x to an s-bit hash value y, where $l > s$. For MD5, $l = 512$, $s = 128$. For a padded message M with multiple (t) of l-bit blocks, the iteration process can be described as: $h_{i+1} = f(h_i, M_i)$, $0 \le i \le t-1$, where $M = (M_0, M_1, \dots, M_{t-1})$, h_i is the 128-bit chaining variable (including four 32-bit words) which is updated during the processing of each block, h_0 is the prescribed initial value in MD5 algorithm, and the final h_t is the digest that we expect to obtain. The concrete padding rule is omitted here, since it has no influence on our attack. The whole process of the i^{th} block can be defined as follows:

$$h_{i+1} = f(h_i, M_i) = h_i + II(M_i, HH(M_i, GG(M_i, FF(M_i, H_i)))). \tag{1}$$

Where the four round functions FF, GG, HH and II are involved. All round functions are similar to one another in structure. The chaining variable h_i is treated as a four-element shift register, with each element being a 32-bit word, referred to as a_0, b_0, c_0 and d_0, respectively. Each 512-bit block M_i is divided into 16 32-bit words, denoted as $M_i = (m_0, m_1, m_2, \dots, m_{15})$, each round consists of 16 steps of operation, in each step of operation, the register is updated with one word from M_i. The 64 steps of operation form a system of equations:

$$a_{i+1} = b_i + (a_i + \Phi_j(b_i, c_i, d_i) + w_j + t_j)^{<<<s_j}. \tag{2}$$

Where $0 \le i \le 16$, $1 \le j \le 16$, a_i, b_i, c_i and d_i ($1 \le i \le 16$) are the internal state variables, $\Phi_j(x, y, z)$ is an auxiliary bitwise function which varies from round to round, w_j is a word chosen from $(m_0, m_1, m_2, \dots, m_{15})$ by a round-wise message permutation $\sigma_k(i)$, $k = 0, 1, 2, 3$, $i = 0, 1, 2, \dots, 15$, t_j and s_j are constant parameters associated with step j. Note that each step operation involves four modular additions ($mod\, 2^{32}$), an auxiliary function $\Phi_j(x, y, z)$ and a left rotation of s_j bits $<<< s_j$. As the step operation of MD5 is reversible, the compression function $f(h_i, M_i)$ uses a feed-forward operation which adds the initial value h_i of the register to their final values, so that $f(h_i, M_i)$ cannot be inverted (the Davies-Meyer construction).

For the sake of understanding how and where some extra conditions are derived from, which are used to prevent the possible unexpected modular differences due to the joint effect of both modular addition and left rotation, we define the part of step operation as

$$\sum a_{i+1} = a_i + \Phi_j(b_i, c_i, d_i) + w_j + t_j. \tag{3}$$

The auxiliary functions and the round-wise permutations $\sigma_k(i)$ for each round are given as follows:

$$\Phi_j(x, y, z) = F(x, y, z) = (x \wedge y) \vee (\bar{x} \wedge z),\ 1 \le j \le 16; \tag{4}$$

$$\Phi_j(x, y, z) = G(x, y, z) = (x \wedge z) \vee (y \wedge \bar{z}),\ 17 \le j \le 32; \tag{5}$$

$$\Phi_j(x, y, z) = H(x, y, z) = x \oplus y \oplus z, \qquad 33 \le j \le 48; \tag{6}$$

$$\Phi_j(x, y, z) = I(x, y, z) = y \oplus (x \vee \bar{z}), \qquad 49 \le j \le 64; \tag{7}$$

$$w_{j+1} = \begin{cases} m_j, & 0 \le j < 16; \\ m_{1+5j\ mod\, 16}, & 16 \le j < 32; \\ m_{5+3j\ mod\, 16}, & 32 \le j < 48; \\ m_{7j\ mod\, 16}, & 48 \le j < 64. \end{cases} \tag{8}$$

Where x, y and z are 32-bit words. The auxiliary functions $\Phi_j(x, y, z)$ each takes three consecutive 32-bit words from the register of chaining variables as input and produces a 32-bit word as output. The four words in the chaining variable register are initialized as: a_0=0x67452301, b_0=0xefcdab89, c_0=0x98badcfe, d_0=0x10325476.

Appendix B: The Differential Paths and Conditions

In Tables 4–7, '+' denote a positive flip ($0 \rightarrow 1$), '-' a negative flip ($1 \rightarrow 0$), 0(1) the conditional bit value, '^' denotes the bit must be equal to the up bit, '!' the bit not equal to the up bit, '*' the free bit, 't' the MD5 step, '#' the number of conditions for each step.

Table 4. The Basic Differential Path Using $\triangle^{+} m_8 = 2^{31}$(Block1)

t	Bits Q_t: $a_0 ... a_{31}$	#
1-6	******** ******** ******** ********	0
7	******1* ******** ******** ********	1
8	******1* ******** ******** ********	1
9	******+* ******** ******** ********	1
10	******+* ******** ******** ****1***	2
11	******+* ******0* ******** ****1^**	4
12	******+0 0*010*00 **111**0 *1**-+0*	16
13	***1*0+1 1^001^+0 ^^010^^1 ^0**+11^	26
14	*111*1-+ -----+-- --+----- -+*0---+	29
15	*01-0-1+ 00+-01+- 100-+111 01*01000	30
16	*+-*1011 10-+0101 100+1101 01*+-001	29
17	***^-+-1 **-1**11 *1*-+101 00**11**	19
18	*^^***** *001***0 *+*0*011 +1*^-+**	16
19	****^^^* *111***0 ***1^--- -+001***	17
20	******** *-*+***- *^****** 11111^**	10
21	***0**** ******** *****^^^ +0++-***	9
22	***1**** *^*^***^ ******** *****00*	6
23	***+**** ******** ******** ^*^^^11*	7
24	******** ******** ******** *****++*	2
25	***^**** ******** *0****** ********	2
26	******** ******** *1****** *****^^*	3
27	******** ******** *+****** ********	1
28	******** ******** ******** ********	0
29	******** ******** *^****** *******0	2
30	******** ******** ******** ********	0
31-47	******** ******** ******** *******-	0
48-60	******** ******** ******** *******-	13
61	******** ******** ******** *******-	1
62	******** ******** ******** *******+	1
63	******** ******** ******** *******-	1
64	******** ******** ******** *******+	0

Table 5. The Modified Differential Path With Additional Absorbing Bits (Block1).

t	Bits Q_t: $a_0 ... a_{31}$	#
1-3	******** ******** ******** ********	0
4	^^^^**** **^***** ******** ****^^^^	9
5_t	****0000 00*00000 00000000 0000****	23
6_t	00001111 11011111 11111111 11110000	32
7	******1* ******** ******** ********	1
8	*****11* **^****^ !******* ^*!^****	9
9	*****0+* ***^**** *^*^***^ ********	6
10_t	00000*+0 000*00*0 0*0*000* 00001*00	25
11_t	11111*+1 11101101 10101110 11111^11	31
12	******+0 0*010*00 **111**0 *1**-+0*	16
13	***1*0+1 1^001^+0 ^^010^^1 ^0**+11^	26
14	*111*1-+ -----+-- --+----- -+*0---+	29
15	*01-0-1+ 00+-01+- 100-+111 01*01000	30
16	*+-*1011 10-+0101 100+1101 01*+-001	29
17	***^-+-1 **-1**11 *1*-+101 00**11**	19
18	*^^***** *001***0 *+*0*011 +1*^-+**	16
19	****^^^* *111***0 ***1^--- -+001***	17
20	******** *-*+***- *^****** 11111^**	10
21	***0**** ******** *****^^^ +0++-***	9
22	***1**** *^*^***^ ******** *****00*	6
23	***+**** 1******* ****11** ^*^^^11*	10
24	******** ******** ******** *****++*	2
25	***^**** ******** *0****** ********	2
26	******** ******** *1****** *****^^*	3
27	******** ******** *+****** ********	1
28	******** ******** ******** ********	0
29	******** ******** *^****** *******0	2
30	******** ******** ******** ********	0
31-47	******** ******** ******** *******-	0
48-60	******** ******** ******** *******-	13
61	******** ******** ******** *******-	1
62	******** ******** ******** *******+	1
63	******** ******** ******** *******-	1
64	******** ******** ******** *******+	0

Table 6. The Basic Differential Path Using $\triangle^{+}m_i = 0, 0 \le i < 16$ (Block2).

t	Bits Q_t: $a_0 ... a_{31}$	#
-3	******** ******** ******** *******+	0
-2	******** ******** ******** *******+	0
-1	******** ******** ******** *******+	1
0	******** ******** ******** *******+	1
1-31	******** ******** ******** *******+	31
32-47	******** ******** ******** *******+	0
48-63	******** ******** ******** *******+	16

Table 7. The Modified Differential Path With Additional Absorbing Bits (Block2).

t	Bits Q_t: $a_0 ... a_{31}$	#
-3	******** ******** ******** *******+	0
-2	******** ******** ******** *******+	0
-1	******** ******** ******** *******+	1
0	******** ******** ******** *******+	1
1-7	******** ******** ******** *******+	7
8	******^* ******** ******** ^****^*+	4
9	******** ******** ******** *******+	1
10_t	00000000 00000000 00000000 0000000+	32
11_t	11111111 11111111 11111111 1111111+	32
12-31	******** ******** ******** *******+	20
32-47	******** ******** ******** *******+	0
48-63	******** ******** ******** *******+	16