Fast Collision Attack on MD5

Marc Stevens

Department of Mathematics and Computer Science,
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands.
m.m.j.stevens@student.tue.nl

Abstract. In this paper, we present an improved attack algorithm to find two-block collisions of the hash function MD5. The attack uses the same differential path of MD5 and the set of sufficient conditions that was presented by Wang et al. We present a new technique which allows us to deterministically fulfill restrictions to properly rotate the differentials in the first round. We will present a new algorithm to find the first block and we will use an algorithm of Klima to find the second block. To optimize the inner loop of these algorithms we will optimize the set of sufficient conditions. We also show that the initial value used for the attack has a large influence on the attack complexity. Therefore a recommendation is made for 2 conditions on the initial value of the attack to avoid very hard situations if one has some freedom in choosing this initial value. Our attack can be done in an average of about 1 minute (avg. complexity $2^{32.3}$) on a 3Ghz Pentium4 for these random recommended initial values. For arbitrary random initial values the average is about 5 minutes (avg. complexity $2^{34.1}$). With a reasonable probability a collision is found within mere seconds, allowing for instance an attack during the execution of a protocol.

Keywords: MD5, collision, differential cryptanalysis

1 Introduction

Hash functions are among the primitive functions used in cryptography, because of their one-way and collision free properties. They are used in a wide variety of security applications such as authentication schemes, message integrity codes, digital signatures and pseudo-random generators. MD5 [4] is a hash function developed by Rivest in 1992 and is based on the Merkle-Damgård construction.

Recently there have been significant advances in the cryptanalysis of MD5. In 2004, Wang et al[6] presented the first MD5 collision. Later in [7] they presented their differential path and a set of sufficient conditions for this differential path to happen. They also introduced message modification techniques to efficiently find message blocks for which the conditions hold. In March 2005, Klima[2] presented a new collision search algorithm to find collisions on a 1Ghz desktop pc in 4 hours. While writing this, two new results[5,3] have been published which improve Wang's message modification techniques. In [3] a running time of 5 hours on a 1.7Ghz Pentium4 is mentioned.

In our attack we will use Klima's algorithm to find the second block and we will present a new algorithm based on Klima's to find the first block. Both algorithms can deterministically choose message blocks that satisfy the set of sufficient conditions for the first round. However certain restrictions necessary for differentials to rotate properly are fulfilled probabilistically. With our new technique we can also deterministically satisfy these restrictions for the first round. Furthermore we optimize the set of conditions to optimize the inner loop of the algorithms. This allows us to find full two-block collisions in an average of 5 minutes on a 3Ghz Pentium4. We will show that the complexity of finding collisions based on Wang's differential in general depends on the initial value. To avoid some worst case initial values we recommend 2 conditions on the initial value. This reduces the average to 67 seconds.

2 MD5 compression function

We use the same description of the compression function in MD5 as is used in [1]. The compression function will take as input a block m of 512 bits together with an intermediate hash value IHV of 128 bits and outputs the new intermediate hash value IHV' of 128 bits. The IHV for the first block is fixed in MD5 and is called the MD5 initial value.

We will denote $RL(x, n)$ and $RR(x, n)$ for resp. left and right rotation of the word (32-bits unsigned integer) x over n bits. Each block m is split into 16 words $m_0, \ldots, m_{15}$ and is expanded into a series of 64 words W_t:

$$W_t = \begin{cases} m_t, & 0 \leq t < 16; \\ m_{1+5t \bmod 16}, & 16 \leq t < 32; \\ m_{5+3t \bmod 16}, & 32 \leq t < 48; \\ m_{7t \bmod 16}, & 48 \leq t < 64; \end{cases}$$

The compression function has a working state of $Q_t, Q_{t-1}, Q_{t-2}, Q_{t-3}$ which is initialized to the intermediate hash value IHV split into 4 words $IHV_0, \ldots, IHV_3$ in the following order:

$$Q_0 = IHV_1, \quad Q_{-1} = IHV_2, \quad Q_{-2} = IHV_3, \quad Q_{-3} = IHV_0.$$

There are 64 steps in the compression function where addition modulo 2^{32}, left rotation and a nonlinear function f_t are used. Here

$$f_t(X, Y, Z) = \begin{cases} (X \wedge Y) \oplus (\overline{X} \wedge Z), & 0 \leq t < 16; \\ (Z \wedge X) \oplus (\overline{Z} \wedge Y), & 16 \leq t < 32; \\ X \oplus Y \oplus Z, & 32 \leq t < 48; \\ Y \oplus (X \vee \overline{Z}), & 48 \leq t < 64. \end{cases}$$

Now step t for $t = 0, \ldots, 63$ is defined as

$$T_t = f_t(Q_t, Q_{t-1}, Q_{t-2}) + Q_{t-3} + AC_t + W_t;$$
$$R_t = RL(T_t, RC_t); \quad Q_{t+1} = Q_t + R_t.$$

Here AC_t and RC_t are resp. the addition and rotation constants defined in MD5[4].

3 Conditions on Q_t for block 1

In [1] a thorough analysis of the collisions presented by Wang et al. is presented. Not only a set of sufficient conditions on Q_t, similarly as those presented in [7], is derived but also a set of necessary restrictions on T_t for the differential to be realized. These restrictions are necessary to correctly rotate the add-differences in T_t to R_t. Efficiently finding collisions can therefore be done by finding message blocks that satisfy the sufficient conditions on Q_t and the necessary restrictions on T_t.

Fast collision finding algorithms as presented in [2] can choose message blocks which satisfy the conditions for $Q_1, \ldots, Q_{16}$. However the restrictions on T_t are fulfilled probabilistically.

The technique presented here allows to fulfill the restrictions on T_t by using extra conditions on Q_{t+1} and Q_t. By using the relation $Q_{t+1} - Q_t = R_t = RL(T_t, RC_t)$ we can control specific bits in T_t. In our analysis we searched for those restrictions on T_t, for $t = 0, \ldots, 20$, with a significant probability that they are not fulfilled. All these restrictions can be found in [1] with a description why they are necessary for the differential.

To compactly describe bitconditions on Q_t we use the notation $Q_t[b]$ ($T_t[b]$) for bit b in the variable Q_t (T_t) for $t = -3, \ldots, 64$ and $b = 0, \ldots, 31$. Bits are shown in the order $Q_t[31] \ldots Q_t[0]$. For $Q_t[b]$ we adopt the following notation:

$$Q_t[b] = \begin{cases} \texttt{'.'} & \text{if there is no restriction} \\ \texttt{'0'}, \texttt{'1'} & \text{if } Q_t[b] \text{ must be the value 0 or 1} \\ \texttt{'\~'} & \text{if } Q_t[b] \text{ must be equal to } Q_{t-1}[b] \\ \texttt{'!'} & \text{if } Q_t[b] \text{ must not be equal to } Q_{t-1}[b] \end{cases}$$

An overview of all conditions for block 1 is included in the tables A-1 and A-2.

3.1 Restriction: $\delta T_4 = -2^{31}$

The condition $T_4[31] = 1$ is necessary and sufficient for $\delta T_4 = -2^{31}$ to happen. Bit 31 of T_4 is equal to bit 6 of R_4, since T_4 is equal to $RR(R_4, 7)$. By adding the conditions $Q_4[4] = Q_4[5] = 1$ and $Q_5[4] = 0$ to the conditions $Q_4[6] = Q_5[6] = 0$ and $Q_5[5] = 1$, it is guaranteed that $T_4[31] = 1$. Satisfying other Q_t conditions, this also implies that $Q_6[4] = Q_5[4] = 0$.

$Q_5[6-4]$	010 ⋯
$Q_4[6-4]$	011 ⋯ −
$R_4[6-4]$	11. ⋯ =

(The original set of sufficient conditions in [7] are shown in black, the added conditions for T_t will be underlined and in blue.)

3.2 Restriction: add-difference -2^{14} in δT_6 must propagate to at least bit 15

This restriction implies that $T_6[14]$ must be zero. Since $T_6[14] = R_6[31]$, the condition $T_6[14] = 0$ is guaranteed by the added conditions $Q_6[30-28, 26] = 0$. This also implies that $Q_5[30-28, 26] = 0$ because of other conditions on Q_t.

$Q_7[31-23]$	000000111 ⋯
$Q_6[31-23]$	000000 1.0 ⋯ −
$R_6[31-23]$	0000000.. ⋯ =

Note: In [8] these conditions were also found by statistical means.

3.3 Restriction: add-difference $+2^{13}$ in δT_{10} must not propagate past bit 14

The restriction is satisfied by the condition $T_{10}[13] = R_{10}[30] = 0$. The conditions $Q_{11}[29-28] = Q_{10}[29] = 0$ and $Q_{10}[28] = 1$ are sufficient.

$Q_{11}[31-28]$	0010 ⋯
$Q_{10}[31-28]$	0111 ⋯ −
$R_{10}[31-28]$	101. ⋯ =

3.4 Restriction: add-difference -2^8 in δT_{11} must not propagate past bit 9

This restriction can be satisfied by the condition $T_{11}[8] = R_{11}[30] = 1$. With the above added condition $Q_{11}[29] = 1$ we only need the extra condition $Q_{12}[29] = 0$.

$Q_{12}[31-29]$	000 ⋯
$Q_{11}[31-29]$	001 ⋯ −
$R_{11}[31-29]$	11. ⋯ =

3.5 Restriction: add-difference -2^{30} in δT_{14} must not propagate past bit 31

For T_{14} the add difference -2^{30} must not propagate past bit 31, this is satisfied by either $T_{14}[30] = R_{14}[15] = 1$ or $T_{14}[31] = R_{14}[16] = 1$. This always happens when $Q_{15}[16] = 0$ and can be shown for the case if no carry from the lower order bits happens as well as the case if a negative carry does happen. A positive carry is not possible since we are subtracting.

no carry

$Q_{15}[16-15]$	01 ⋯
$Q_{14}[16-15]$	11 ⋯ −
$R_{14}[16-15]$	10 ⋯ =

negative carry from lower bits

$Q_{15}[16-15]$	01 ⋯
$Q_{14}[16-15]$	11 ⋯ −
$R_{14}[16-15]$	01 ⋯ =

3.6 Restriction: add-difference -2^7 in δT_{15} must not propagate past bit 9

This can be satisfied by the added condition $Q_{16}[30] = \overline{Q_{15}[30]}$. Since then either $T_{15}[7] = R_{15}[29] = 1, T_{15}[8] = 1$ or $T_{15}[9] = 1$ holds. This can be shown if we distinguish between $Q_{15}[30] = 0$ and $Q_{15}[30] = 1$ and also distinguish whether or not a negative carry from the lower order bits happens.

no carry			negative carry from lower bits		
$Q_{16}[31-29]$	0$\underline{0}$1	$\cdots$	$Q_{16}[31-29]$	0$\underline{0}$1	$\cdots$
$Q_{15}[31-29]$	0$\underline{1}$1	$\cdots -$	$Q_{15}[31-29]$	0$\underline{1}$1	$\cdots -$
$R_{15}[31-29]$	110	$\cdots =$	$R_{15}[31-29]$	101	$\cdots =$

no carry			negative carry from lower bits		
$Q_{16}[31-29]$	0$\underline{1}$1	$\cdots$	$Q_{16}[31-29]$	0$\underline{1}$1	$\cdots$
$Q_{15}[31-29]$	0$\underline{0}$1	$\cdots -$	$Q_{15}[31-29]$	0$\underline{0}$1	$\cdots -$
$R_{15}[31-29]$	010	$\cdots =$	$R_{15}[31-29]$	001	$\cdots =$

3.7 Restriction: add-difference $+2^{25}$ in δT_{15} must not propagate past bit 31

This is satisfied by the added condition $Q_{16}[17] = \overline{Q_{15}[17]}$. Since then either $T_{15}[25] = R_{15}[15] = 0$, $T_{15}[26] = 0$ or $T_{15}[27] = 0$ holds. We compactly describe all cases by mentioning which values were assumed for each result:

no carry

$Q_{16}[17-15]$	...	$\cdots$	
$Q_{15}[17-15]$	$\underline{!}$01	$\cdots -$	
$R_{15}[17-15]$	011	$\cdots =$	$(Q_{16}[17-15] = .00)$
	100	$\cdots$	$(Q_{16}[17-15] = .01)$
	101	$\cdots$	$(Q_{16}[17-15] = .10)$
	110	$\cdots$	$(Q_{16}[17-15] = .11)$

negative carry from lower bits

$Q_{16}[17-15]$	...	$\cdots$	
$Q_{15}[17-15]$	$\underline{!}$01	$\cdots -$	
$R_{15}[17-15]$	010	$\cdots =$	$(Q_{16}[17-15] = .00)$
	011	$\cdots$	$(Q_{16}[17-15] = .01)$
	100	$\cdots$	$(Q_{16}[17-15] = .10)$
	101	$\cdots$	$(Q_{16}[17-15] = .11)$

3.8 Restriction: add-difference $+2^{24}$ in δT_{16} must not propagate past bit 26

This can be achieved with the added condition $Q_{17}[30] = \overline{Q_{16}[30]}$, since then always either $T_{16}[24] = R_{16}[29] = 0$ or $T_{16}[25] = R_{16}[30] = 0$.

no carry

$Q_{17}[30-29]$	..	$\cdots$	
$Q_{16}[30-29]$	$\underline{!}$1	$\cdots -$	
$R_{16}[30-29]$	01	$\cdots =$	$(Q_{17}[30-29] = 00)$
	10	$\cdots$	$(Q_{17}[30-29] = 01)$
	01	$\cdots$	$(Q_{17}[30-29] = 10)$
	10	$\cdots$	$(Q_{17}[30-29] = 11)$

negative carry from lower bits

$Q_{17}[30-29]$	..	$\cdots$	
$Q_{16}[30-29]$	!1	$\cdots$	$-$
$R_{16}[30-29]$	00	$\cdots$	$=$ $(Q_{17}[30-29]=00)$
	01	$\cdots$	$(Q_{17}[30-29]=01)$
	00	$\cdots$	$(Q_{17}[30-29]=10)$
	01	$\cdots$	$(Q_{17}[30-29]=11)$

3.9 Restriction: add-difference -2^{29} in δT_{19} must not propagate past bit 31

This can be achieved with the added condition $Q_{20}[18] = \overline{Q_{19}[18]}$, since then always either $T_{19}[29] = 1$ or $T_{19}[30] = 1$.

no carry

$Q_{20}[18-17]$	..	$\cdots$	
$Q_{19}[18-17]$	!0	$\cdots$	$-$
$R_{19}[18-17]$	10	$\cdots$	$=$ $(Q_{20}[18-17]=00)$
	11	$\cdots$	$(Q_{20}[18-17]=01)$
	10	$\cdots$	$(Q_{20}[18-17]=10)$
	11	$\cdots$	$(Q_{20}[18-17]=11)$

negative carry from lower bits

$Q_{20}[18-17]$	..	$\cdots$	
$Q_{19}[18-17]$	!0	$\cdots$	$-$
$R_{19}[18-17]$	01	$\cdots$	$=$ $(Q_{20}[18-17]=00)$
	10	$\cdots$	$(Q_{20}[18-17]=01)$
	01	$\cdots$	$(Q_{20}[18-17]=10)$
	10	$\cdots$	$(Q_{20}[18-17]=11)$

3.10 Restriction: add-difference $+2^{17}$ in δT_{22} must not propagate past bit 17

It is possible to satisfy this restriction with two Q_t conditions. However T_{22} will always be calculated in the algorithm we used, therefore it is better to verify directly that $T_{22}[17] = 0$. This restriction holds for both block 1 and 2.

3.11 Restriction: add-difference $+2^{15}$ in δT_{34} must not propagate past bit 15

This restriction also holds for both block 1 and 2 and it should be verified with $T_{34}[15] = 0$.

4 Conditions on Q_t for block 2

Using the same technique as in the previous section we found 19 Q_t-conditions for block 2. An overview of all conditions for block 2 is included in the tables A-3 and A-4.

- Restriction: add-difference $+2^{31}$ in δT_2 must not propagate past bit 31.
  Conditions: $Q_1[16] = Q_2[16] = Q_3[15] = 0$ and $Q_2[15] = 1$.

- Restriction: add-difference -2^{31} in δT_6 must propagate past bit 31.
  Conditions: $Q_6[14] = 1$ and $Q_7[14] = 0$.

- Restriction: add-difference -2^{31} in δT_8 must propagate past bit 31.
  Conditions: $Q_8[5] = 1$ and $Q_9[5] = 0$.

– Restriction: add-difference -2^{27} in δT_{10} must not propagate past bit 31.
Conditions: $Q_{10}[11] = 1$ and $Q_{11}[11] = 0$.

– Restriction: add-difference -2^{12} in δT_{13} must not propagate past bit 19.
Conditions: $Q_{13}[23] = 0$ and $Q_{14}[23] = 1$.

– Restriction: add-difference $+2^{30}$ in δT_{14} must not propagate past bit 31.
Conditions: $Q_{15}[14] = 0$.

– Restriction: add-difference -2^{25} in δT_{15} must not propagate past bit 31.
Conditions: $Q_{16}[17] = \overline{Q_{15}[17]}$.

– Restriction: add-difference -2^{7} in δT_{15} must not propagate past bit 9.
Conditions: $Q_{16}[28] = 0$.

– Restriction: add-difference $+2^{24}$ in δT_{16} must not propagate past bit 26.
Conditions: $Q_{17}[30] = \overline{Q_{16}[30]}$.

– Restriction: add-difference -2^{29} in δT_{19} must not propagate past bit 31.
Conditions: $Q_{20}[18] = \overline{Q_{19}[18]}$.
– Restriction: add-difference $+2^{17}$ in δT_{22} must not propagate past bit 17. See previous section.
– Restriction: add-difference $+2^{15}$ in δT_{34} must not propagate past bit 15. See previous section.

5 Conditions on the Initial Value for the attack

The intermediate hash value used for compressing block 1 is called the initial value IV for the attack. This does not necessarily have to be the MD5 initial value, it could also result from compressing leading blocks. Although not completely obvious, the expected complexity and thus running time of the attack does depend on this initial value IV.

The intermediate value IHV resulting from the compression of block 1 is used for compressing block 2 and has the necessary conditions $IHV_2[25] = 1$ and $IHV_3[25] = 0$ for the collision in block 2 to happen. The IHV depends on the IV for the attack and $Q_{61}, \ldots, Q_{64}$ of the compression of block 1:

$$IHV_0 = IV_0 + Q_{61}$$
$$IHV_1 = IV_1 + Q_{64}$$
$$IHV_2 = IV_2 + Q_{63}$$
$$IHV_3 = IV_3 + Q_{62}$$

In [1] the sufficient conditions $Q_{62}[25] = 0$ and $Q_{63}[25] = 0$ are given. These conditions on $IHV_2[25]$ and $Q_{63}[25]$ can only be satisfied at the same time when

– either $IV_2[25] = 1$ and there is no carry from bits 0-24 to bit 25 in the addition $IV2 + Q_{63}$;
– or $IV_2[25] = 0$ and there is a carry from bits 0-24 to bit 25 in the addition $IV2 + Q_{63}$.

The conditions on $IHV_3[25]$ and $Q_{62}[25]$ can only be satisfied at the same time when

– either $IV_3[25] = 0$ and there is no carry from bits 0-24 to bit 25 in the addition $IV3 + Q_{62}$;
– or $IV_3[25] = 1$ and there is a carry from bits 0-24 to bit 25 in the addition $IV3 + Q_{62}$.

Satisfying all these conditions at the same time can even be impossible if for instance $IV_2[25-0] = 0$, or $IV_3[25] = 1 \wedge IV_3[24-0] = 0$, since the necessary carry can never happen.

Luckily this doesn't mean the attack cannot be done for those IV's, since the conditions $Q_{62}[25] = 0$ and $Q_{63}[25] = 0$ are only sufficient. They allow the most probable differential path at those steps to happen, however there are other (less probable) differential paths that are also valid. If this normally most probable differential path cannot happen or happens with low probability

(depending on the carry) then the average complexity of the attack depends on the probability that other differential paths happen. Experimentations clearly indicated that the average runtime for this situation is significantly larger than the average runtime in the situation where the most probable differential path happens with high probability.

Therefore we relaxed all conditions on bit 25 of $Q_{60}, \ldots, Q_{63}$ to allow those other differential paths to happen. We also give a recommendation for the following two IV conditions to avoid this worst case:

$$IV_2[25] = \overline{IV_2[24]} \wedge IV_3[25] = IV_3[24]$$

6 Algorithm

The algorithm used for the second block is an algorithm proposed by Klima (in section 5.8.2 of [2]). We propose a new similar algorithm for the first block where fulfilling conditions of Q_{17} no longer happens probabilistically. Both algorithms use the fact that the value of W_t ($= m_k$ for some k) can be calculated using the values of $Q_{t-3}, \ldots, Q_{t+1}$ if step t is reversed.

Algorithm 6-1 Block 1 search algorithm

Note: conditions are listed in tables A-1 and A-2.

1. *Choose $Q_1, Q_3, \ldots, Q_{16}$ fulfilling conditions;*
2. *Calculate $m_0, m_6, \ldots, m_{15}$;*
3. *Loop until $Q_{17}, \ldots, Q_{21}$ are fulfilling conditions:*
   (a) *Choose Q_{17} fulfilling conditions;*
   (b) *Calculate m_1 at $t = 16$;*
   (c) *Calculate Q_2 and m_2, m_3, m_4, m_5;*
   (d) *Calculate $Q_{18}, \ldots, Q_{21}$;*
4. *Loop over all possible Q_9, Q_{10} satisfying conditions such that m_{11} does not change:*
   (a) *Calculate $m_8, m_9, m_{10}, m_{12}, m_{13}$;*
   (b) *Calculate $Q_{22}, \ldots, Q_{64}$;*
   (c) *Verify conditions on $Q_{22}, \ldots, Q_{64}, T_{22}, T_{34}$ and the iv-conditions for the next block. Stop searching if all conditions are satisfied and a near-collision is verified.*
5. *Start again at step 1.*

Algorithm 6-2 Block 2 search algorithm

Note: conditions are listed in tables A-3 and A-4.

1. *Choose $Q_2, \ldots, Q_{16}$ fulfilling conditions;*
2. *Calculate $m_5, \ldots, m_{15}$;*
3. *Loop until $Q_{17}, \ldots, Q_{21}$ are fulfilling conditions:*
   (a) *Choose Q_1 fulfilling conditions;*
   (b) *Calculate $m_0, \ldots, m_4$;*
   (c) *Calculate $Q_{17}, \ldots, Q_{21}$;*
4. *Loop over all possible Q_9, Q_{10} satisfying conditions such that m_{11} does not change:*
   (a) *Calculate $m_8, m_9, m_{10}, m_{12}, m_{13}$;*
   (b) *Calculate $Q_{22}, \ldots, Q_{64}$;*
   (c) *Verify conditions on $Q_{22}, \ldots, Q_{64}, T_{22}, T_{34}$. Stop searching if all conditions are satisfied and a near-collision is verified.*
5. *Start again at step 1.*

To maximize the time spend in steps 4a, 4b and 4c, we can maximize the set of all possible Q_9 and Q_{10}. If we look at the relation following from step $t = 11$

$$m_{11} = RR(Q_{12} - Q_{11}, RC_{11}) - f_{11}(Q_{11}, Q_{10}, Q_9) - Q_8 - AC_{11} \quad ,$$

it is obvious that we can only change Q_9 and Q_{10} if $f_{11} = f_{11}(Q_{11}, Q_{10}, Q_9)$ does not change. Since

$$f_{11} = (Q_{11} \wedge Q_{10}) \oplus (\overline{Q_{11}} \wedge Q_9) \quad ,$$

it follows that for any b if bit $Q_{11}[b]$ is 1 then $f_{11}[b] = Q_{10}[b]$ and if it is 0 then $f_{11}[b] = Q_9[b]$. We can use this by restricting $Q_{11}[b]$ so that if there is a condition on either $Q_9[b]$ or $Q_{10}[b]$ it will always let the value of $f_{11}[b]$ depend on that condition. If for instance there is a condition $Q_9[b] = 1$ and there is no condition on $Q_{10}[b]$ then we have a free bit in $Q_{10}[b]$ if and only if $Q_{11}[b] = 0$. For block 1 we have added in this manner the conditions

$$Q_{11}[2-5, 9-11, 24-27] = 0 \quad \text{and} \quad Q_{10}[14] = Q_{11}[14] = Q_{11}[23] = 1 \quad .$$

These conditions always give the maximum number of 15 free bits total in Q_9 and Q_{10} to choose. For block 2 we have added in this manner the conditions

$$Q_{11}[5, 27-28] = 0 \quad \text{and} \quad Q_{10}[19] = Q_{11}[19] = 1 \quad .$$

These conditions also always give the maximum number of 15 free bits to choose. This allows us to try 2^{15} different messages starting at step $t = 21$ before we have to reinitialize. Most of the work will therefore be done starting at step $t = 21$. In the tables A-1 and A-3 we have listed these conditions, which are only needed to optimize above algorithms, in green.

7 Results

We have done an implementation using our proposed algorithm (see section 6) and our new Q_t conditions (listed in appendix A). We ran several tests on a 3Ghz Pentium4 using for IV either:

- the MD5 initial value
- random IV's satisfying our recommended conditions:
  $IV_2[25] = \overline{IV_2[24]} \wedge IV_3[25] = IV_3[24]$
- arbitrary random IV's

The complexity, average and maximum runtime in seconds and the total number of tests are shown in table 7-1 for each case. It clearly shows a gap between random IV's with and without the recommended IV-conditions, especially in the maximum runtime of the tests: 4 hours for arbitrary random IV's compared to 11 minutes for recommended IV's.

For recommended IV's we have found that the average number of steps of the MD5 compression function for the first block is $2^{33.6}$, due to early abortion in the compression function if a condition is not met. This is the workequivalent of $2^{27.6}$ full MD5 block compressions, since the MD5 compression function has $64 = 2^6$ steps.

At the time of this writing, the latest report[3] on MD5 collision finding using Wang's differential mentions a running time of 5 hours on a Pentium4 1.7Ghz. However Klima already mentions in [2] an average running time of over 4 hours on a Pentium 1Gz.

Our attack is a significant speedup of MD5 collision finding. If one has some freedom and can choose a recommended initial value for the attack then the average running time can even be as low as 67 seconds on a 3Ghz Pentium4. It should be noted that with a reasonable probability a collision was found in mere seconds, therefore allowing for instance to find collisions during a timeout in a protocol execution.

Table 7-1. Timing results

	Avg. time	Max. time	Avg. complexity	Nr. of tests
MD5 IV	64	428	$2^{32.25}$	1048
recommended IV's	67	660	$2^{32.33}$	1138
random IV's	291	15787	$2^{34.08}$	879

Note: Avg. complexity is the average number of MD5 block compressions.

8 Conclusion

Our presented algorithm together with new conditions we've found allows us to find full MD5 collisions in only minutes on a 3Ghz Pentium4. We have shown that the initial value for the attack can have a significant impact in the average complexity of MD5 collision finding. Using 2 conditions on the initial value to avoid very hard situations we reduce our average running time to 67 seconds. Also with reasonable probability a collision can be found in mere seconds, which allows collision finding during a protocol execution.

References

1. Philip Hawkes, Michael Paddon, and Gregory G. Rose. Musings on the Wang et al. MD5 collision. Cryptology ePrint Archive, Report 2004/264, 2004. http://eprint.iacr.org/2004/264.
2. Vlastimil Klima. Finding MD5 collisions on a notebook PC using multi-message modifications. Cryptology ePrint Archive, Report 2005/102, 2005. http://eprint.iacr.org/2005/102.
3. Jie Liang and Xuejia Lai. Improved collision attack on hash function MD5. Cryptology ePrint Archive, Report 2005/425, 2005. http://eprint.iacr.org/2005/425.
4. R.L. Rivest. The MD5 Message-Digest algorithm. Internet RFC, April 1992. RFC 1321.
5. Yu Sasaki, Yusuke Naito, Noboru Kunihiro, and Kazuo Ohta. Improved collision attack on MD5. Cryptology ePrint Archive, Report 2005/400, 2005. http://eprint.iacr.org/2005/400.
6. Xiaoyun Wang, Dengguo Feng, Xuejia Lai, and Hongbo Yu. Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD. Cryptology ePrint Archive, Report 2004/199, 2004. http://eprint.iacr.org/2004/199.
7. Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In *EUROCRYPT*, 2005. http://www.infosec.sdu.edu.cn/paper/md5-attack.pdf.
8. Jun Yajima and Takeshi Shimoyama. Wang's sufficient conditions of MD5 are not sufficient. Cryptology ePrint Archive, Report 2005/263, 2005. http://eprint.iacr.org/2005/263.

A Block conditions

Table A-1. Optimized block 1 conditions - part 1

Conditions on Q_t		
t	$Q_t[31]$ $Q_t[0]$	#
3	....0... ....0... .0......	3
4	1....... 0^^^1^^^ ^^^^1^^^ ^011....	19 + 2
5	100010. 01..0000 00000000 0010.1.1	22 + 5
6	0000001^ 01111111 10111100 0100^0^1	32
7	00000011 11111110 11111000 00100000	32
8	00000001 1..10001 0.0.0101 01000000	28
9	11111011 ...10000 0.1^1111 00111101	28
10	0111.... 0..11111 1101...0 01....00	16 + 3
11	00100000 1...0001 11000000 11000010	15 + 14
12	000...00 ....1000 0001...1 0.......	14 + 1
13	01....01 ....1111 111....0 0...1...	14
14	0.0...00 ....1011 111....1 1...1...	14
15	0.1...01 ....0 1....... ....0...	6 + 1
16	0!1..... ......!.	2 + 2
17	0!...... ......0. ^....... ....^...	4 + 1
18	0.^..... ......1.	3
19	0....... ......0.	2
20	0....... .....!..	1 + 1
21	0....... ......^.	2
	Sub-total # conditions	257 + 30

Table A-2. Optimized block 1 conditions - part 2

Conditions on Q_t		
t	$Q_t[31]$ $Q_t[0]$	#
22	0.....	1
23	0.....	1
24	1.....	1
$25-45$	....	0
46	I.....	0
47	J.....	0
48	I.....	1
49	J.....	1
50	K.....	1
51	J.....	1
52	K.....	1
53	J.....	1
54	K.....	1
55	J.....	1
56	K.....	1
57	J.....	1
58	K.....	1
59	J.....	1
60	I.....	1
61	J.....	1
62	I.....	1
63	J.....	1
64	....	0
	Sub-total # conditions	19
	Sub-total # T_t restrictions (see section 3.10)	2
	Sub-total # IV conditions from 2nd block	8
	Total # conditions	$284+30+2$ $=316$

Note: $I, J, K \in \{0, 1\}$ and $K = \overline{I}$.

Table A-3. Optimized block 2 conditions - part 1

Conditions on Q_t		
t	$Q_t[31]$ $Q_t[0]$	#
−2	......0.	(1)
−1	^....01.	(3)
0	^....00. ....0.....	(4)
	Total # IV conditions for 1st block	(8)
1	!...010. ..1....0 ....0... .10.....	8 + 1
2	^^^^110. ..0^^^^0 1..^1... ^10..00.	20 + 1
3	^011111. ..011111 0..01..1 011^^11.	23 + 1
4	^011101. ..000100 ...00^^0 0001000^	26
5	!10010.. ..101111 ...01110 01010000	25
6	^..0010. 1.10..10 11.01100 01010110	24 + 1
7	!..1011^ 1.00..01 10.11110 00.....1	20 + 1
8	^..00100 0.11..10 1.....11 111...^0	18 + 1
9	^..11100 0.....01 0..^..01 110...01	17 + 1
10	^....111 1...1011 11001.11 11....00	18 + 2
11	^..00... ....1101 11000.11 110...11	15 + 4
12	^^^00^^^ ....1000 0001.... 1.......	17
13	!0111111 0...1111 111..... 0...1...	17 + 1
14	^1000000 1...1011 111..... 1...1...	17 + 1
15	01111101 .... 00...... ....0...	10 + 1
16	0.10.... ......!.	2 + 2
17	0!...... ......0. ^....... ....^...	4 + 1
18	0.^..... ......1.	3
19	0....... ......0.	2
20	0....... .....!..	1 + 1
21	0....... ......^.	2
	Sub-total # conditions	289 + 20

Table A-4. Optimized block 2 conditions - part 2

Conditions on Q_t		
t	$Q_t[31]$ $Q_t[0]$	#
22	0.....	1
23	0.....	1
24	1.....	1
$25-45$	....	0
46	I.....	0
47	J.....	0
48	I.....	1
49	J.....	1
50	K.....	1
51	J.....	1
52	K.....	1
53	J.....	1
54	K.....	1
55	J.....	1
56	K.....	1
57	J.....	1
58	K.....	1
59	J.....	1
60	I.....	1
61	J.....	1
62	I.....	1
63	J.....	1
64	....	0
	Sub-total # conditions	19
	Sub-total # T_t restrictions (see section 3.10)	2
	Total # conditions	$308+20+2$ $=330$

Note: $I, J, K \in \{0,1\}$ and $K = \overline{I}$.

B Source Code

The source code will be made available at:

http://www.win.tue.nl/hashclash/