# Finding MD5 Collisions on a Notebook PC Using Multi-message Modifications

**Vlastimil Klima**[1, 2]

v.klima@volny.cz

http://cryptography.hyperlink.cz/
Prague, Czech Republic

March 31, 2005
version 1

## Abstract

In this paper, we summarize the results achieved during our brief three months long research on collisions of the MD5 hash function. Being inspired by the results announced by Wang et al. [1] we independently developed methods for finding collisions which work for any initialization value and which are quicker than the methods presented in [1, 8]. It enables us to find a MD5 collision on a standard notebook PC roughly in 8 hours [7]. Independently on [1, 8], we discovered and propose several multi-message modification methods, which are more effective than methods described in [1, 8]. We show their principle.

**Keywords:** MD5, collisions, multi-message modification.

## 1 Introduction

One of the major cryptographic "break-through" of the recent years was a discovery of collisions for a set of hash functions (MD4, MD5, HAVAL-128, RIPEMD) by the Chinese cryptographers in August 2004 [1]. Their authors (Wang et al.) kept the algorithm secret, however. During October 2004, the Australian team (Hawkes et al.) tried to reconstruct the methodology in their great work [3]. The most important "Chinese trick" was not discovered, although they succeeded in describing a differential scheme of conditions that hold for the published collisions. Nevertheless, fulfilling the conditions of this scheme was still considerably computationally difficult in comparison to what the results of [1] showed.

## 2 Results of our Research [7]

During our research [7], we also analyzed the available data using differential cryptanalysis. We found a way to generate the first message block of the collision about 1000 - 2000 times faster than the Chinese team - that corresponds to reaching the first colliding block in 2 minutes using a notebook PC. The same computation phase took the Chinese team about an hour using an IBM P690 supercomputer. On the other hand, the Chinese team was 2 - 80 times faster when computing the second message block of their collisions. Therefore, our and the Chinese methods probably differs in several details in both parts of the computation. Overall, our method is about 3 - 6 times faster. More specifically, finding the first (complete) collision took 8 hours using a notebook PC (Intel Pentium 1.6 GHz). Note that our method works for any initialization vector. It can be abused in forging signatures of software packages and digital certificates as some papers show ([4], [5], [6]). We showed that it is possible to find MD5 collisions using an ordinary home PC. That should be a warning towards persisting usage of MD5. In the appendix, we show new examples of collisions for a standard and chosen initialization values.

---

[1] This research was done during Christmas vacation and during January - March 2005. At that time, the author was working for the company LEC, s.r.o., Prague, Czech Republic which supported the project by material and financial means.

[2] This paper will be presented at 3rd International Scientific Conference *Security and Protection of Information*, Brno, Czech Republic, May 3 - 5, 2005, http://www.unob.cz/spi/defaulten.asp.

# 3 Results of Wang et al. [1]

Hash functions are very useful cryptographic tool. To be one-way and collision-free, the hash functions have to be very robust and complex. Therefore, it is always exciting when a collision is found. One of the most important cryptanalytic articles of the last year was precisely the work of the Chinese team [1]. MD5 was the most challenging hash function, so we will further focus on this function only.

Let us remind that there was no algorithm nor explanation in [1] as to how to find the collisions, only some brief data was provided which we shall recall now: The colliding pair of messages (M, N) and (M', N') consists of two message blocks. The first blocks differ only in a predefined constant vector C1 (M' = M + C1) and the second blocks also differ only in a predefined constant vector C2 = -C1 mod $2^{32}$ (N' = N + C2) whereas MD5(M, N) = MD5(M', N').

Wang et al. stated that it takes about an hour to find the block M using their supercomputer IBM P690. Finding N then takes only 15 seconds to 5 minutes. In the first version of [1], two pairs of the colliding messages were presented. The initialization vector (IV) value the authors chose, however, was not the one used in MD5 algorithm, since the order of bytes was reversed (little-endian vs. big-endian). In the corrected version of the paper, Wang et al. showed two pairs of colliding messages for MD5, with the right IV that time. They made a remark that their attack works for any initialization value IV.

## 3.1 Collision Abuse

After having published their results, we had only four pairs of the colliding messages. Nevertheless, it was shown that even these data could be used to mount a successful attack [4], [5]. It is shown in [4] that a single collision is enough to create a pair of different self-extracting archives with an identical hash value. It can be abused, for example, to put backdoors into large software packages during their distribution. Furthermore, it was shown in cooperation with one of the authors of [1], how to forge a digital certificate using the ability of making a collision for any initialization vector, too [6].

# 4 An Attempt to Unveil the Chinese Trick

A great work by Hawkes et al. [3] was published in October 2004. The authors tried to unveil the "Chinese method" basing on the raw data provided in [1]. In this work, they inspect inner differences and conditions for messages to hold in order to create a collision using the Chinese method. It was the first rigorous analysis and attempt to explain the Chinese method. Basing on one pair of colliding messages (with the correct IV), the authors described the differential scheme that the published collision (the one with correct IV) satisfies. This scheme was probably in the background of the collision design. However, they did not manage to explain how this schema was created. Furthermore, they described what conditions must hold for one message of the colliding pair so that the differential scheme is fulfilled. A long list of conditions was acquired. The first set of conditions (so called ft- and Tt-conditions) comes up by the first block passing the 64 steps of MD5.

If the conditions are met in the first 16 steps (more than 200 conditions) by selecting an appropriate M block, 39 ft-conditions and "3.2" Tt-conditions are left to be fulfilled in the remaining steps. These conditions are met only on probabilistic basis. To sum it up, we need to generate about $2^{42.2}$ messages M to find the one, such that it meets all the ft- and Tt- conditions from steps 17 - 64. Similarly, to fulfill the ft- and Tt- conditions for the second block N, it is necessary to generate $2^{42.2}$ messages. The overall complexity is then $2^{43}$. Hawkes et al. judged that the complexity was too large for the collision to be computed in one hour. Basing on that observation, they concluded that Wang et al. must have used another trick. Obviously, this trick is the key trick.

# 5 Our Method

In our research [7], we started by the results of [3] and observed the differential scheme from the point of additive differences (arithmetic difference modulo $2^{32}$) and binary differences (xor, modulo 2), in the same way as in [3]. Furthermore, we examined other colliding pair, which was the one with wrong IV. We verified that the differential scheme holds for both colliding pairs, however there was no more data available. In our research, it emerged that some of the ft- and Tt- conditions could have been met in different ways than those Hawkes et al. chose. That could theoretically decrease the computational complexity. However, it would lead to increasing the complexity of the collision-finding program and its memory demands. Therefore, we did not go this way. Yet, the

analysis of ft- and Tt- conditions has shown that the real complexity of collisions finding could be smaller than the one in the theoretical model. As the research went further, we found our multi-message modification method to generate the first blocks of colliding messages very quickly (independently on [8], which was published later). Using a standard PC notebook it took 2 minutes to find the first block of the message whereas it took one hour using the supercomputer in [1]. Due to the briefness of research we did not go further in speeding up the search for second blocks as we did for the first one, even though we reached the complexity significantly lower than $2^{42}$ according to [3] and lower than $2^{32}$ according to [8]. The fact that we are able to find the collision in 8 hours using the PC notebook attests that. According to [1], the search for the second block should be 12 - 240 times faster than searching for the first block. That would yield a collision in 2 minutes instead of 8 hours on a notebook. So there are reserves in finding second block of the collision.

## 5.1 Results of Our Experiments

We did not use any supercomputer to find the collisions, just ordinary desktop computers. The author conducted his experiments on his notebook where he found tens of thousands of collisions for the first block and subsequently complete MD5 collisions for both original IV and chosen IVs. To test the program functionality, the author asked a few friends to try it on their own computers. In this way, during a week of experimentation, tens of thousands first-block collisions and a few dozens of full collisions were found.

The results obtained using an ordinary notebook (Acer Travelmate 450 LMi, Intel Pentium 1.6 GHz) are as follows: During 8 hours, 331 collisions of first block were found and one complete collision was reached. According to the fact that it took the Chinese team one hour to find the first block collision, searching for 331 of these collisions would take 331 hours, which is 40 times more. It is hard to compare the power of a notebook and a supercomputer due to different architectures, but if we take to account that the IBM P690 is about 25 - 50 times faster than the notebook (estimate provided by Ondrej Mikle based on simple bogomips ratio) we get the result that our method of searching for first-block collision is 1000 - 2000 times faster than the one in [1]. On the other hand, the searching for second-block collision is 2 - 80 times slower. Overall, if we compare the time to find a complete collision by the Chinese team (1 - 1.08 hour) with us (8 hours) on 25 - 50 times slower machine, our method is 3 - 6 times faster. All these comparisons are for illustrational purposes only and the author makes no claim about their accuracy (except for the time values).

## 5.2 Principle of the Chinese Trick

Results of our research [7] were published before the papers [8] and [9] were placed on the web. The "trick" of finding collisions was described in [8], but without details, enabling to program and reconstruct it. We can recall the key ideas from [8] here and explain the differences between our and the Chinese method.

## 5.3 The Choice of a Difference Between Message Blocks

The starting point of [8] was a decision to find collisions for two-block messages and a choice of the particular difference between message words. According to [8], the differences were chosen to increase the probability of the differential in rounds 3 and 4 of MD5. Let us note that there are also other such basic differences leading to effective differential schemes.

## 5.4 Variables Q ($Q^*$), Stationary Conditions and the Differential Scheme

Let us denote M and $M^*$ the first blocks of colliding messages and let Q and $Q^*$ be the corresponding variables according to [3]. Q[1] (resp. $Q^*$[1] ) denotes the first value which is calculated in the first step of processing the block M (resp. M*) in MD5. Q[64] (resp. $Q^*$[64]) is the last such value. Denote x[i], i = 0,..., 15 the particular words of the block M. According to [3], we get the following relationships between Q and x. At the end of the equations for Q[17 - 20] bellow, we note the number of conditions to be fulfilled according to the differential scheme of [3]. By RL and RR, we denote a cyclic rotation to the left and to the right, respectively.

```
Q[ 1]=Q[0 ]+RL(F(Q[0 ],Q[-1],Q[-2])+Q[-3]+x[0 ]+0xd76aa478, 7);
Q[ 2]=Q[1 ]+RL(F(Q[1 ],Q[0 ],Q[-1])+Q[-2]+x[1 ]+0xe8c7b756,12);
Q[ 3]=Q[2 ]+RL(F(Q[2 ],Q[1 ],Q[0 ])+Q[-1]+x[2 ]+0x242070db,17);
```

```
Q[ 4]=Q[3 ]+RL(F(Q[3 ],Q[2 ],Q[1 ])+Q[0 ]+x[3 ]+0xc1bdceee,22);
Q[ 5]=Q[4 ]+RL(F(Q[4 ],Q[3 ],Q[2 ])+Q[1 ]+x[4 ]+0xf57c0faf, 7);
Q[ 6]=Q[5 ]+RL(F(Q[5 ],Q[4 ],Q[3 ])+Q[2 ]+x[5 ]+0x4787c62a,12);
Q[ 7]=Q[6 ]+RL(F(Q[6 ],Q[5 ],Q[4 ])+Q[3 ]+x[6 ]+0xa8304613,17);
Q[ 8]=Q[7 ]+RL(F(Q[7 ],Q[6 ],Q[5 ])+Q[4 ]+x[7 ]+0xfd469501,22);
Q[ 9]=Q[8 ]+RL(F(Q[8 ],Q[7 ],Q[6 ])+Q[5 ]+x[8 ]+0x698098d8, 7);
Q[10]=Q[9 ]+RL(F(Q[9 ],Q[8 ],Q[7 ])+Q[6 ]+x[9 ]+0x8b44f7af,12);
Q[11]=Q[10]+RL(F(Q[10],Q[9 ],Q[8 ])+Q[7 ]+x[10]+0xffff5bb1,17);
Q[12]=Q[11]+RL(F(Q[11],Q[10],Q[9 ])+Q[8 ]+x[11]+0x895cd7be,22);
Q[13]=Q[12]+RL(F(Q[12],Q[11],Q[10])+Q[9 ]+x[12]+0x6b901122, 7);
Q[14]=Q[13]+RL(F(Q[13],Q[12],Q[11])+Q[10]+x[13]+0xfd987193,12);
Q[15]=Q[14]+RL(F(Q[14],Q[13],Q[12])+Q[11]+x[14]+0xa679438e,17);
Q[16]=Q[15]+RL(F(Q[15],Q[14],Q[13])+Q[12]+x[15]+0x49b40821,22);

Q[17]=Q[16]+RL(G(Q[16],Q[15],Q[14])+Q[13]+x[1 ]+0xf61e2562, 5); 4 conds
Q[18]=Q[17]+RL(G(Q[17],Q[16],Q[15])+Q[14]+x[6 ]+0xc040b340, 9); 3 conds
Q[19]=Q[18]+RL(G(Q[18],Q[17],Q[16])+Q[15]+x[11]+0x265e5a51,14); 2 conds
Q[20]=Q[19]+RL(G(Q[19],Q[18],Q[17])+Q[16]+x[0 ]+0xe9b6c7aa,20); 1 cond
```

Let us consider first the differential scheme in the first block. Both papers [3] and [8] present similar differential scheme from which a lot of conditions for bits of variables Q[1 - 64] follow, together with four further intermediate values (between the first and the second block). Some bits of Q are set to the value of one or zero; some have to fulfill the relation of equality or negation. We call these conditions stationary conditions. Their are chosen in [3] and [8] to be the sufficient conditions for reaching the differential scheme in steps 1 - 64. The entire message block pairs M and $M^*$, where M fulfills the stationary conditions and $M^*$ differs from M in a defined way in x[1] (by $2^{31}$), x[4] (by $2^{15}$) and x[11] (by $2^{31}$), will fulfill the whole differential scheme.

We note that there are other ways on how to fulfill the differential scheme, therefore the (slightly different) schemes from [3] and [8] are just only two examples of possible solutions. Actually, there are a lot of differential schemes.

## 5.5 Fulfilling the Stationary Conditions

Our procedure is general and independent on a particular differential scheme chosen. In our method, we choose variables Q[1 - 16] arbitrarily, but satisfying the stationary conditions. From them, we derive x[0 - 15] and check if the other stationary conditions for Q[17 - 64] are fulfilled (we have 36 conditions and 7 more conditions for four sums between the first and the second block). These conditions are fulfilled with a certain probability only. According to [8], the complexity of this procedure is $2^{43}$.

Our goal is, therefore, to find such values x[0 - 15] that conditions Q[1 - 16] and as many as possible of conditions for Q[17], Q[18], ... are fulfilled deterministically.

In [8], they state that it is possible to fulfill 6 stationary conditions in Q[17], Q[18], .... by the multi-message modifications in the first block. The collision complexity then decreases from $2^{43}$ to $2^{37}$ in the first block.

Our method [7], developed independently on [8], decreases the complexity to $2^{33}$, because it fulfills 10 conditions in steps Q[17 - 20] deterministically.

## 5.6 Our Multi-message Modification Method in the First Block

In our multi-message modification method, we fulfill conditions Q[1 - 20] deterministically. The remaining 31 conditions are then fulfilled in a probabilistic manner.

Procedure:

1. We choose Q[3 - 16] fulfilling stationary conditions.
2. We compute x[6 - 15] :

```
x[ 6]=RR(Q[7 ]-Q[6 ],17)-F(Q[6 ],Q[5 ],Q[4 ])-Q[3 ]-0xa8304613;
x[ 7]=RR(Q[8 ]-Q[7 ],22)-F(Q[7 ],Q[6 ],Q[5 ])-Q[4 ]-0xfd469501;
x[ 8]=RR(Q[9 ]-Q[8 ], 7)-F(Q[8 ],Q[7 ],Q[6 ])-Q[5 ]-0x698098d8;
x[ 9]=RR(Q[10]-Q[9 ],12)-F(Q[9 ],Q[8 ],Q[7 ])-Q[6 ]-0x8b44f7af;
x[10]=RR(Q[11]-Q[10],17)-F(Q[10],Q[9 ],Q[8 ])-Q[7 ]-0xffff5bb1;
x[11]=RR(Q[12]-Q[11],22)-F(Q[11],Q[10],Q[9 ])-Q[8 ]-0x895cd7be;
x[12]=RR(Q[13]-Q[12], 7)-F(Q[12],Q[11],Q[10])-Q[9 ]-0x6b901122;
x[13]=RR(Q[14]-Q[13],12)-F(Q[13],Q[12],Q[11])-Q[10]-0xfd987193;
x[14]=RR(Q[15]-Q[14],17)-F(Q[14],Q[13],Q[12])-Q[11]-0xa679438e;
x[15]=RR(Q[16]-Q[15],22)-F(Q[15],Q[14],Q[13])-Q[12]-0x49b40821;
```

3. We change Q[17] until conditions Q[17 - 19] are fulfilled. When necessary, it is also possible to change Q[3 - 16].

```
Q[18]=Q[17]+RL(G(Q[17],Q[16],Q[15])+Q[14]+x[6 ]+0xc040b340, 9),
Q[19]=Q[18]+RL(G(Q[18],Q[17],Q[16])+Q[15]+x[11]+0x265e5a51,14).
```

Then we compute

```
x[1]=RR(Q[17]-Q[16], 5)-G(Q[16],Q[15],Q[14])-Q[13]-0xf61e2562;
```

and

```
x[2]=RR(Q[3]-Q[2],17)-F(Q[2],Q[1],Q[0])-Q[-1]-0x242070db;
x[3]=RR(Q[4]-Q[3],22)-F(Q[3],Q[2],Q[1])-Q[0]-0xc1bdceee;
x[4]=RR(Q[5]-Q[4], 7)-F(Q[4],Q[3],Q[2])-Q[1]-0xf57c0faf;
x[5]=RR(Q[6]-Q[5],12)-F(Q[5],Q[4],Q[3])-Q[2]-0x4787c62a;
```

4. All stationary conditions Q[3 - 19] are fulfilled now. Moreover, we have free value x[0].
5. We choose Q[20] arbitrarily, but fulfilling the one stationary conditions for it. Then we compute x[0]:

```
x[0]=RR(Q[20]-Q[19],20)-G(Q[19],Q[18],Q[17])-Q[16]-0xe9b6c7aa;
```

6. The remaining 33 conditions are fulfilled probabilistically. We have $2^{31}$ possible values for Q[20]. If we don't find a collision, we return to the point 3 and continue by choosing a new value of Q[17]. With the complexity of $2^{33}$, we find a collision of the first block (including four conditions for four sums between the first and second block).

**Note.** If we would describe the method as in [8], it would not be very transparent. In our way, we can see the goal of multi-message modifications obviously. Anyway, there are two multi-message modifications. The first modification consists of changing x[1], x[6], and x[11] simultaneously to fulfill conditions Q[17 - 19]. Furthermore, we modify x[0] to fulfill the condition for Q[20].

## 5.7 Our Multi-Message Modification Method in the Second Block

Now, we present a method which is not optimal, but which was used in [7]. We will present a multi-message modification methods then which are quicker, but which were not experimentally verified.

Our multi-message modification method consists of deterministically fulfilling conditions for Q[1 - 18]. The remaining 29 conditions are fulfilled in a probabilistic manner.

Procedure:

1. We choose Q[3 - 16] fulfilling stationary conditions.
2. We change arbitrarily the bit number 22 (weight $2^{22}$) of variables Q[4 - 16].

3. We choose the value Q[1] arbitrarily, but fulfilling stationary conditions (we have here at minimum $2^{24}$ possible values of Q[1])
4. We compute x[0], x[7 - 15] :

```
x[0]=RR(Q[1]-Q[0],7)-F(Q[0],Q[-1],Q[-2])-Q[-3]-0xd76aa478;

x[7]= RR(Q[8 ]-Q[7 ],22)-F(Q[7 ],Q[6 ],Q[5 ])-Q[4 ]-0xfd469501;
x[8]= RR(Q[9 ]-Q[8 ], 7)-F(Q[8 ],Q[7 ],Q[6 ])-Q[5 ]-0x698098d8;
x[9]= RR(Q[10]-Q[9 ],12)-F(Q[9 ],Q[8 ],Q[7 ])-Q[6 ]-0x8b44f7af;
x[10]=RR(Q[11]-Q[10],17)-F(Q[10],Q[9 ],Q[8 ])-Q[7 ]-0xffff5bb1;
x[11]=RR(Q[12]-Q[11],22)-F(Q[11],Q[10],Q[9 ])-Q[8 ]-0x895cd7be;
x[12]=RR(Q[13]-Q[12], 7)-F(Q[12],Q[11],Q[10])-Q[9 ]-0x6b901122;
x[13]=RR(Q[14]-Q[13],12)-F(Q[13],Q[12],Q[11])-Q[10]-0xfd987193;
x[14]=RR(Q[15]-Q[14],17)-F(Q[14],Q[13],Q[12])-Q[11]-0xa679438e;
x[15]=RR(Q[16]-Q[15],22)-F(Q[15],Q[14],Q[13])-Q[12]-0x49b40821;
```

5. We choose Q[2] arbitrarily, but fulfilling stationary conditions (we have here at minimum $2^{12}$ possible values).
6. We compute x[1] from Q[1 - 2] and check if the four stationary conditions for Q[17] are fulfilled:

```
x[1]=RR(Q[2]-Q[1],12)-F(Q[1],Q[0],Q[-1])-Q[-2]-0xe8c7b756,
Q[17]=Q[16]+RL(G(Q[16],Q[15],Q[14])+Q[13]+x[1]+0xf61e2562, 5).
```

If they are not, we go to step 5 and choose a new Q[2] (if necessary, it is possible to go to step 3 and choose a new value for Q[1]).

7. We set two lower bits of Q[3] as in Q[2] according to appropriate stationary conditions.
8. From Q[3 - 7] we compute x[2 - 6]

```
x[2]=RR(Q[3]-Q[2],17)-F(Q[2],Q[1],Q[0])-Q[-1]-0x242070db;
x[3]=RR(Q[4]-Q[3],22)-F(Q[3],Q[2],Q[1])-Q[0 ]-0xc1bdceee;
x[4]=RR(Q[5]-Q[4], 7)-F(Q[4],Q[3],Q[2])-Q[1 ]-0xf57c0faf;
x[5]=RR(Q[6]-Q[5],12)-F(Q[5],Q[4],Q[3])-Q[2 ]-0x4787c62a;
x[6]=RR(Q[7]-Q[6],17)-F(Q[6],Q[5],Q[4])-Q[3 ]-0xa8304613;
```

and we check if the conditions for Q[18] are fulfilled. If they are not, we go back in step 5 to choose a new value for Q[2] (resp. Q[1] in step 3). According to the terminology of [8], we use the multi-message modifications of x[2 - 6].

9. Now, all the conditions for Q[1 - 18] are fulfilled.
10. The remaining 29 conditions (according to [8]), resp. 27 conditions (according to [3]) are fulfilled probabilistically. If we do not find a collision, we have more than $2^{29}$ possibilities of choices of Q[1 - 2], eventually with further choices of $22^{nd}$ bits of Q[3 - 16]. With the complexity of $2^{29}$ ($2^{27}$) we find a collision of the second block. We note that, according to [8], the complexity of finding the collision of the second block is $2^{30}$.

## 5.8 Another Multi-Message Modification Methods in the Second Block

### 5.8.1 Fulfilling Stationary Conditions Q[1 - 19] Deterministically

We propose another procedure, consisting of fulfilling conditions Q[1 - 19] deterministically.

1. We choose Q[1 - 16] arbitrarily, but fulfilling stationary conditions.
2. We compute x[0 - 15].

3. We change Q[2] until the conditions for Q[17] are fulfilled (we compute x[1 - 5] and we check if x[1] corresponds with Q[17]).
4. We change Q[7] until conditions for Q[18] are fulfilled (we compute x[6 - 10] and we check if x[6] corresponds with Q[18]).
5. We change Q[12] until conditions for Q[19] are fulfilled (we compute x[11 - 15] and we check if x[11] corresponds with Q[19]).

Now, all conditions for Q[1 - 19] are fulfilled deterministically. The remaining 27 conditions according to [8] (resp. 25 conditions according to [3]) are fulfilled probabilistically. We have more than $2^{27}$ choices of Q[1 -16]. With the complexity of $2^{27}$ we find a collision of the second block. This is a further reduction of the complexity. This procedure was not experimentally verified.

### 5.8.2 Fulfilling Stationary Conditions Q[1 - 21] Deterministically

We propose another procedure, consisting of fulfilling conditions Q[1 - 21] deterministically.

1. We choose Q[2 - 16] arbitrarily, but fulfilling stationary conditions.
2. We compute x[5 -15].
3. We change Q[1] until the conditions for Q[17 - 21] are fulfilled (From Q[1] we compute new x[0 - 4]).
4. Now we change Q[8 - 12] (fulfilling stationary conditions) in such a way that x[11] remains unchanged. Because the values x[1, 6, 11, 0, 5] remain unchanged, the conditions Q[17 - 21] remain fulfilled. We compute x[7 -15]. The remaining 24 stationary conditions are fulfilled probabilistically.

We have more than $2^{24}$ choices of Q[8 -12]. With the complexity of $2^{24}$ we find a collision of the second block. This is a further reduction of the complexity.

This procedure was not experimentally verified.

# 6 Conclusion

We proposed new methods for MD5 collisions finding, independently on a description of the method by Wang et al. [1, 8]. Our methods are quicker. We reached complexity $2^{33}$ vs. $2^{37}$ in the first block and $2^{29}$ (or even $2^{24}$) vs. $2^{30}$ in the second block. It enables us to find collisions on a standard notebook PC in several hours. Furthermore, they work for arbitrarily chosen initialization value.

**Acknowledgements**

I would like to thank my friends for their help. To Milan Nosál (LEC, s.r.o) for his help with debugging the program, to Tomáš Rosa, Ondřej Pokorný, and Milan Nosál for conducting experiments on their home computers, to Tomáš Jabůrek for technical help with experiments, to Ondrej Mikle and Tomáš Rosa for helping me with the translation of the paper and to all of them for valuable comments.

**Note**

In the last experiment, provided by Ondřej Pokorný on his home PC (Intel Pentium, 1GHz), he obtained 14 collisions in 58 hours and 32 minutes. It gives even more optimistic time for finding a collision (1 collision per 4 hours 11 minutes) than on the author's notebook.

**Homepage of the project**

http://cryptography.hyperlink.cz/MD5_collisions.html

# 7 Appendix: Examples

## 7.1 Example: MD5 Collision with the Standard IV

IV according to [2]:

```
  context->state[0] = 0x67452301;
  context->state[1] = 0xefcdab89;
  context->state[2] = 0x98badcfe;
  context->state[3] = 0x10325476;
```

**First message:**

```
0xA6,0x64,0xEA,0xB8,0x89,0x04,0xC2,0xAC,
0x48,0x43,0x41,0x0E,0x0A,0x63,0x42,0x54,
0x16,0x60,0x6C,0x81,0x44,0x2D,0xD6,0x8D,
0x40,0x04,0x58,0x3E,0xB8,0xFB,0x7F,0x89,
0x55,0xAD,0x34,0x06,0x09,0xF4,0xB3,0x02,
0x83,0xE4,0x88,0x83,0x25,0x71,0x41,0x5A,
0x08,0x51,0x25,0xE8,0xF7,0xCD,0xC9,0x9F,
0xD9,0x1D,0xBD,0xF2,0x80,0x37,0x3C,0x5B,
0x97,0x9E,0xBD,0xB4,0x0E,0x2A,0x6E,0x17,
0xA6,0x23,0x57,0x24,0xD1,0xDF,0x41,0xB4,
0x46,0x73,0xF9,0x96,0xF1,0x62,0x4A,0xDD,
0x10,0x29,0x31,0x67,0xD0,0x09,0xB1,0x8F,
0x75,0xA7,0x7F,0x79,0x30,0xD9,0x5C,0xEB,
0x02,0xE8,0xAD,0xBA,0x7A,0xC8,0x55,0x5C,
0xED,0x74,0xCA,0xDD,0x5F,0xC9,0x93,0x6D,
0xB1,0x9B,0x4A,0xD8,0x35,0xCC,0x67,0xE3.
```

**Second message:**

```
0xA6,0x64,0xEA,0xB8,0x89,0x04,0xC2,0xAC,
0x48,0x43,0x41,0x0E,0x0A,0x63,0x42,0x54,
0x16,0x60,0x6C,0x01,0x44,0x2D,0xD6,0x8D,
0x40,0x04,0x58,0x3E,0xB8,0xFB,0x7F,0x89,
0x55,0xAD,0x34,0x06,0x09,0xF4,0xB3,0x02,
0x83,0xE4,0x88,0x83,0x25,0xF1,0x41,0x5A,
0x08,0x51,0x25,0xE8,0xF7,0xCD,0xC9,0x9F,
0xD9,0x1D,0xBD,0x72,0x80,0x37,0x3C,0x5B,
0x97,0x9E,0xBD,0xB4,0x0E,0x2A,0x6E,0x17,
0xA6,0x23,0x57,0x24,0xD1,0xDF,0x41,0xB4,
0x46,0x73,0xF9,0x16,0xF1,0x62,0x4A,0xDD,
0x10,0x29,0x31,0x67,0xD0,0x09,0xB1,0x8F,
```

```
0x75,0xA7,0x7F,0x79,0x30,0xD9,0x5C,0xEB,
0x02,0xE8,0xAD,0xBA,0x7A,0x48,0x55,0x5C,
0xED,0x74,0xCA,0xDD,0x5F,0xC9,0x93,0x6D,
0xB1,0x9B,0x4A,0x58,0x35,0xCC,0x67,0xE3.
```

**Common MD5 hash:**

```
0x2B,0xA3,0xBE,0x5A,0xA5,0x41,0x00,0x6B,
0x62,0x37,0x01,0x11,0x28,0x2D,0x19,0xF5.
```

## 7.2 Example: MD5 Collision with a Chosen IV

```
context->state[0] = 0xabaaaaaa;
context->state[1] = 0xaaacaaaa;
context->state[2] = 0xaaaadaaa;
context->state[3] = 0xaaaaaaea;
```

**First message:**

```
0x9E,0x83,0x2A,0x4C,0x95,0x64,0x5E,0x2B,
0x2E,0x1B,0xB0,0x70,0x47,0x1E,0xBA,0x13,
0x7F,0x1A,0x53,0x43,0x22,0x34,0x25,0xC1,
0x40,0x04,0x58,0x3E,0xB8,0xFB,0x7F,0x89,
0x55,0xAD,0x34,0x06,0x09,0xF4,0xB3,0x02,
0x83,0xE4,0x88,0x83,0x25,0x71,0x41,0x5A,
0x08,0x51,0x25,0xE8,0xF7,0xCD,0xC9,0x9F,
0xD9,0x1D,0xBD,0xF2,0x80,0x37,0x3C,0x5B,
0x89,0x62,0x33,0xEC,0x5B,0x0C,0x8D,0x77,
0x19,0xDE,0x93,0xFA,0xA1,0x44,0xA8,0xCC,
0x56,0x91,0x9E,0x47,0x00,0x0C,0x00,0x4D,
0x40,0x29,0xF1,0x66,0xD1,0x09,0xB1,0x8F,
0x75,0x27,0x7F,0x79,0x30,0xD5,0x5C,0xEB,
0x42,0xE8,0xAD,0xBA,0x78,0xCC,0x55,0x5C,
0xED,0xF4,0xCA,0xDD,0x5F,0xC5,0x93,0x6D,
0xD1,0x9B,0x0A,0xD8,0x35,0xCC,0xE7,0xE3.
```

**Second message:**

```
0x9E,0x83,0x2A,0x4C,0x95,0x64,0x5E,0x2B,
0x2E,0x1B,0xB0,0x70,0x47,0x1E,0xBA,0x13,
0x7F,0x1A,0x53,0xC3,0x22,0x34,0x25,0xC1,
0x40,0x04,0x58,0x3E,0xB8,0xFB,0x7F,0x89,
0x55,0xAD,0x34,0x06,0x09,0xF4,0xB3,0x02,
0x83,0xE4,0x88,0x83,0x25,0xF1,0x41,0x5A,
0x08,0x51,0x25,0xE8,0xF7,0xCD,0xC9,0x9F,
0xD9,0x1D,0xBD,0x72,0x80,0x37,0x3C,0x5B,
```

```
0x89,0x62,0x33,0xEC,0x5B,0x0C,0x8D,0x77,
0x19,0xDE,0x93,0xFA,0xA1,0x44,0xA8,0xCC,
0x56,0x91,0x9E,0xC7,0x00,0x0C,0x00,0x4D,
0x40,0x29,0xF1,0x66,0xD1,0x09,0xB1,0x8F,
0x75,0x27,0x7F,0x79,0x30,0xD5,0x5C,0xEB,
0x42,0xE8,0xAD,0xBA,0x78,0x4C,0x55,0x5C,
0xED,0xF4,0xCA,0xDD,0x5F,0xC5,0x93,0x6D,
0xD1,0x9B,0x0A,0x58,0x35,0xCC,0xE7,0xE3.
```

**Common hash:**

```
0xef,0x2e,0xae,0x54,0xe0,0x34,0x70,0x7c,
0xa2,0x6e,0xb0,0x9b,0x45,0xc7,0xe4,0x87.
```

# References


[1] Xiaoyun Wang, Dengguo Feng, Xuejia Lai, Hongbo Yu: Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD, rump session, CRYPTO 2004, *Cryptology ePrint Archive*, Report 2004/199, first version (August 16, 2004), second version (August 17, 2004), http://eprint.iacr.org/2004/199.pdf

[2] Ronald Rivest: The MD5 Message Digest Algorithm, RFC1321, April 1992, ftp://ftp.rfc-editor.org/in-notes/rfc1321.txt

[3] Philip Hawkes, Michael Paddon, Gregory G. Rose: Musings on the Wang et al. MD5 Collision, *Cryptology ePrint Archive*, Report 2004/264, 13 October 2004, http://eprint.iacr.org/2004/264.pdf

[4] Ondrej Mikle: Practical Attacks on Digital Signatures Using MD5 Message Digest, *Cryptology ePrint Archive*, Report 2004/356, http://eprint.iacr.org/2004/356.pdf, 2nd December 2004

[5] Dan Kaminsky: MD5 To Be Considered Harmful Someday, *Cryptology ePrint Archive*, Report 2004/357, http://eprint.iacr.org/2004/357.pdf, 6 December 2004

[6] Arjen Lenstra, Xiaoyun Wang and Benne de Weger: Colliding X.509 Certificates, *Cryptology ePrint Archive*, Report 2005/067, http://eprint.iacr.org/2005/067.pdf

[7] Vlastimil Klima: Finding MD5 Collisions – a Toy For a Notebook, Cryptology ePrint Archive, Report 2005/075, http://eprint.iacr.org/2005/075.pdf, March 5, 2005

[8] Xiaoyun Wang, Hongbo Yu: How to Break MD5 and Other Hash Functions, http://www.infosec.sdu.edu.cn/paper/md5-attack.pdf, published on the web on March 6, 2005

[9] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, Xiuyuan Yu: Cryptanalysis of the Hash Functions MD4 and RIPEMD, http://www.infosec.sdu.edu.cn/paper/md4-ripemd-attck.pdf, published on the web on March 6, 2005

[10] Xiaoyun Wang, Yiqun Lisa Yin, Hongbo Yu: Collision Search Attacks on SHA1, http://theory.csail.mit.edu/~yiqun/shanote.pdf, February 2005