# Improved Collision Attack on the Hash Function Proposed at PKC'98[⋆]

Florian Mendel[⋆⋆], Norbert Pramstaller, and Christian Rechberger

Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology, Inffeldgasse 16a, A-8010 Graz, Austria

Florian.Mendel@iaik.TUGraz.at

**Abstract.** In this article, we present an improved collision attack on the hash function proposed by Shin *et al.* at PKC'98. The attack has a complexity of about $2^{20.5}$ hash computations, while the previous attack of Chang *et al.* presented at SAC 2002 has a complexity of about $2^{37.13}$ hash computations. In the analysis of the hash function we combined existing approaches with recent results in cryptanalysis of hash functions. We show that message-dependent rotations can be exploited to construct collisions. The weak design of the step function facilitates high-probability multi-block collisions.

**Keywords:** cryptanalysis, collision attack, differential attack, collision, near-collision, hash functions

## 1 Introduction

Recently, weaknesses in many commonly used hash functions, such as MD5 and SHA-1 have been found. These breakthrough results in the cryptanalysis of hash functions are the motivation for intensive research in the design and analysis of hash functions. It is of special interest whether or not existing attack methods, which have been successfully used to break MD5 and SHA-1, can be extended to other hash functions as well. Based on this motivation, we present a collision attack on the hash function proposed by Shin *et al.* at PKC'98. The attack adapts and extends existing methods leading to an attack complexity of about $2^{20.5}$ hash computations. This is an improvement by a factor of $2^{16.5}$ compared to the collision attack presented by Chang *et al.* at SAC 2002. In addition to the improved collision attack this article illustrates how powerful recently invented methods are and shows how they can be extended to other hash functions such as the hash function proposal from PKC'98.

[⋆] The work described in this paper has been supported in part by the European Commission through the IST Programme under contract IST2002507 932 ECRYPT. The information in this paper is provided as is, and no guarantee or warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

[⋆⋆] This author is supported by the Austrian Science Fund (FWF), project P18138.

http://dx.doi.org/10.1007/11927587_3

**Table 1.** Notation

| Notation | Meaning |
|---|---|
| $A \vee B$ | logical OR of two bit-strings $A$ and $B$ |
| $A \wedge B$ | logical AND of two bit-strings $A$ and $B$ |
| $A \oplus B$ | logical XOR of two bit-strings $A$ and $B$ |
| $A \lll n$ | bit-rotation of $A$ by $n$ positions to the left |
| $A \ggg n$ | bit-rotation of $A$ by $n$ positions to the right |
| $M_j$ | message block $j$ (512-bits) |
| $m_i$ | message word $i$ (32-bits) |
| $w_i$ | expanded message word $i$ (32-bits) |
| *step* | single execution of the step function |
| *round* | set of consecutive *steps*, has a size of 24 (1 *round* = 24 *steps*) |

An important contribution of this article is that we analyze message-dependent rotations, a property not existing for instance in MD5 and SHA-1. Our conclusions are that the message-dependent rotations decrease the security of the hash function. The weak design of the step function in combination with the used Boolean functions facilitates the construction of high-probability multi-block collisions.

The remainder of this article is structured as follows. A description of the hash function is given in Section 2. The basic attack strategy that is used to improve all existing collision attacks on the hash function is described in Section 3. In Section 4, we present the characteristic used for the improved collision attack. Based on this characteristic we construct a near-collision in Section 5 and finally we present a collision in Section 6. A detailed analysis of the overall attack complexity is given in Section 7. A sample colliding message pair is presented in Section 8 and conclusions are given in Section 9.

## 2 Description of the Hash Function proposed at PKC'98

The hash function was proposed by Shin *et al.* [5] at PKC'98. It is an iterative hash function that processes 512-bit input message blocks and produces a 160-bit hash value. In the following, we briefly describe the hash function. It basically consists of two parts: message expansion and state update transformation. A detailed description of the hash function is given in [5].

Since Shin *et al.* did not name their hash function, we will refer to it as PKC-hash for the remainder of this article. Throughout the remainder of this article, we will follow the notation given in Table 1.

**Message Expansion.** The message expansion of PKC-hash is a permutation of 24 expanded message words in each round, where different permutation values are used in each round. The 24 expanded message words $w_i$ used in each round are constructed from the 16 input message words $m_i$ in the following way:

$$w_i = \begin{cases} m_i & 0 \leq i \leq 15 \\ (w_{i-4} \oplus w_{i-9} \oplus w_{i-14} \oplus w_{i-16}) \lll 1 & 16 \leq i \leq 23. \end{cases}$$

For the ordering of the message words the permutation $\rho$ is used.

| Round 1 | Round 2 | Round 3 | Round 4 |
|---|---|---|---|
| $id$ | $\rho$ | $\rho^2$ | $\rho^3$ |

The permutation $\rho$ is defined as following.

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\rho(i)$ | 4 | 21 | 17 | 1 | 23 | 18 | 12 | 10 | 5 | 16 | 8 | 0 |

| $i$ | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\rho(i)$ | 20 | 3 | 22 | 6 | 11 | 19 | 15 | 2 | 7 | 14 | 9 | 13 |

**State Update Transformation.** The state update transformation starts from a (fixed) initial value $IV$ of five 32-bit registers and updates them in 4 rounds of 24 steps each. Figure 1 shows one step of the state update transformation of the hash function.

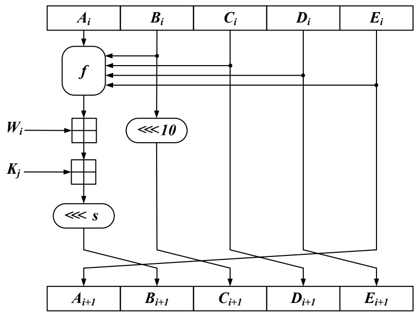

**Fig. 1.** The step function of the hash function.

The function $f$ is different in each round: $f_0$ is used in the first round, $f_1$ is used in round 2 and round 4, and $f_2$ is used in round 3.

$$f_0(x_1, x_2, x_3, x_4, x_5) = (x_1 \wedge x_2) \oplus (x_3 \wedge x_4) \oplus (x_2 \wedge x_3 \wedge x_4) \oplus x_5$$
$$f_1(x_1, x_2, x_3, x_4, x_5) = x_2 \oplus ((x_4 \wedge x_5) \vee (x_1 \wedge x_3))$$
$$f_2(x_1, x_2, x_3, x_4, x_5) = x_1 \oplus (x_2 \wedge (x_1 \oplus x_4)) \oplus (((x_1 \wedge x_4) \oplus x_3) \vee x_5)$$

A step constant $K_j$ is added in every step; the constant is different for each round. Different rotation values $s_i$ are used in each step. The rotation values are dependent on the message words. The rotation values $s_i$, for $i = 0, \ldots, 23$ are calculated in the following way:

$$s_i = w_i \mod 32$$

The rotation values are permuted in each round. Again the permutation $\rho$ is used, but in reverse sequence.

| Round 1 | Round 2 | Round 3 | Round 4 |
|---|---|---|---|
| $\rho^3$ | $\rho^2$ | $\rho^1$ | $id$ |

After the last step of the state update transformation, the initial value and the output values of the last step are combined, resulting in the final value of one iteration known as Davies-Meyer hash construction (feed forward). In detail, the feed forward is a word-wise modular addition of the $IV$ and the output of the state update transformation. The result is the final hash value or the initial value for the next message block.

## 3 Our Attack Strategy

In the following, we present the attack strategy we use to improve the collision attack on PKC-hash. It is based on recent results in cryptanalysis of hash functions [6,7,8]. The attack can be basically described as follows.

1. Find a characteristic for the hash function that holds with high probability for the last 3 rounds of the hash function.
2. Find a characteristic (not necessary with high probability) for the first round.
3. Use basic message modification techniques to fulfill all conditions for the characteristic in the first round.
4. Use random trials to find values for the message bits such that the message also follows the characteristic in the last 3 rounds.

**Observation 1** *For the first steps the probability of the characteristic is not important, because the conditions that have to be satisfied such that the characteristic holds can be easily fulfilled for these steps using basic message modification techniques [6,8].*

**Observation 2** *Multi-block messages can be used to turn related near-collisions into a collision.*

Since Biham and Chen observed in [1] that near-collisions are easier to find than collisions, we will use Observation 2 in Section 4 to improve our attack.

To find a characteristic which holds with high probability, we exploit the structure of the hash function. Considering the design of the step function we made the following observations. We use these observations in Section 4 to construct a characteristic which holds with high probability.

**Observation 3** *The rotation values $s_j$ of the state update transformation are dependent on the values of the expanded message words.*

This gives the attacker more degrees of freedom for constructing a *good* characteristic. The observation can be used to improve the probability of the characteristic significantly, see Section 4.

**Observation 4** *Due to the message-dependent rotation values the step function is not invertible.*

This means we could try to construct collisions by using different $w_j, w'_j$ and $s_j, s'_j$ which have the property that $B_{j+1} = B'_{j+1}$. However, the complexity for constructing a collision with this method is too high for a reasonable attack.

**Observation 5** *The function $f$ can either preserve or absorb an input difference. This gives the attacker more flexibility for constructing the characteristic.*

**Observation 6** *Only the expanded message word $w$ and the output of the $f$ function is used to update state variable $B$ in each step.*

From Observation 5 it follows that differences in the state variables can be canceled by using the differential properties of the $f$ function. In particular, a difference in $B_{j+1}$ introduced in step $j$ through a difference in the expanded message word $w_j$ (referred to as disturbance) can be canceled within a few steps.

The number of steps needed for canceling a single disturbance depends on the function $f$ and the rotation values $s_j$ of the step function. While we need at least 5 steps to cancel a disturbance in round 3, we need at least 6 steps to cancel a disturbance in round 1, 2 and 4. This is due to the fact that we cannot always block the input differences of $f_0$ and $f_1$. A detailed analysis of the differential properties of $f_0$, $f_1$ and $f_2$ is given in [2]. In Appendix A, we give the local collisions and related probabilities for each round of the hash function.

However, to get a characteristic that holds with high probability, we have to minimize the number of disturbances in each round. This can be done by minimizing the number of differences in the expanded message.

**Observation 7** *The minimal number of differences in the expanded message words is 2 for each round (8 in total).*

This follows from the inspection of the linear message expansion, which uses 16 input message words to generate 24 expanded message words. A permutation of these 24 words is used in each round of the hash function and hence the number of disturbances in each round is the same. Based on these observations, we will construct a characteristic which holds with high probability in Section 4.

## 4 The New Improved Collision Attack

In this section, we describe the characteristic we use for the improved collision attack on PKC-hash. Before presenting the characteristic in Section 4.2 and Section 4.3, we briefly describe how the characteristic has been obtained in the following section.

### 4.1 Finding a *good* Characteristic

In the past, it has been shown that it is easier to find near-collision than collision within a hash function. Since two message blocks can be used to turn a near-collision into a collision (see Observation 2), we will consider near-collisions in the analysis as well. Note that the attacker has full control on the expanded message words in the first 16 steps of the hash function. Hence, it is easy to find a message that follows the characteristic in the first 16 steps, because the conditions for the first 16 steps can be fulfilled by using basic message modification techniques. Therefore, only the probability of the characteristic in the remaining 80 steps determines the attack complexity. In order to keep the complexity low, we want to have as few disturbances in the last 80 steps as possible. This can be achieved by choosing the differences in the message words in such a way that there are only a few differences in the expanded message. We have found the following 4 options which have only 2 differences in the expanded message words in each round. Note that 2 is a matching lower bound for the number of differences in the expanded message words in each round (see Observation 7).

$$\Delta w_{8,j} = \Delta w_{13,j} \tag{1}$$
$$\Delta w_{9,j} = \Delta w_{14,j} \tag{2}$$
$$\Delta w_{10,j} = \Delta w_{15,j} \tag{3}$$
$$\Delta w_{11,j} = \Delta w_{20,(j+1) \mod 32} \tag{4}$$

Out of this four cases, we select option (2) with $j = 32$, *i.e.* $\Delta w_9 = \Delta w_{14} =$ 80000000 to maximize the probability of the characteristic. By choosing the difference in the MSB no conditions are needed for the modular addition, which decreases the attack complexity. This is due to the fact that modular addition behaves like an XOR for differences in the MSB.

Furthermore, we reduce the number of local collisions in round 1, 2, and 4 from the expected value of 2 to 1 by selecting option (2). Therefore, the probability of our characteristic is much higher than the probability of the characteristics used in [2] and [3]. On the one hand, we increase the probability of the characteristic remarkably, but on the other hand this leads to a nonzero difference in the state variables after the last step of the state update transformation (a near-collision). However, as it will be described in Section 6, we can use a second message block to turn this near-collision into a collision without significantly increasing the attack complexity.

## 4.2 Characteristic for the First Round

The characteristic for the first round (24 steps) has probability $2^{-7.8}$. However, all the conditions for the first 16 steps can be fulfilled by basic message modification techniques. Since the characteristic has probability 1 for steps 16-23 (see Table 2), the probability for the characteristic in the first round is always 1. Therefore, the attack complexity only depends on the probability of the characteristic in the last 3 rounds. In Section 4.3, we give a characteristic for the remaining 3 rounds which holds with high probability. The characteristic for the first round of the hash function is given in Table 2. To improve readability, we use hexadecimal notation and denote the zero difference by '0'.

**Table 2.** Characteristic for the first round of PKC-hash.

| step | $\Delta A$ | $\Delta B$ | $\Delta C$ | $\Delta D$ | $\Delta E$ | $\Delta w$ | $s$ | probability |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | - | 1 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | - | 1 |
| 9 | 0 | 0 | 0 | 0 | 0 | 80000000 | 22 | 1 |
| 10 | 0 | 00200000 | 0 | 0 | 0 | 0 | 0 | 1/4 |
| 11 | 0 | 00200000 | 80000000 | 0 | 0 | 0 | - | 3/8 |
| 12 | 0 | 0 | 80000000 | 80000000 | 0 | 0 | - | 3/4 |
| 13 | 0 | 0 | 0 | 80000000 | 80000000 | 0 | - | 1/4 |
| 14 | 80000000 | 0 | 0 | 0 | 80000000 | 80000000 | - | 1/2 |
| 15 | 80000000 | 0 | 0 | 0 | 0 | 0 | - | 1/2 |
| 16 | 0 | 0 | 0 | 0 | 0 | 0 | - | 1 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 23 | 0 | 0 | 0 | 0 | 0 | 0 | - | 1 |
|  | 0 | 0 | 0 | 0 | 0 |  |  | $2^{-7.8}$ |

## 4.3 Characteristic for the Last 3 Rounds

Since all the conditions on the characteristic in the first round can be easily fulfilled, only the probability of the characteristic in the last 3 rounds determines the attack complexity. Therefore, we are searching for a characteristic which holds with high probability in the last 3 rounds. In this section, we give a characteristic for the last 3 rounds which has probability of $2^{-20.5}$. To maximize the probability of the characteristic, we use the fact that the rotation values of PKC-hash are dependent on the expanded message words (see Observation 3). The rotation values can be set to arbitrary values by setting additional conditions on the expanded message words. The characteristic and necessary rotation values are given Table 3. Note that we do not count the conditions for the rotation values to the attack complexity, since these can be easily fulfilled in advance by fixing the values of the expanded message words.

**Table 3.** Characteristic for the last 3 rounds (round 2-4) of PKC-hash.

| step | $\Delta A$ | $\Delta B$ | $\Delta C$ | $\Delta D$ | $\Delta E$ | $\Delta w$ | $s$ | probability |
|---|---|---|---|---|---|---|---|---|
| 24 | 0 | 0 | 0 | 0 | 0 | 0 | - | 1 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 44 | 0 | 0 | 0 | 0 | 0 | 0 | - | 1 |
| 45 | 0 | 0 | 0 | 0 | 0 | 80000000 | 0 | 1 |
| 46 | 0 | 80000000 | 0 | 0 | 0 | 80000000 | - | 1 |
| 47 | 0 | 0 | 00000200 | 0 | 0 | 0 | - | 5/8 |
| 48 | 0 | 0 | 0 | 00000200 | 0 | 0 | - | 1/2 |
| 49 | 0 | 0 | 0 | 0 | 00000200 | 80000000 | 0 | 1/2 |
| 50 | 00000200 | 80000000 | 0 | 0 | 0 | 0 | - | 1/4 |
| 51 | 0 | 0 | 00000200 | 0 | 0 | 0 | - | 1/2 |
| 52 | 0 | 0 | 0 | 00000200 | 0 | 0 | - | 1/2 |
| 53 | 0 | 0 | 0 | 0 | 00000200 | 0 | - | 1/2 |
| 54 | 00000200 | 0 | 0 | 0 | 0 | 0 | - | 1/2 |
| 55 | 0 | 0 | 0 | 0 | 0 | 0 | - | 1 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 61 | 0 | 0 | 0 | 0 | 0 | 0 | - | 1 |
| 62 | 0 | 0 | 0 | 0 | 0 | 80000000 | 0 | 1 |
| 63 | 0 | 80000000 | 0 | 0 | 0 | 0 | - | 1/2 |
| 64 | 0 | 0 | 00000200 | 0 | 0 | 0 | - | 1/2 |
| 65 | 0 | 0 | 0 | 00000200 | 0 | 0 | - | 1/2 |
| 66 | 0 | 0 | 0 | 0 | 00000200 | 0 | - | 1/2 |
| 67 | 00000200 | 0 | 0 | 0 | 0 | 0 | - | 1/2 |
| 68 | 0 | 0 | 0 | 0 | 0 | 0 | - | 1 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 74 | 0 | 0 | 0 | 0 | 0 | 0 | - | 1 |
| 75 | 0 | 0 | 0 | 0 | 0 | 80000000 | 0 | 1 |
| 76 | 0 | 80000000 | 0 | 0 | 0 | 0 | 10 | 1 |
| 77 | 0 | 00000200 | 00000200 | 0 | 0 | 0 | - | 3/8 |
| 78 | 0 | 0 | 00080000 | 00000200 | 0 | 0 | - | 25/64 |
| 79 | 0 | 0 | 0 | 00080000 | 00000200 | 0 | - | 25/64 |
| 80 | 00000200 | 0 | 0 | 0 | 00080000 | 0 | - | 25/64 |
| 81 | 00080000 | 0 | 0 | 0 | 0 | 0 | - | 5/8 |
| 82 | 0 | 0 | 0 | 0 | 0 | 0 | - | 1 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 92 | 0 | 0 | 0 | 0 | 0 | 0 | - | 1 |
| 93 | 0 | 0 | 0 | 0 | 0 | 80000000 | 0 | 1 |
| 94 | 0 | 80000000 | 0 | 0 | 0 | 0 | 0 | 1 |
| 95 | 0 | 80000000 | 00000200 | 0 | 0 | 0 | 0 | 5/8 |
|  | 0 | 80000000 | 00000200 | 00000200 | 0 |  |  | $2^{-20.5}$ |

## 5 A Near-Collision Producing Characteristic

The characteristic for the first round (Table 2) and the characteristic for the remaining 3 rounds (Table 3) can be combined to construct a near-collision in one iteration of the hash function. Using the most naive method (random trials) to find a message following the characteristic in the last 80 steps, we get a complexity close to $2^{20.5}$ hash computations. Hence, a near-collision can be found in PKC-hash with complexity about $2^{20.5}$ hash computations. By using two message blocks, we can turn this near-collision into a collision. This is described in detail in the following section.

## 6 Collision Producing Characteristic

In [8], Wang *et al.* show how a two-block message can be used to construct a collision for MD5. The main idea is that a second message block can be used to turn a near-collision after the first block into a collision after feed forward of the second block with a certain probability. Therefore, a slightly modified characteristic is required in the first round of the second block. This is depicted in Fig. 2.

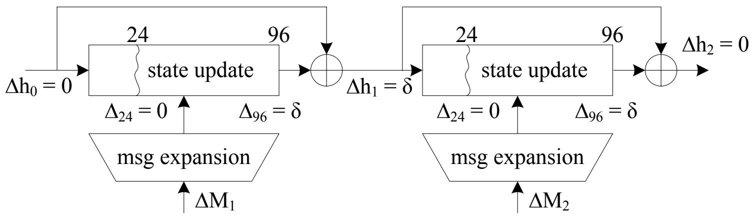

**Fig. 2.** A two-block collision in the hash function.

While we use a characteristic of the form:

$$\varDelta h_0(0,0,0,0,0) \mapsto \varDelta_{24}(0,0,0,0,0) \mapsto \varDelta_{96}(0,2^{31},2^9,2^9,0) \tag{5}$$

in the first message block, we need a characteristic of the form:

$$\varDelta h_1(0,2^{31},2^9,2^9,0) \mapsto \varDelta_{24}(0,0,0,0,0) \mapsto \varDelta_{96}(0,2^{31},2^9,2^9,0) \tag{6}$$

in the second block. Constructing such a characteristic is quite easy in our particular case. Due to the weak design of the step function we can block differences in the state variables in each step of the hash function depending on the differential properties of the $f$ function (see Observation 5). Hence, we can cancel all differences in the state variables at the beginning of the second block within a few steps in the first round. Note that the differential behavior of the $f$ function depends on the values of the state variables and we cannot control it in the first steps of the second block. This is due to the fact that the initial value of the second block is fixed by calculating the first block. Therefore, in principle the characteristic for the second block cannot be fixed until the first block has been calculated.

However, in practice due to the large degree of freedom we have in our collision attack, we can use the same characteristic in the second block as we use in the first block and cancel all differences in the state variables in the first 9 steps of the second block without affecting any of the conditions of the original characteristic.

Note that the probability of the characteristic of the second message block for the first 16 steps can again be neglected. Hence, the probability of the second block is also $2^{20.5}$. By combining both message blocks we can construct a collision after the feed forward of the second block with a certain probability. In detail, we can find a two message block collision for PKC-hash with probability close to $2^{-22.85}$. A detailed analysis is given the following section.

## 7 Overall Collision Attack Complexity

The complexity of the collision attack only depends on the probability of the characteristic in the last 3 rounds of both message blocks. Note that some additional conditions have to be met to guarantee that all differences cancel out in the feed forward after the second block. Therefore, the second block has a slightly lower probability than the first block.

As shown in Section 4, the characteristic for the last 3 rounds has a probability of $2^{-20.5}$ and therefore a straightforward implementation of the collision-search algorithm would yield a complexity of about $2^{20.5}$ hash computations for the first block and $2^{20.5} \cdot 2^2$ hash computations for the second block. In order to obtain a collision after the feed-forward of the second block, the following two conditions on the state variables at the output of the second block have to be satisfied

$$C_{0,10} \oplus C_{81,10} = 1$$
$$D_{0,10} \oplus D_{81,10} = 1$$

to guarantee that all differences cancel in the feed forward of the second block. Since the third difference is in the MSB (see Equation (6)), the difference cancels out with probability 1 and no condition is needed. Hence, the second block has a complexity of $2^{20.5} \cdot 2^2$ hash computations and the final attack complexity would be $2^{20.5} + 2^{20.5} \cdot 2^2 = 2^{22.85}$ hash computations to construct a collision in the hash function.

However, there are several simple methods to improve the efficiency of the attack. In [9], Wang *et al.* use a so-called *early-stop* technique to improve the attack complexity for SHA-0 by a factor of 8. The main idea is that only a few steps have to be computed after the basic message modification to check whether or not the message follows the characteristic. If the message does not follow the characteristic the collision-search algorithm aborts the current computation and starts again with a new message. It is clear that this reduces the attack complexity. In our case, we can improve the attack complexity by a factor of 8/3, since we have to calculate on average 36 steps out of 96 steps to check whether the chosen message follows the characteristic or not.

Furthermore, it has been shown recently in [4] that it is useful to look at all possible characteristics in the second part of the attack (the part after message modification, *i.e.* round 2-4) to get an accurate estimation of the attack complexity. Since we use random trials to find a message following the characteristic after the first round, we do not need to stick to the characteristic given in Section 4.3. The only constraint we have is that there has to be a certain output difference after step 96. Hence, other characteristics (with lower probability) for the last 3 rounds do contribute as well. Therefore, the effective attack complexity is slightly lower. We have done this analysis for the third round of the hash function and have achieved an improvement by a factor of about 2. Note that we have fixed additional rotation values in round 3 to maximize that factor.

Hence, we can update the final attack complexity to $2^{20.5-2.4}$ hash computations for the first message block and $2^{22.5-2.4}$ for the second block. Therefore, the final attack complexity is $2^{20.5-2.4} + 2^{22.5-2.4} \approx 2^{20.5}$ hash computations.

## 8 A Colliding Message for the Hash Function

Applying our improved collision attack to PKC-hash, we can construct a two message block collision with a complexity of $2^{20.5}$ hash computations. The colliding message pair is given in Table 4. Note that $h_0$ is the initial value, $h_1$ is the intermediate hash value after the first block, and $h_2$ is the final hash value after the second block (see Fig. 2 in Section 4).

**Table 4.** Colliding message pair for the hash function.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $h_0$ | 67452301 | EFCDAB89 | 98BADCEF | 10325476 | C3D2E1F0 | | | |
| $M_0$ | 210A7ED6 | 69EC9B20 | 71E79487 | ECDB11C0 | CCE394EA | EBA83742 | 44A26AC0 | 9A644570 |
| | E78BA0F0 | D0CD3794 | AC8A28BB | 29303480 | F9A7F632 | 0F886620 | 28E118E9 | 39E4CF77 |
| $M_0'$ | 210A7ED6 | 69EC9B20 | 71E79487 | ECDB11C0 | CCE394EA | EBA83742 | 44A26AC0 | 9A644570 |
| | E78BA0F0 | 50CD3794 | AC8A28BB | 29303480 | F9A7F632 | 0F886620 | A8E118E9 | 39E4CF77 |
| $\Delta M_0$ | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| | 80000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 80000000 | 00000000 |
| $h_1$ | E1A286A2 | 5E619A9E | F341C16E | 8A3B4927 | AFCFF8D2 | | | |
| $h_1'$ | E1A286A2 | DE619A9E | F341C36E | 8A3B4B27 | AFCFF8D2 | | | |
| $\Delta h_1$ | 00000000 | 80000000 | 00000200 | 00000200 | 00000000 | | | |
| $M_1$ | 89221C96 | 237E9860 | 76346FC0 | C5F4F3E0 | B66B5EAA | D025B4C9 | BE742420 | E1362EC6 |
| | 084DB7A0 | 3F231F9A | D883A03A | AFCB10A0 | CDA285EE | 24630660 | BE9599C0 | F6F63E65 |
| $M_1'$ | 89221C96 | 237E9860 | 76346FC0 | C5F4F3E0 | B66B5EAA | D025B4C9 | BE742420 | E1362EC6 |
| | 084DB7A0 | BF231F9A | D883A03A | AFCB10A0 | CDA285EE | 24630660 | 3E9599C0 | F6F63E65 |
| $\Delta M_1$ | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| | 80000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 80000000 | 00000000 |
| $h_2$ | B141281F | FC5A987C | FB473C39 | A9864410 | 21ACD08E | | | |
| $h_2'$ | B141281F | FC5A987C | FB473C39 | A9864410 | 21ACD08E | | | |

## 9 Conclusion

In this article, we used recent results in the cryptanalysis of hash functions to improve the collision attack on the hash function proposed by Shin *et al.* at PCK'98. We have shown that a collision can be found in the hash function with a complexity below $2^{20.5}$ hash computations. In detail, we improve the results of Chang *et al.* [2] with the new collision attack by a factor of $2^{16.5}$ using a new differential characteristic and exploiting basic message modification techniques and multi-block collisions.

We point out that the weakness of the hash function comes from the message-dependent rotation values and the weak design of step function. Firstly, the degrees of freedom the attacker has to choose the rotation values can be used to increase the probability of the attack. Secondly, the weak design of the step function facilitates high-probability multi-block collisions. Differences in state variables in the first step can easily be canceled within a few steps using the differential properties of the $f$ function.

Hence, we conclude that the Boolean functions used in the state update transformation have to be chosen carefully and using only the expanded message words and the output of the $f$ function to update the state variables is insufficient. Furthermore, rotation values depending on the message words can reduce the security of hash functions.

## Acknowledgements

The authors wish to thank Vincent Rijmen, and the anonymous referees for useful comments and discussions.

## References

1. Eli Biham and Rafi Chen. Near-Collisions of SHA-0. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *LNCS*, pages 290–305. Springer, 2004.
2. Donghoon Chang, Jaechul Sung, Soo Hak Sung, Sangjin Lee, and Jongin Lim. Full-Round Differential Attack on the Original Version of the Hash Function Proposed at PKC'98. In Kaisa Nyberg and Howard M. Heys, editors, *Selected Areas in Cryptography, 9th Annual International Workshop, SAC 2002, St. John's, Newfoundland, Canada, August 15-16, 2002. Revised Papers*, volume 2595 of *Lecture Notes in Computer Science*, pages 160–174. Springer, 2002.
3. Daewan Han, Sangwoo Park, and Seongtaek Chee. Cryptanalysis of the Modified Version of the Hash Function Proposed at PKC'98. In Joan Daemen and Vincent Rijmen, editors, *Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002, Revised Papers*, volume 2365 of *Lecture Notes in Computer Science*, pages 252–262. Springer, 2002.

4. Florian Mendel, Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. The Impact of Carries on the Complexity of Collision Attacks on SHA-1. In Matt Robshaw, editor, *Fast Software Encryption, 13th International Workshop, FSE 2006, Graz, Austria, March 15-17, 2006, Pre-Proceedings*, 2006.
5. Sang Uk Shin, Kyung Hyune Rhee, Dae-Hyun Ryu, and Sangjin Lee. A New Hash Function Based on MDx-Family and Its Application to MAC. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography, First International Workshop on Practice and Theory in Public Key Cryptography, PKC '98, Pacifico Yokohama, Japan, February 5-6, 1998, Proceedings*, volume 1431 of *Lecture Notes in Computer Science*, pages 234–246. Springer, 1998.
6. Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the Hash Functions MD4 and RIPEMD. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings*, volume 3494 of *LNCS*, pages 1–18. Springer, 2005.
7. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding Collisions in the Full SHA-1. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005, 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *LNCS*, pages 17–36. Springer, 2005.
8. Xiaoyun Wang and Hongbo Yu. How to Break MD5 and Other Hash Functions. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings*, volume 3494 of *LNCS*, pages 19–35. Springer, 2005.
9. Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin. Efficient Collision Search Attacks on SHA-0. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005, 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *LNCS*, pages 1–16. Springer, 2005.

## A Local Collisions

In this section, we will give the best local collision for each round of the PKC-hash. Since $f_1$ is used in round 2 and round 4 the local collisions are equal for both rounds.

**Table 5.** Local Collision in Round 1 ($f_0$).

| step | $\Delta A$ | $\Delta B$ | $\Delta C$ | $\Delta D$ | $\Delta E$ | $\Delta w$ | $s$ | probability |
|---|---|---|---|---|---|---|---|---|
| j | 0 | 0 | 0 | 0 | 0 | 80000000 | 0 | 1 |
| j+1 | 0 | 80000000 | 0 | 0 | 0 | 0 | 0 | 1/2 |
| j+2 | 0 | 80000000 | 00000200 | 0 | 0 | 0 | - | 3/8 |
| j+3 | 0 | 0 | 00000200 | 00000200 | 0 | 0 | - | 3/4 |
| j+4 | 0 | 0 | 0 | 00000200 | 00000200 | 0 | - | 1/4 |
| j+5 | 00000200 | 0 | 0 | 0 | 00000200 | 0 | - | 1/2 |
| j+6 | 00000200 | 0 | 0 | 0 | 0 | 0 | - | 1/2 |
|  | 0 | 0 | 0 | 0 | 0 |  |  | $2^{-6.8}$ |

**Table 6.** Local Collision in Round 2/4 ($f_1$).

| step | $\Delta A$ | $\Delta B$ | $\Delta C$ | $\Delta D$ | $\Delta E$ | $\Delta w$ | $s$ | probability |
|---|---|---|---|---|---|---|---|---|
| j | 0 | 0 | 0 | 0 | 0 | 80000000 | 0 | 1 |
| j+1 | 0 | 80000000 | 0 | 0 | 0 | 0 | 10 | 1 |
| j+2 | 0 | 00000200 | 00000200 | 0 | 0 | 0 | - | 3/8 |
| j+3 | 0 | 0 | 00080000 | 00000200 | 0 | 0 | - | 25/64 |
| j+4 | 0 | 0 | 0 | 00080000 | 00000200 | 0 | - | 25/64 |
| j+5 | 00000200 | 0 | 0 | 0 | 00080000 | 0 | - | 25/64 |
| j+6 | 00080000 | 0 | 0 | 0 | 0 | 0 | - | 5/8 |
| | 0 | 0 | 0 | 0 | 0 | | | $2^{-7.2}$ |

**Table 7.** Local Collision in Round 3 ($f_2$).

| step | $\Delta A$ | $\Delta B$ | $\Delta C$ | $\Delta D$ | $\Delta E$ | $\Delta w$ | $s$ | probability |
|---|---|---|---|---|---|---|---|---|
| j | 0 | 0 | 0 | 0 | 0 | 80000000 | 0 | 1 |
| j+1 | 0 | 80000000 | 0 | 0 | 0 | 0 | - | 1/2 |
| j+2 | 0 | 0 | 00000200 | 0 | 0 | 0 | - | 1/2 |
| j+3 | 0 | 0 | 0 | 00000200 | 0 | 0 | - | 1/2 |
| j+4 | 0 | 0 | 0 | 0 | 00000200 | 0 | - | 1/2 |
| j+5 | 00000200 | 0 | 0 | 0 | 0 | 0 | - | 1/2 |
| | 0 | 0 | 0 | 0 | 0 | | | $2^{-5}$ |