Efficient Collision Search Attacks on SHA-0

Xiaoyun Wang[1⋆], Hongbo Yu[2], and Yiqun Lisa Yin[3]

[1] Shandong University, China, xywang@sdu.edu.cn
[2] Shandong University, China, yhb@mail.sdu.edu.cn
[3] Independent Security Consultant, Greenwich CT, US, yyin@princeton.edu

Abstract. In this paper, we present new techniques for collision search in the hash function SHA-0. Using the new techniques, we can find collisions of the full 80-step SHA-0 with complexity less than 2^{39} hash operations.

Keywords: Hash functions, Collision search attacks, SHA-0, SHA-1.

1 Introduction

The hash function SHA-0 was issued in 1993 as a federal standard by NIST. A revised version called SHA-1 was later issued in 1995 as a replacement for SHA-0. The only difference between the two hash functions is the additional rotation operation in the message expansion of SHA-1, which is supposed to provide more security. Both hash functions are based on the design principles of MD4.

In 1997, Wang found an attack on SHA-0 [14] which produces a collision with probability 2^{-58} by utilizing algebraic methods to derive a collision differential path. In 1998, Chabaud and Joux [6] independently found the same differential path through computer search. In August 2004, Joux [7] announced the first real collision of SHA-0, which consists of four message blocks (a pair of 2048-bit input messages). The collision search took about 80,000 hours of CPU time (three weeks of real time) and is estimated to have a complexity of about 2^{51} hash operations. To our knowledge, this is the best existing attack on the full 80-step SHA-0 prior to the work reported here.

The attacks in [14, 6] found a differential path which is composed of certain 6-step local collisions. There is an obstacle to further improve these attacks, as finding a differential characteristic for two consecutive local collisions corresponding to two consecutive disturbances in the first round turns out to be impossible. This phenomenon makes it difficult to find a differential path which has a smaller number of local collisions in rounds 2-4 and no consecutive local collisions in the first round.

In this paper, we introduce a new cryptanalytic method to cope with this difficulty. Our analysis includes the following techniques: Firstly, we identify an "impossible" differential path with few local collisions in rounds 2-4 and some

⋆ Supported by the National Natural Science Foundation of China (NSFC Grant No.90304009) and Program for New Century Excellent Talents in University.

consecutive local collisions in round 1. Secondly, we transform the impossible differential path into a possible one. Thirdly, we derive a set of conditions which guarantee that the modified differential path holds. Finally, we design message modifications to correct all the unfulfilled conditions in the first round as well as some such conditions in the second round. With these techniques, we can find collisions of the full SHA-0 with at most 2^{39} hash operations, which is a major improvement over existing attacks. The same techniques can be used to find near collisions of SHA-0 with complexity about 2^{33} hash operations.

We note that the new techniques have also been proven to be effective in the analysis of SHA-1[16].

The rest of the paper is organized as follows. In Section 2, we give a description of SHA-0. In Section 3, we provide an overview of the original attack on SHA-0 [14] and subsequent improvements [15, 1, 2, 7, 3]. In Section 4, we review the "message modification techniques" presented in [11–13] to break HAVA-128, MD5, MD4 and RIPEMD, and consider their effectiveness in improving existing attacks on SHA-0. In Section 5, we present our new collision search attacks on SHA-0. In Section 6, we give an example of real collision of SHA-0 found by computer search using the new techniques. We conclude the paper in Section 7.

2 Description of SHA-0

The hash function SHA-0 takes a message of length less than 2^{64} bits and produces a 160-bit hash value. The input message is padded and then processed in 512-bit blocks in the Damgård/Merkle iterative structure. Each iteration invokes a so-called compression function which takes a 160-bit chaining value and a 512-bit message block and outputs another 160-bit chaining value. The initial chaining value (called IV) is a set of fixed constants, and the final chaining value is the hash of the message.

In what follows, we describe the compression function of SHA-0. For each 512-bit block of the padded message, divide it into 16 32-bit words, $(m_0, m_1, ..., m_{15})$. The message words are first expanded as follows: for $i = 16, ..., 79$,

$$m_i = m_{i-3} \oplus m_{i-8} \oplus m_{i-14} \oplus m_{i-16}.$$

The expanded message words are then processed in four rounds, each consisting of 20 steps. The step function is defined as follows.

For $i = 1, 2, ..., 80$,

$$\begin{aligned}
a_i &= (a_{i-1} << 5) + f_i(b_{i-1}, c_{i-1}, d_{i-1}) + e_{i-1} + m_{i-1} + k_i \\
b_i &= a_{i-1} \\
c_i &= b_{i-1} << 30 \\
d_i &= c_{i-1} \\
e_i &= d_{i-1}
\end{aligned}$$

The initial chaining value $IV = (a_0, b_0, c_0, d_0, e_0)$ is defined as:

$$(0x67452301,\ 0xefcdab89,\ 0x98badcfe,\ 0x10325476,\ 0xc3d2e1f0)$$

Each round employs a different Boolean function f_i and constant k_i, which is summarized in Table 1.

rounds	steps	Boolean function f_i	constant k_i
1	$1-20$	IF: $(x \wedge y) \vee (\neg x \wedge z)$	0x5a827999
2	$21-40$	XOR: $x \oplus y \oplus z$	0x6ed6eba1
3	$41-60$	MAJ: $(x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$	0x8fabbcdc
4	$61-80$	XOR: $x \oplus y \oplus z$	0xca62c1d6

Table 1. Boolean functions and constants in SHA-0

3 Previous Attacks on SHA-0

In this section, we first describe the original collision attack on SHA-0 given by Wang in 1997 [14]. This sets up the basic framework for introducing our new techniques later on. For other independent attacks on SHA-0 the reader may wish to refer to [15, 6, 1, 7, 3].

3.1 Local Collisions of SHA-0

Informally, a local collision is a collision within a few steps of the hash function. A simple yet very important observation is that SHA-0 has a 6-step local collision that can start at any step i, and this type of local collision is the basic component in constructing full collisions.

Suppose a message difference in bit j first occurs in Step i (e.g., $\Delta m_{i-1,j} = 1$.) The difference will affect the chaining variables a, b, c, d, e consecutively in the next five steps. In order to offset these differences and reach a local collision, more message differences are introduced in subsequent message words. In Table 2, we illustrate the differential path of such a local collision. The chaining variable conditions under which the local collisions hold were given in [14, 15].

The probability associated with the above local collision depends on the Boolean function, the bit position j, and some conditions on the message bits. The differential attack in [14] and [6] chooses $j = 2$ so that $j + 30$ becomes the MSB[4] to eliminate the carry effect in the last three steps. In addition, the following condition

$$m_{i,2} = \neg m_{i+1,7}$$

[4] Throughout this paper, we label the bit positions in a 32-bit word as $32, 31, 30, ..., 3, 2, 1$, where bit 32 is the most significant bit and bit 1 is the least significant bit. Please note that this is different from the convention of labelling bit positions from 31 to 0.

step	Δm	Δa	Δb	Δc	Δd	Δe	Conditions
i	2^j	2^j					nc
$i+1$	2^{j+5}		2^j				
$i+2$	2^j			2^{j+30}			nc, $\Delta f = 2^j$
$i+3$	2^{j+30}				2^{j+30}		nc, $\Delta f = 2^{j+30}$
$i+4$	2^{j+30}					2^{j+30}	nc, $\Delta f = 2^{j+30}$
$i+5$	2^{j+30}						nc

Table 2. A 6-step local collision of SHA-0 starting at step i. The measure of difference is $\oplus$. Addition in the exponents is modulo 32. "nc" stands for no carry. Δf is the output difference of the Boolean function.

helps to offset completely the chaining variable difference in the second step of the local collision, where $x_{i,j}$ $(x = m)$ denotes the j-th bit of message word x_i.

The message condition in round 3

$$m_{i,2} = \neg m_{i+2,2}$$

helps to offset the difference caused by the non-linear function in the third step of the local collision.

3.2 Differential Paths of SHA-0

At a high level, the differential path used in [14] is a sequence of local collisions joined together with possible overlaps. To construct such a path, we need to find a set of appropriate starting step for each local collision. We can use an 80-bit 0-1 vector $x = (x_0, ..., x_{79})$ to specify these starting steps, and the vector is called a *disturbance vector*. It is easy to show that the disturbance vector satisfies the same recursion defined by the message expansion. That is, for $i = 16, ...79$,

$$x_i = x_{i-3} \oplus x_{i-8} \oplus x_{i-14} \oplus x_{i-16}.$$

For the 80 variables x_i, any 16 consecutive ones determine the rest. So there are 16 free variables to be set for a total of 2^{16} possibilities.

In order for the disturbance vector to lead to a possible collision, several conditions on the disturbance vectors need to be imposed, and they are discussed in details in [14]. These conditions are summarized in Table 3.

From [15], we know condition 1 in Table 3 holds if and only if the following equations hold:

$$\begin{aligned}
x_{11} &= x_3 + x_8 \\
x_{12} &= x_4 + x_9 \\
x_{13} &= x_5 + x_{10} \\
x_{14} &= x_0 + x_3 + x_6 + x_8 \\
x_{15} &= x_1 + x_4 + x_7 + x_9
\end{aligned}$$

	Condition	Purpose
1	$x_i = 0$ for $i = 75, 76, 77, 78, 79$	to produce a collision in the last step 5
2	$x_i = 0$ for $i = -5, ..., -1$	to avoid truncated local collisions in first few steps
3	no consecutive ones in the first 17 variables	to avoid an impossible collision path due to a property of IF

Table 3. Conditions on disturbance vectors for SHA-0 with t steps

Condition 2 in Table 3 holds if and only if

$$
\begin{aligned}
x_6 &= x_0 + x_1 + x_2 + x_4 \\
x_7 &= x_0 + x_4 \\
x_8 &= x_0 + x_1 + x_5 \\
x_9 &= x_4 \\
x_{10} &= x_0 + x_5
\end{aligned}
$$

We can also search for a disturbance vector using $(x_0, ..., x_{15})$ as the 16 variables. After imposing Conditions 1 and 2, there are 6 free variables remaining: $(x_0, ..., x_5)$. With Condition 3, only 3 choices are left for the 6 free variables, namely (001000) and (000100) and (000101), the first of which corresponds to the differential path given in [14].

We remark that the Hamming weight of the disturbance vector is closely related to the complexity of the attack. Given a disturbance vector x, we define $hw_{r+}(x)$ as the Hamming weight of x from step r to 80. To minimize the complexity, the Hamming weight $hw_{17+}(x)$ should as small as possible (although there are other more subtle conditions). The corresponding vector used in [14] have $hw_{17+} = 27$, and the complexity of collision search attack is about 2^{58}.

3.3 Existing Techniques for Improving the Attack

In the past year, there have been some major advances in the analysis of SHA-0. These latest attacks are built upon the differential attack by Chaband and Joux, while introducing new ideas for significant improvements. We summarize these techniques below.

- *Neutral bit techniques* [1]. This allows the collision search to start at a step $i > 17$. [5] Biham and Chen showed how to start the collision search of SHA-0 at step $i = 22$ [1] and reduce complexity of finding full collisions to 2^{56}.

[5] Since the first 16 message words are independent, in general one can bypass the first 16 steps and start the search at $i = 17$.

More interestingly, they were able to find near collisions of SHA-0 with complexity 2^{40}, and this provides a basis for finding multi-block collisions. Since a near collision does not require the first set of conditions on the disturbance vector, vectors with much lower Hamming weight can be found.
- Multi-block collision techniques [2, 7, 12]. The idea is to use near collisions in several message blocks to produce a collision. Using this technique together with the neutral bit technique, Joux reported the first real collision of the full 80-step SHA-0. The search complexity is estimated to be 2^{51} hash operations.

4 Message Modification Techniques and SHA-0

At the Rump Session of Crypto'04, Wang [10] announced collisions of several hash functions, including MD4, MD5, RIPEMD, and HAVEL-128. The collision search attacks on these hash functions [11–13] adopt a three-step approach: find a differential path leading to possible collisions, derive a set of sufficient conditions for the differential path to hold, and modify the message words to satisfy all conditions in the first round (as well as most conditions in the second round) so that the success probabilities can be greatly enhanced.

The"message modification"employed in the last step is a major innovation that makes these collision search attacks feasible. Message modification techniques have been introduced in attacking HAVAL-128, MD5, MD4, RIPEMD [11–13] and SHA-0 [15] (not gave the precise description in [15]). However, the more sophisticated hash functions such as SHA-0 and SHA-1 pose considerable new challenges, and require more powerful message modification techniques in their attack. We shall discuss various components of the message modification approach which, when suitably combined, can yield an effective attack. Full details will be omitted in this presentation.

4.1 Message Modification Techniques

In what follows, we provide a description of the more complicated message modification techniques for SHA-0. Following the terminology in [12], we also categorize the techniques into basic techniques and advanced techniques.

For the MD4-family of hash functions, including SHA, the step function F has the form of

$$a_i = F(\text{input chaining variables}, m_{i-1}),$$

where a_i is the output chaining variable and m_{i-1} is the message word applied in step i. Given a differential path that may lead to possible collisions, it is not hard to derive a set of sufficient conditions on a_i. The conditions are of the following forms:

- $a_{i,j} = 0$ or $a_{i,j} = 1$.
- $a_{i,j} = a_{i',j'}$ or $a_{i,j} = \neg a_{i',j'}$, for $i' < i$.

In fact, all these conditions can be combined into one general form:

$$a_{i,j} = v,$$

where v is a bit value that is fixed to be 0/1 or has been computed *before* step i (since $i' < i$). Therefore, we can treat them uniformly.

The main idea of the "'basic modification technique"is simply to set $a_{i,j}$ to the correct bit by modifying the corresponding bit of m_{i-1}. More specifically, the following operation is performed for each derived condition $a_{i,j} = v$.

- If $a_{i,j} \neq v$, then set $m_{i-1} = m_{i-1} \pm 2^{j-1}$ to correct the condition.

If there is a condition on $m_{i-1,j}$ in the collision differential path, the above modification isn't available. So, it needs to modify some message bits in the previous steps (maybe only one message bit of the previous step is modified) to correct the condition of $a_{i,j}$. These message modification techniques can be applied to a hash function up to the first 16 steps.

If the message word m_i is dependent on one or more of the earlier message words, then "advanced modification technique"are needed to deal with the complication. Roughly, a change in m_i will cause a change in m_t for some $t < 16$ and hence a change in a_{t+1}. The advanced technique can "correct" this change within the next few steps during which each of the chaining variables are updated once. Effectively, the correction process is the same as constructing a local collision. The process works if and only if modification of the message words in those steps does not affect any existing conditions on the chaining variables.

4.2 Application to SHA-0

Given a differential path of SHA-0 in any existing attacks, we can easily derive a set of sufficient conditions on the chaining variables by analyzing each local collision separately. Using the basic modification techniques, we can make all the conditions in the first 16 steps to hold in a systematic way.

The advanced modification techniques, however, do not seem to be directly applied to SHA-0 as in the cases of MD4, HAVAL-128 and MD5 etc. The effectiveness largely depends on how the conditions are "distributed" after step 16. For MD5, the conditions are very concentrated in steps 17 and 18, while for SHA-0 the conditions are spread out due to the local collisions. Another reason is the use of message expansion in SHA-0. As a result, the advanced modification techniques in [11–13] can only help to make a few conditions satisfied in steps $i > 16$. This would improve over the original attack on SHA-0 [14]. Given the neutral bit techniques [1] which already allow the bypass of the first 22 steps, the modification techniques, as they were used in MD5, are difficult to offer additional improvements over the best existing attacks on SHA-0. Therefore, new ideas are required in order to launch a more practical attack on SHA-0 and especially for extending the attack to SHA-1.

5 New Techniques for Searching Collisions in SHA-0

In this section, we present our new techniques for collision search in SHA-0. The techniques are quite effective for SHA-0 and can also be extended to attack SHA-1 [16].

5.1 Overview

The first key idea in our new techniques is to remove both Condition 2 and Condition 3 (see Table 3) on the disturbance vectors. Such relaxation provides a larger search space and allows us to find disturbance vectors whose hamming weights are much lower than those used in existing attacks, thereby greatly decreasing the complexity of the attack.

The cost, though, is much more complicated differential paths in the first round. In particular, the disturbance vector consists of consecutive ones in the first 16 steps as well as truncated local collision. We introduce several new techniques to construct a valid differential path given such a disturbance vector. This is the most difficult yet crucial part of the new analysis, without which it would be impossible to produce a *real* collision.

We also present a variation of the basic modification technique to deal with conditions in steps 17 through 20, effectively starting the collision search at step 21. Combining all these new techniques and some simple implementation tricks, we are able to reduce the collision search complexity of SHA-0 to below 2^{40}.

5.2 Finding Disturbance Vectors with Low Hamming Weight

In existing attacks, the difficulty of finding disturbance vectors of low Hamming weights is largely due to the following difference between the IF and XOR function: when c and d both change, the output of IF always changes, while the output of XOR never changes. For MAJ, the output changes with probability 1/2. This motivates us to treat the first round differently so that Condition 3 can be relaxed.

We only impose Condition 1 in the search for good disturbance vectors. By doing so, we obtain many vectors with very small Hamming weights. Since we can use modification techniques to make all conditions in the first round to hold, we focus on vectors with small Hamming weight in rounds 2-4. Among the 2^{16} choices that satisfy Condition 1, about 30 of them have $17 \leq hw_{21+} \leq 19$, and four of which have Hamming weight 3 in the third round. We then picked the following one from the four candidates as the disturbance vector.

5.3 New Analysis Techniques

As we can see from the chosen vector, there are four consecutive 1s in the first 16 steps. In addition, there are three truncated local collisions since $x_{-3} = x_{-2} =$

step	vector
-5...-1	0 0 1 1 1
1...20	0 1 1 1 1 0 0 1 0 1 0 0 1 0 1 0 1 0 0 0
21...40	0 1 1 0 0 0 1 0 0 0 1 0 1 1 1 0 0 0 0 0
41...60	0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0
61...80	1 1 0 0 1 1 0 1 0 1 1 1 0 1 0 0 0 0 0 0

Table 4. A disturbance vector for producing a collision of SHA-0

$x_{-1} = 1$. The corresponding message differences for the first 16 steps are given in Table 8 in the appendix. The most difficult part is to derive a differential path for the first 16 steps given the irregular message difference.

There are several techniques that we used to construct a valid differential path. Before diving into the details, we first present a few general ideas.

- Use "subtraction" instead of "exclusive-or" as the measure of difference to facilitate the precision of the analysis.
- Take advantage of special differential properties of IF. In particular, if there is a bit difference in one of the three inputs, the output will have a difference with probability 1/2. In addition, when the bit does flip, it can maintain or change the sign of the difference. Therefore, the function can either *preserve* or *absorb* an input difference, giving good flexibility for constructing differential paths.
- Take advantage of the carry effect. Since $2^j = -2^j - 2^{j+1} ... - 2^{j+k-1} + 2^{j+k}$ for any k, a single bit difference j can be expanded into several bits. This property makes it possible to introduce extra bit differences. To use the idea in a more sophisticated way, we can combine two sets of differences to produce one difference.
- Regroup the message differences. Some differences in local collisions shall remain unchanged to guarantee that the local collisions hold. Some other differences in a local collision will be reset to cancel out certain changed chaining variable bits – especially those bits produced by the message differences in the truncated collisions, and those arising from two consecutive local collisions.

5.4 Constructing the Specific Path

We first introduce some notation. Let $a_{i,j}$ denote the jth bit of variable a_i and $\Delta a_i = a'_i - a_i$ denote the difference. Note that we use *subtraction* difference rather than *exclusive-or* difference since keeping track of the signs is important in the analysis. Following the notation introduced in [11–13], we use $a_i[j]$ to denote $a_i[j] = a_i + 2^{j-1}$ with no bit carry, and $a_i[-j]$ to denote those $a_i[-j] = a_i - 2^{j-1}$ with no bit carry.

To construct a valid differential path, it is important to control the propagation of the differences in each chaining variables. At a high level, differences in

b, c, d are mostly absorbed by the Boolean function IF. The differences in a and e need to be carefully controlled, and most of them are offset by using appropriate differences in b, c, d.

The complete differential path for the first 16 steps is given in Table 8 in the appendix. It may look quite complicated at a first glance, and so we provide a more concise description below which better illustrates the idea. Based on the step function of SHA-0, it is not hard to see that the differences in the chaining values are fully determined by the differences in a, which is given in Table 5 below. Bit differences that are expanded using the carry effect is shown in bold, and the expanded bits are not shown. The bit differences in a can be categorized into four groups as follows, and their rationale can then be better understood.

Δa	I	II	III	IV
a_1	$-2, \mathbf{7}, -32$			
a_2	$\mathbf{-7}, \mathbf{12}, \mathbf{-5}$			
a_3	$-12, \mathbf{17}, -10$	2		
a_4			**20**	9
a_5			25	**4**
a_6		2	$\mathbf{-10}, \mathbf{15}$	
a_7		2	-17	
a_8			$\mathbf{-12}$	
a_{10}		2		
a_{11}			10	
a_{13}		2		
a_{15}		-2		

Table 5. Differences in a. The entries list the bit positions of the differences and their signs. For example, the difference 2^j is listed as $j+1$ and -2^j as $-(j+1)$. Bit positions in bold are expanded using the carry effect in the complete differential path given in Table 8.

- Group I: differences due to Δm_0:
  These are message differences due to the "truncated" local collisions. Hence they are inherent from the chosen path and cannot be changed. They cause differences in e_5, e_6, e_7 that need to be cancelled. Most of them can be cancelled with existing differences in that step, except $e_5[-30]$, $e_6[-5]$,and $e_7[15]$.
- Group II: differences due to disturbance.
  These result in the usual 6-step local collisions.
- Group III: differences introduced to cancel $e_5[-30]$ and $e_7[15]$.
  Note that only $a_6[15]$ is for cancelling $e_7[15]$, and the rest are all for cancelling $e_5[30]$. This part is where the expansion using the carry effect is needed.
- Group IV: differences used for additional adjustments.
  These are $a_4[9]$ for producing $e_8[7]$ in order to cancel $m_9[-7]$ and $a_5[4]$ for both producing $e_9[2]$ to introduce the disturbance equivalent to $m_{10}[2]$ and producing $b_6[5]$ to cancel out $e_6[-5]$.

This is the most difficult yet crucial part of the new analysis, without which it would be impossible to produce a *real* collision. Furthermore, the analysis demonstrates some unexpected weaknesses in the design of the step update function. In particular, certain properties of Boolean function $(x \wedge y) \vee (\neg x \wedge z)$ and the carry effect actually facilitate, rather than prevent, differential attacks.

5.5 Deriving Conditions on m_i and a_i

As we discussed in Section 3, for each local collision starting at step i, the follow conditions on m should hold.

$$m_{i+1,7} = \neg m_{i,2} \tag{1}$$

$$m_{i+2,2} = \neg m_{i,2} \quad (For\ round\ 3) \tag{2}$$

The condition on the disturbance vector given in Table 4, there are a total of 19 disturbances in Rounds 2, 3 and 4. So equation (1) yields 19 conditions on message words. From 3 disturbances in round 3, there are another 3 conditions on message word corresponding to equation (2). From Table 8, there are another 9 necessary conditions on message word position bit 2 and 7. There are total 31 conditions on message positions 2 and 7, and by a straightforward search of the 2^{32} choices for two positions of $m_{0,2}, ..., m_{15,2}$ it turns out that several choices satisfy all the conditions.

After the conditions on the message words are determined, we can derive a set of sufficient conditions on a_i given the differential path. The derivation uses differential properties of the three Boolean functions as well as the carry propagation pattern of addition. The complete description of the conditions is given in Table A in the appendix.

5.6 A Variation of the Modification Techniques

There are a total of 45 conditions from step 17 to step 80. Here we introduce a variation of the message modification techniques to deal with the three conditions in step 17 through 20, and hence reducing the number of conditions to 42. The idea is better explained using an example, say the condition on $a_{17,32}$. Instead of modifying m_{16}, which is dependent on four earlier message words, we modify m_{15} in a way that will flip the bit $a_{16,27}$, which in turn flips the bit $a_{17,32}$ in step 17. The other two conditions are handled similarly.

5.7 Complexity Analysis

In this section, we analyze the complexity of our collision search attack. Since there are a total of 42 conditions after applying message modification, a straightforward implementation would yield a complexity of 2^{42} hash operations.

There are several simple techniques that we can use to further improve the efficiency of the attack. The idea is that we only need to compute a small number of steps of the 80-step hash operation. First, we can precompute a set of "good" message words that make all conditions satisfied in the first 14 steps and only leave the last two message words as free variables. Second, we can use an "early stopping technique". More specifically, we only need to carry out the computation until step 23 and then test whether the four conditions in steps 21 through 24 are satisfied. On average only a fraction of 2^{-4} of the messages will pass the test. Overall, we only need to compute from step 15 to 24, for a total of 10 steps. This immediately gives a factor of $80/10 = 8$ improvements in search complexity. Hence, the complexity of finding a full collision is at most 2^{39} hash operations.

Our analysis can also be used to find near collisions with much lower complexity. For near collisions, we have found quite a few disturbance vectors with $hw_{21+} = 14$, and an example is given in Table 6. For this vector, the total number of conditions in Rounds 2-4 is $(14-4) \times 2 + 4 \times 4 = 36$. Using early stopping techniques, we estimate that near collisions of SHA-0 can be found with complexity about 2^{33}.

step	vector
1...20	1 0 0 1 1 0 1 1 1 1 1 0 1 0 1 1 0 1 1 1
21...40	0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0
41...60	0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0
61...80	1 0 0 1 0 1 1 0 0 0 0 0 0 1 0 0 0 1 0 0

Table 6. A disturbance vector for near collision of SHA-0.

Finally, we remark that using the multi-block technique for attacking MD5 [12], we can use near collisions to construct multi-block collisions with about the same complexity. Therefore, we expect multi-block collisions of SHA-0 can potentially be found with about 2^{33} hash operations.

6 A Collision Example of SHA-0

The two messages that collide are (M_0, M_1) and (M_0, M_1'), where

$$h_1 = compress(h_0, M_0)$$
$$h_2 = compress(h_1, M_1) = compress(h_1, M_1')$$

Note that the first message block M_0 is the same, and it is for producing an intermediate chaining value h_1 that satisfies the 14 conditions on a_0, b_0. (See Table A). M_0 can be found with complexity 2^{14}. After that, the pair (M_1, M_1') can be found with complexity 2^{39}.

We remark that we can adjust the differential path under the conditions of the original initial value h_0 to find a one-block message collision differential path.

h_0:	67452301	efcdab89	98badcfe	10325476	c3d2e1f0			
M_0:	65c24f5c	0c0f89f6	d478de77	ef255245	83ae3a1f	2a96e508	2c52666a	0d6fad5a
	9d9f90d9	eb82281e	218239eb	34e1fbc7	5c84d024	f7ad1c2f	d41d1a14	3b75dc18
h_1:	39f3bd80	c38bf492	fed57468	ed70c750	c521033b			
M_1:	474204bb	3b30a3ff	f17e9b08	3ffa0874	6b26377a	18abdc01	d320eb93	b341ebe9
	13480f5c	ca5d3aa6	b9f3bd88	21921a2d	4085fca1	eb65e659	51ac570c	54e8aae5
M'_1:	c74204f9	3b30a3ff	717e9b4a	3ffa0834	6b26373a	18abdc43	5320eb91	3341ebeb
	13480f1c	4a5d3aa6	39f3bdc8	a1921a2f	4085fca3	6b65e619	d1ac570c	d4e8aaa5
h_2:	2af8aee6	ed1e8411	62c2f3f7	3761d197	0437669d			

Table 7. A collision of 80-step SHA-0. Padding rules are not applied to the input messages

7 Conclusions

In this paper, we present a new collision search attack on SHA-0 with complexity 2^{39} hash operations. Compared with existing attacks on SHA-0, our method is much more efficient and real collisions can be found quickly on a typical PC.

The techniques developed in our analysis of SHA-0 are also applicable to SHA-1. As SHA-0 may be viewed as simpler variant of SHA-1, the analysis presented here serves to verify effectiveness of these new techniques for other SHA variants.

Our analysis demonstrates some weaknesses in the step updating function of SHA-0 and SHA-1. In particular, because of the simple step operation structure, certain properties of the Boolean function $(x \wedge y) \vee (\neg x \wedge z)$ combined with the carry effect actually facilitate, rather than inhibit, differential attacks. We hope that these insights can be useful in the design of more secure hash functions in the future.

Acknowledgements

It is a pleasure to acknowledge Arjen K. Lenstra for his important suggestions and corrections, and for spending his precious time on our research. We would like to thank Andrew C. Yao and Frances. Yao for their support and corrections on this paper. We also thank Ronald L. Rivest and many other anonymous reviewers for their important comments.

References

1. E. Biham and R. Chen. *Near Collisions of SHA-0.* Advances in Cryptology – Crypto'04, pp.290-305, Springer-Verlag, August 2004.
2. E. Biham and R. Chen. *New Results on SHA-0 and SHA-1.* Crypto'04 Rump Session, August 2004.

3. E. Biham, R. Chen, A. Joux, P. Carribault, W. Jalby and C. Lemuet. *Collisions in SHA-0 and Reduced SHA-1.* Advances in Cryptology–Eurocrypt'05, pp.36-57, May 2005.
4. NIST. *Secure hash standard.* Federal Information Processing Standard, FIPS-180, May 1993.
5. NIST. *Secure hash standard.* Federal Information Processing Standard, FIPS-180-1, April 1995.
6. F. Chabaud and A. Joux. *Differential Collisions in SHA-0.* Advances in Cryptology – Crypto'98, pp.56-71, Springer-Verlag, August 1998.
7. A. Joux. *Collisions for SHA-0.* Rump session of Crypto'04, August 2004.
8. K. Matusiewicz and J. Pieprzyk. *Finding Good Differential Patterns for Attacks on SHA-1.* IACR Eprint archive, December 2004.
9. V. Rijmen and E. Osward. *Update on SHA-1.* RSA Crypto Track 2005, February 2005.
10. X. Y. Wang, D. G. Feng, X. J. Lai, and H. B. Yu. *Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD.* In Rump session of Crypto'04 and IACR Eprint archive, August 2004.
11. X. Y. Wang, D. G. Feng, X. Y. Yu. The Collision Attack on Hash Function HAVAL-128. In Chinese, Science in China, Series E, Vol. 35(4), pp.405-416, April, 2005.
12. X. Y. Wang and H. B. Yu. *How to Break MD5 and Other Hash Functions.* Advances in Cryptology–Eurocrypt'05, pp.19-35, Springer-Verlag, May 2005.
13. X. Y. Wang. X. J. Lai, D. G. Feng, H. Chen, X. Y. Yu. *Cryptanalysis for Hash Functions MD4 and RIPEMD.* Advances in Cryptology–Eurocrypt'05, pp.1-18, Springer-Verlag, May 2005.
14. X. Y. Wang. *The Collision attack on SHA-0*, In Chinese, to appear on www.infosec.edu.cn, 1997.
15. X. Y. Wang. *The Improved Collision attack on SHA-0.* In Chinese, to appear on www.infosec.edu.cn, 1998.
16. X. Y. Wang. Y. Lisa Yin, H. B. Yu. *Finding Collisions in the Full SHA-1.* These proceedings.

A The differential path and derived conditions

In Table 8 we describe the details of the differential path that leads to a full collision of SHA-0. In Table A, we list a set of sufficient conditions on the chaining variables a_i for the given differential path.

For the first 20 steps, since there are many conditions for each a_i, we use a compact representation for the conditions so that they can be easily visualized. More specifically, for the condition $a_{i,j} = v$ we put one symbol `w` in the row for a_i under bit position j, where `w` is defined as follows:

- If $v = 0$, then `w = 0`.
- If $v = 1$, then `w = 1`.
- If $v = a_{i-1,j}$, then `w = a`.
- If $v = \neg a_{i-1,j}$, then w = $\overline{a}$.
- If no condition on $a_{i,j}$, then `w = -`.

step i	x_{i-1}	Δm_{i-1}	Δa_i	Δb_i	Δc_i	Δd_i	Δe_i
1	0	$-2, 7, 32$	-2 $-7, -8, 9$ -32				
2	1		$7, 8, -9$ $-12, ..., -21, 22$ $5, -6$	Δa_1			
3	1	$2, 7, 32$	-12 $-17, -18, 19$ -10 2	Δa_2	$\Delta a_1^{\lll 30}$		
4	1	-7	$-20, ..., -24, 25$ 9	...	$\Delta a_2^{\lll 30}$	$\Delta a_1^{\lll 30}$	
5	1	-7	25 $-4, 5$		...	$\Delta a_2^{\lll 30}$	$\Delta a_1^{\lll 30}$
6	0	$2, 7$	2 $10, 11, -12$ $-15, 16$			...	$\Delta a_2^{\lll 30}$
7	0	$-2, 32$	2 -17				...
8	1	$2, 32$	$12, ..., 22, -23$	...			
9	0	-7			...		
10	1	32	2			...	
11	0	$7, 32$	10	...			...
12	0	$2, 32$		Δa_{11}	...		
13	1	2	2		$\Delta a_{11}^{\lll 30}$	...	
14	0	$-7, 32$		Δa_{13}		$\Delta a_{11}^{\lll 30}$	...
15	1	32	-2		$\Delta a_{13}^{\lll 30}$		$\Delta a_{11}^{\lll 30}$
16	0	$-7, 32$		Δa_{15}		$\Delta a_{13}^{\lll 30}$	

Table 8. A differential path for the first round of SHA-0. For ease of notation, the entries list the bit positions of the differences and their signs. For example, the difference 2^j is listed as $(j + 1)$ and -2^j as $-(j + 1)$.

The rest of the path consists of 19 6-step local collisions. The starting step of these collisions is specified by the disturbance vector given in Table 4.

chaining variable	conditions on bits			
	$32-25$	$24-17$	$16-9$	$8-1$
a_0	--------	1-1100-1	--1--1-1	10------
b_0	--------	--------	------a-	----a-$\overline{a}$-
a_1	1-------	0a0011a0	aa1-10a0	11----1-
a_2	0-0-----	--011111	111111-1	0010a---
a_3	1-1--aaa	aa0-0011	0010101-	-010100-
a_4	1----a-0	11111000	--1111-0	110011--
a_5	0------0	-0001001	00100-0-	01-01---
a_6	-------0	-1011110	010-100-	-0--100-
a_7	0------1	a1011111	0100--00	0----10-
a_8	--------	-1000000	00000-11	1---1---
a_9	1-------	---00000	0011001-	----0---
a_{10}	--------	---11111	1111111-	------0-
a_{11}	0-------	--------	------0-	----1---
a_{12}	0-------	--------	--------	0---0---
a_{13}	--------	--------	--------	1---0-0-
a_{14}	1-------	--------	--------	----1---
a_{15}	0-------	--------	--------	----1-1-
a_{16}	1-------	--------	--------	----0---
a_{17}	0-------	--------	--------	------1-
a_{18}	1-------	--------	--------	--------
a_{19}	--------	--------	--------	--------
a_{20}	--------	--------	--------	--------

Table 9. A set of sufficient conditions on a_i for the differential path given in Table 8.

The conditions for the 19 local collisions in Rounds 2-4 are derived as follows. Note that the conditions depend on the bit $m_{i,2}$ which has been pre-determined.

- XOR rounds:

$$\begin{aligned} a_{i-1,4} &= \neg a_{i-2,4} \ (or\ a_{i-1,4} = a_{i-2,4}) \\ a_{i,2} &= m_{i,2}. \end{aligned}$$

- MAJ round:

$$\begin{aligned} a_{i-1,4} &= \neg a_{i-2,4} \\ a_{i,2} &= m_{i,2} \\ a_{i+1,32} &= \neg a_{i-1,2} \\ a_{i+2,32} &= \neg a_{i+1,2} \end{aligned}$$