Finding Collisions in the Full SHA-1

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

[1] Shandong University, Jinan 250100, China, xywang@sdu.edu.cn
[2] Independent Security Consultant, Greenwich CT, US, yyin@princeton.edu
[3] Shandong University, Jinan250100, China, yhb@mail.sdu.edu.cn

Abstract. In this paper, we present new collision search attacks on the hash function SHA-1. We show that collisions of SHA-1 can be found with complexity less than 2^{69} hash operations. This is the first attack on the full 80-step SHA-1 with complexity less than the 2^{80} theoretical bound.

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

1 Introduction

The hash function SHA-1 was issued by NIST in 1995 as a Federal Information Processing Standard [5]. Since its publication, SHA-1 has been adopted by many government and industry security standards, in particular standards on digital signatures for which a collision-resistant hash function is required. In addition to its usage in digital signatures, SHA-1 has also been deployed as an important component in various cryptographic schemes and protocols, such as user authentication, key agreement, and pseudorandom number generation. Consequently, SHA-1 has been widely implemented in almost all commercial security systems and products.

In this paper, we present new collision search attacks on SHA-1. We introduce a set of strategies and corresponding techniques that can be used to remove some major obstacles in collision search for SHA-1. Firstly, we look for a near-collision differential path which has low Hamming weight in the "disturbance vector" where each 1-bit represents a 6-step local collision. Secondly, we suitably adjust the differential path in the first round to another possible differential path so as to avoid impossible consecutive local collisions and truncated local collisions. Thirdly, we transform two one-block near-collision differential paths into a two-block collision differential path with twice the search complexity. We show that, by combining these techniques, collisions of SHA-1 can be found with complexity less than 2^{69} hash operations. This is the first attack on the *full* 80-step SHA-1 with complexity less than the 2^{80} theoretical bound.

In the past few years, there have been significant research advances in the analysis of hash functions. The techniques developed in these early works provide an important foundation for the attacks on SHA-1 presented in this paper. In particular, our analysis is built upon the original differential attack on

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

SHA-0 [14], the near collision attack on SHA-0 [1], the multi-block collision techniques [12], as well as the message modification techniques used in the collision search attacks on HAVAL-128, MD4, RIPEMD and MD5 [11, 13, 12].

Our attack naturally is applied to SHA-0 and all reduced variants of SHA-1. For SHA-0, the attack is so effective that we are able to find real collisions of the full SHA-0 with less than 2^{39} hash operations [16]. We also implemented the attack on SHA-1 reduced to 58 steps and found real collisions with less than 2^{33} hash operations. In a way, the 58-step SHA-1 serve as a simpler variant of the full 80-step SHA-1 which help us to *verify* the effectiveness of our new techniques. Furthermore, our analysis shows that the collision complexity of SHA-1 reduced to 70 steps is less than 2^{50} hash operations.

The rest of the paper is organized as follows. In Section 2, we give a description of SHA-1. In Section 3, we provide an overview of previous work on SHA-0 and SHA-1. In Section 4, we present the techniques used in our new collision search attacks on SHA-1. In Section 5, we elaborate on the analysis details using the real collision of 58-step SHA-1 as a concrete example. We discuss the implication of the results in Section 6.

2 Description of SHA-1

The hash function SHA-1 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 Damgard/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-1.

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 $i = 16, ..., 79$,

$$m_i = (m_{i-3} \oplus m_{i-8} \oplus m_{i-14} \oplus m_{i-16}) \lll 1.$$

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

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

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

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.

round	step	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-1

3 Previous Work on SHA-0 and SHA-1

In 1997, Wang [14] presented the first attack on SHA-0 based on an algebraic method, and showed that collisions can be found with complexity 2^{58}. In 1998 Chabaud and Joux independently found the same collision differential path for SHA-0 by the differential attack. In the present work, as well as in the SHA-0 attack by [16], the algebraic method (see also Wang [15]) again plays an important role, as it is used to deduce message conditions both on SHA-0 and SHA-1 that should hold for a collision (or near-collision) differential path and be handled in advance.

3.1 Local Collisions of SHA-1

Informally, a local collision is a collision within a few steps of the hash function. A simple yet very important observation made in [14] is that SHA-0 has a 6-step local collision that can start at any step i. A kind of local collision can be referred to [16], and the chaining variable conditions for a local collision were taken from Wang [14].

The collision differential path on SHA-0 chooses $j = 2$ so that $j + 30 = 32$ becomes the MSB [4] to eliminate the carry effect in the last three steps. In addition, the following condition

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

helps to offset completely the chaining variable difference in the second step of the local collision, where $m_{i,j}$ denotes the j-th bit of message word m_i.

[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.

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.

Since the local collision of SHA-0 does not depend on the message expansion, it also applies to SHA-1. Hence, this type of local collision can be used as the basic component in constructing collisions and near collisions of the full 80-step SHA-0 and SHA-1.

3.2 Differential Paths of SHA-1

We start with the differential path for SHA-0 given in [14, 15]. At a high level, the path is a sequence of local collisions joined together. To construct such a path, we need to find appropriate starting steps for the local collisions. They can be specified by an 80-bit 0-1 vector $x = (x_0, ..., x_{79})$ called a *disturbance vector*. It is easy to show that the disturbance vector satisfies the same recursion defined by the message expansion.

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. Then a "good" vector satisfying certain conditions can be easily searched with complexity 2^{16}.

In [2, 9], the method for constructing differential paths of SHA-0 is naturally extended to SHA-1. In the case of SHA-1, each entry x_i in the disturbance vector is a 32-bit word, rather than a single bit. The vectors thus defined satisfy the SHA-1 message expansion.

That is, for $i = 16, ..., 79$,

$$x_i = (x_{i-3} \oplus x_{i-8} \oplus x_{i-14} \oplus x_{i-16}) \lll 1.$$

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 [15] [6]. These conditions also extend to SHA-1 in a straightforward way, and we summarize them in Table 2.

	Condition	Purpose
1	$x_i = 0$ for $i = t-5, ..., t-1$	to produce a collision in the last step t
2	$x_i = 0$ for $i = -5, ..., -1$	to avoid truncated local collisions in first few steps
3	no consecutive ones in same bit position in the first 16 variables	to avoid an impossible collision path due to a property of IF

Table 2. Conditions on disturbance vectors for SHA-1 with t steps

In the case of SHA-0, 3 vectors are found among the 2^{16} choices, and two of them are valid when all three conditions are imposed.

In the case of SHA-1, it becomes more complicated to find a good disturbance vector with low Hamming weight due to large search space. Biham and Chen [2] used clever heuristics to search for such vectors for reduced step variants and they were able to find real collisions of SHA-1 up to 40 steps. They estimated that collisions of SHA-1 can be found up to 53-round reduced SHA-1 with about 2^{48} complexity, where the reduction is to the last 53 rounds of SHA-1. Rijmen and Osward [9] did a more comprehensive search using methods from coding theory, and their estimates on the complexity are similar.

Overall, since the Hamming weight of a valid disturbance vector grows quickly as the number of steps increases, it seems that finding a collision of the full 80-step SHA-1 is beyond the 2^{80} theoretical bound with existing techniques.

4 New Collision Search Attacks on SHA-1

In this section, we present our new techniques for search collisions in SHA-1. The techniques used in the attack on SHA-1 are largely built upon our new analysis of SHA-0 [16], in which we showed how to greatly reduces the search complexity to below the 2^{40} bound.

4.1 Overview

As we have seen in existing analysis of SHA-1, finding a disturbance vector with low Hamming weight is a necessary step in constructing valid differential paths that can lead to collision. On the other hand, the three conditions imposed on disturbance vectors seem to a major obstacle. There have been attempts to remove some of the conditions. For example, finding multi-block collisions using near collisions effectively relax the first condition, and finding collisions for SHA-1 without the first round effectively relax the second condition (although it is no longer SHA-1 itself). Even with both relaxation, the Hamming weight of the disturbance vectors is still too high to be useful for the full 80-step SHA-1.

A key idea of our new attack is to relax *all* the conditions on the disturbance vectors. In other words, we impose *no* condition on the vectors other than they satisfy the message expansion recursion. T his allows us to find disturbance vectors whose Hamming weights are much lower than those used in existing attacks.

We then present several new techniques for constructing a valid differential path given such disturbance vectors. The resulting path is very complex in the first round due to consecutive disturbances as well as truncated local collisions that initiate from steps -5 through -1. This is the most difficult yet crucial part of new analysis, without which it would be impossible to produce a *real* collision.

Once a valid differential path is constructed, we apply the message modification techniques, first introduced by Wang et. al in breaking MD5 and other hash functions [15, 11–13], to further reduce the search complexity. Such extension requires carefully deriving the exact conditions on the message words and chaining variables, which is much more involved in the case of SHA-1 compared with SHA-0 and other hash functions.

Besides the above techniques, we also introduce some new methods that are tailored to the SHA-1 message expansion. Combining all these techniques and a simple "early stopping" trick when implementing the search, we are able to present an attack on SHA-1 with complexity less than 2^{69}. These techniques are presented in more detail in Sections 4 and 5.

4.2 Finding Disturbance Vectors with Low Hamming Weight

Finding good disturbance vectors is the first important step in our analysis. Without imposing any conditions other than the message expansion recursion, the search becomes somewhat easier. However, since there are 16 32-bit free variables, the search space can be as large as 2^{512}. Instead of searching the entire space for a vector with minimum weight, we use heuristics to confine our search within a subspace that most likely contains good vectors.

We note that the 80 disturbance vectors $x_0, ..., x_{79}$ can be viewed as an 80-by-32 matrix where each entry is a single 0/1 bit. A simple observation is that for a matrix with low hamming weight, the non-zero entries are likely to concentrate in several consecutive columns of the matrix. Hence, we can first pick two entries $x_{i,j-1}$ and $x_{i,j}$ in the matrix and let two 16-bit columns starting at $x_{i,j-1}$ and $x_{i,j}$ to vary through all 2^{32} possibilities. There are 64 choices for i ($i = 0, 1, ..., 63$) and 32 choices for j ($j = 1, 2, ..., 32$). In fact, with the same i, different choices of j produce disturbance vectors that are rotations of each other, which would have the same Hamming weight. By setting $j = 2$, we can minimize the carry effect as discussed in Section 3.1. Overall, the size of the search space is at most $64 \times 2^{32} = 2^{38}$.

Using the above strategy, we first search for the best vectors predicting *one-block collisions*. For the full SHA-1, the best one is obtained by setting $x_{64,2} = 1$ and $x_{i,2} = 0$ for $i = 65, .., 79$. The resulting disturbance vector is given in Table 5. The best disturbance vectors for SHA-1 reduced to t-step is the same one with the first $80 - t$ vectors omitted. For SHA-1 variants up to 75 steps, the Hamming weight is still small enough up to allow an attack with complexity less than 2^{80}, and Table 7 summarizes the results for these variants.

In order to break the 2^{80} barrier for the full SHA-1, we continue to search for good disturbance vectors that predict *near collisions* and *two-block collisions*. To do so, we compute more vectors after step 80 using the same SHA-1 message expansion formula (also listed in Table 5).

Then we search all possible 80-vector intervals $[x_i, ..., x_{i+79}]$. *Any* set of 80 vectors with small enough Hamming weight can be used for constructing a near collision. In fact, we found a total of 12 good sets of vectors, and this gives us some freedom to pick the one that achieves the best complexity when taking into account other criteria and techniques (other than just the Hamming weight).

	Existing results				Our new results	
	SHA-1		SHA-1 w/o Round 1		SHA-1	
	conditions		conditions		conditions	
	1,2,3	2,3	1,2	2	1	-
step	1BC	NC,2BC	1BC	NC,2BC	1BC	NC,2BC
47	**26**	12	24	12	5	5
53	42	20	**16**	16	10	7
54	39	**24**	36	16	10	7
60					14	11
70					14	17
75					**26**	21
80					31	**25**

Table 3. Hamming weights (for Rounds 2-4) of best disturbance vectors for SHA-1 variants found by experiments. The comparison is made among different subsets of conditions listed in Table 2. The notation 1BC denotes one-block collision, 2BC is two-block collision, and NC implies near collision

Finally, we compare the minimal Hamming weight of disturbance vectors found by experiments when different conditions are imposed. In Table 3, the last two columns are obtained from our new analysis and other data are from [2]. Provided that the average probability in 2-4 rounds is 2^{-3}, a valid disturbance vector should have a Hamming weight less than a threshold 27, because the corresponding collision (or near-collision) differential has the probability higher than 2^{-80} which can result in an attack faster than the 2^{80} theoretical bound. In the table, we mark the step in bold for which this threshold is reached. It is now easy to see that removing all the conditions has a significant effect in reducing the Hamming weight of the disturbance vectors.

4.3 Techniques for Constructing Differential Paths

In this section, we present our new techniques for constructing a differential path given a disturbance vector with low Hamming weight. Since the vector no longer satisfies the seemly required conditions listed in Table 2, constructing a valid differential path that leads to collisions becomes more difficult. Indeed, this is the most complicated part of our new attacks on SHA-1. It is also a crucial part of the analysis, since without a concrete differential path, we would not be able to search for *real* collisions.

Below, we describe the high-level ideas in these new analysis techniques.

- 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, when an input difference is 1, the output difference can be $1, -1$ or 0. Hence, the function can *preserve*, *flip* or *absorb* an input difference, giving good flexibility for constructing differential paths.

- Take advantage of the carry effect. Since $2^j = -2^j - 2^{j+1} \ldots - 2^{j+k-1} + 2^{j+k}$ for any k, a single bit difference j can be expanded into several bits. This property makes it possible to introduce extra bit differences.
- Use different message differences for the 6-step local collision. For example, $(2^j, 2^{j+5}, 0, 0, 0, 2^{j+30})$ is a valid message differences for a local collision in the first round.
- Introduce extra bit differences to produce the impossible bit-differences in the consecutive local collisions corresponding to the consecutive disturbances in the first 16 steps, or to offset the bit differences of chaining variables produced by truncated local collisions.

A near-collision differential path for the first message block is given in Table 11.

4.4 Deriving Conditions

Given a valid differential path for SHA-1 or its reduced variants, we are ready to derive conditions on messages and chaining variables. The derivation method was originally introduced in [14] for breaking SHA-0, and can be applied to SHA-1 since SHA-0 and SHA-1 have the same step update function. Most details can be found in our analysis of SHA-0 [16], and hence are omitted. Here we focus on the differences between SHA-0 and SHA-1 and discuss a new technique that is tailored to SHA-1.

Due to the extra shift operation in the message expansion of SHA-1, a disturbance can occur in bit positions other than bit 2 of the message words (as can be seen from Table 5), while for SHA-0, all disturbances initiate in bit 2. If this happens in the XOR rounds (round 2 and 4), the number of conditions will increase from 2 to 4 for each local collision. This can blow up the total number of conditions if not handled properly.

We describe a useful technique for utilizing two sets of message differences corresponding to two consecutive disturbances within the *same* step i to produce one 6-step local collision. For example, if there is a disturbance in both bit 1 and bit 2 of x_i, we can set the signs of the message differences Δm_i to be opposite in those two bits. This way, the actual message difference can be regarded as one difference bit in position 1, since $2^1 - 2^0 = 2^0$. Hence the number of conditions can be reduced from $4 + 2 = 6$ to 4.

The conditions for the near-collision path in Table 11 are given in Table 12.

4.5 Message Modification Techniques

Using the basic message modification techniques in [11–13], we can modify an input message so that all conditions on the chaining variables can hold in the first 16 steps. With some additional effort, we can modify the messages so that all conditions in step 17 to 22 also hold.

Note that message modification should keep all the message conditions to hold in order to satisfy the differential path. All the message conditions can be expressed as equations of bit variables in $m_0, m_1, ...m_{15}$ (message words before message expansion). Because of the 1-bit shift in message recursion, all the equations aren't contradictory. Suppose we would like to correct 10 conditions from step 17 to 22 by modifying the last 6 message words $m_{10}, m_{11}, ...m_{15}$. From Table 12, we know there are 32 chaining variable conditions, together with total 47 message equations from step 11 to step 16, the total number of conditions is 79 in step 11-16. Intuitively, this leaves a message space of size 2^{113}, which is large enough for modifying some message bits to correct 10 conditions.

4.6 Picking the Best Disturbance Vector

Once the conditions are derived and message modifications are applied, we can analyze the complexity in a very precise way, by counting the remaining number of conditions in Rounds 2 to 4. The counting rules depend on the Boolean function and locations of the disturbances occur in each round, and local collisions across boundaries of rounds need to be handled differently. The details are summarized in Table 8 in the appendix.

Given the disturbance vectors in Table 5, we find that for an 80-step near collision, the minimum Hamming weight is 25 using the 80 vectors with index [15,94]. However, the *minimum number of conditions* is 71 using the 80 vectors with index [17,96]. This is because the conditions in step 79 and 80 can be ignored for the purpose of near collisions, and the condition in step 21 can be made to hold (see Section 4.5). The step-by-step counting for the number of conditions for this vector is given in Table 9.

Using minimum number of conditions as the selection criteria, we pick the vectors with index [17,96] as the disturbance vectors for constructing an 80-step near collision.

4.7 Using Near Collisions to Find Collisions

Using the idea of multi-block collisions in [7, 2, 3, 12], we can construct two-block collisions using near collisions. For MD5 [12], the complexity of finding the first block near-collision is higher than those of the second block near-collision because of the determination for the bit-difference positions and signs in the last several steps. Here we show that by keeping the bit-difference positions and the signs as free variables in the last two steps, we can maintain essentially twice the search complexity while moving from near collisions to two-block collisions. This idea is also applicable to MD5 to further improve its collision probability from 2^{-37} to 2^{-32}.

Let M_0 and M_0' be the two message blocks and $\Delta h_1 = h_1' - h_1$ be the output difference for the 80-step near collision. If we look closely at the disturbance vectors that we have chosen, there are 4 disturbances in the last 5 steps that will propagate to Δh_1, which become the input differences in the initial values for the second message block.

There are two techniques that we use to construct the differential path for the second message blocks M_1 and M_1'. First, we apply the techniques described in

Section 4.3 so that Δh_1 can be "absorbed" in the first 16 steps of the differential paths. Second, we set the conditions on M_1 so that the output difference Δh_2 will have opposite signs for each of the differences in Δh_1. In other words, we set the signs so that $\Delta h_2 + \Delta h_1 = 0$, meaning a collision after the second message block. We emphasize that setting these conditions on the message does *not* increase the number of conditions on the resulting differential path, and hence it does not affect the complexity.

To summarize, the near collision on the second message block can be found with the same complexity as the near collision for the first message block. Therefore, there is only a factor of two increase in the overall complexity for getting a two-block full collision.

4.8 Complexity Analysis and Additional Techniques

Using the modification techniques described in this section, we can correct the conditions of steps 17-22. Furthermore, message modification will not result in increased complexity if we use suitable implementation tricks such as "precomputation". First, we can precompute and fix a set of messages in the first 10 steps and leave the rest as free variables. By Table 9, we know that there are 70 conditions in steps 23-77. For three conditions in steps 23-24, we use the "early stopping technique". That is, we only need to carry out the computation up to step 24 and then test whether three conditions in steps 23-24 hold. This needs about 12 step operations including message modification for correcting conditions of steps 17-22. This is equivalent to about two SHA-1 operations. Hence, the total complexity of finding the near-collision for the full SHA-1 is about 2^{68} computations. Considering the complexity of finding the second near-collision differential path, the total complexity of finding a full SHA-1 collision is thus about 2^{69}.

The results for SHA-1 reduced variants are summarized in Table 6 and Table 7 in the appendix.

5 Detailed Analysis: a 58-step Collision of SHA-1

When $t = 58$, our analysis suggests that collisions can be found with about 2^{33} hash operations, which is within the reach of computer search. In this section, we describe some details on how to find a real collision for this SHA-1 variant. The collision example is given in Table 4.

5.1 Constructing the Specific Differential 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 [12], we use $a_i[j]$ to denote

h_0:	67452301 efcdab89 98badcfe 10325476 c3d2e1f0
M_0:	132b5ab6 a115775f 5bfddd6b 4dc470eb 0637938a 6cceb733 0c86a386 68080139
	534047a4 a42fc29a 06085121 a3131f73 ad5da5cf 13375402 40bdc7c2 d5a839e2
M_0':	332b5ab6 c115776d 3bfddd28 6dc470ab e63793c8 0cceb731 8c86a387 68080119
	534047a7 e42fc2c8 46085161 43131f21 0d5da5cf 93375442 60bdc7c3 f5a83982
h_1:	9768e739 b662af82 a0137d3e 918747cf c8ceb7d4

Table 4. A collision of SHA-1 reduced to 58 steps. Note that padding rules are not applied to the messages, and $compress(h_0, M_0) = compress(h_0, M_0') = h_1$.

$a_i[j] = a_i + 2^{j-1}$ with no bit carry, and $a_i[-j]$ to denote that $a_i[-j] = a_i - 2^{j-1}$ with no bit carry.

We use step 23 to step 80 of the disturbance vector in Table 5 to construct a 58-step differential path that leads to a collision. The specific path for the first 16 steps is given in Table 10, and the rest of the path consists of the usual local collisions.

As we discussed before, there are two major complications that we need to deal with in constructing a valid differential path in the first 16 steps. In what follows, we describe high-level ideas as how to deal with the above two problems, and some technical details are omitted.

1. Message differences from a disturbance initiated in steps -5 to -1. These differences are $m_0[30], m_1[-5, 6, -30, 31], m_2[-1, 30, -31]$.
2. Consecutive disturbances in the same bit position in the first 16 steps. There are two such sequences: (1) $x_{1,2}, x_{2,2}, x_{3,2}$ and (2) $x_{8,2}, x_{9,2}, x_{10,2}$.

It is more instructive to focus on the values of Δa_i without carry expansion, which is the left column for Δa_i in Table 10. We first consider the propagation of the difference $m_1[-5, 6]$. It produces the following differences:

$$a_2[5] \to a_3[10] \to a_4[15] \to a_5[20] \to a_6[25].$$

These differences in a propagate through b, c, d to the following differences in the chaining variable e:

$$e_6[3] \to e_7[8] \to e_9[13] \to e_9[18] \to e_{10}[23].$$

The differences in b, c, d are easy to deal with since they can be *absorbed* by the Boolean function. So we only need to pay attention to variables a and e. The difference $a_6[25]$ as well as the five differences in e_i are cancelled in the step *immediately after* the step in which they first occur. This way, they will not propagate further. The cancellation is done using either existing differences in other variables or extra differences from the carry effect. For example, we expand $a_8[-18]$ to $a_8[18, 19, ..., -26]$ so that $a_8[25, -26]$ can produce the bit difference $c_{10}[23, -24]$ to offset $e_{10}[23]$, and $a_8[-26]$ produce $b_9[-26]$ to cancel out $e_9[26]$.

The consecutive disturbances are handled in different ways. For the first sequence, the middle disturbance $m_2[2]$ is combined with $m_2[1]$ so that the disturbance is shifted from bit 2 to bit 1. For the second sequence, the middle disturbance $m_9[2]$ is offset by $c_9[2]$, which comes from the difference $a_7[4]$.

One might get too swamped with the technicality for deriving such a complicated differential path. It is helpful to summarize the flow in the main approach: (1) analyze the propagation of differences, (2) identify wanted and un-wanted differences, and (3) use the Boolean function and the carry effect to introduce and absorb these differences.

5.2 Deriving Conditions on a_i and m_i

The method for deriving conditions on the chaining variables is essentially the same as in our analysis of SHA-0 [16], and so the details are omitted here.

The method for deriving conditions on the messages is more complicated since it involves more bit positions in the message words. To simplify the analysis, we first find a partial message (the first 12 words) that satisfies all the conditions in the first 12 steps. This can be done using message modification techniques in a systematic way. This leaves us with four free variables, namely $m_{12}, m_{13}, m_{14}, m_{15}$. Next we can write each m_i $(i \geq 16)$ as a function of the four free variables using the message expansion recursion. Conditions on these m_i then translate to conditions on $m_{12}, m_{13}, m_{14}, m_{15}$, and these bits will be fixed during the collision search.

6 Conclusions

In this paper, we present the first attack on the full SHA-1 with complexity less than 2^{69} hash operations. This attack is also available to find one-block collisions for the SHA-1 reduced variants less than 76 rounds. For example, we can find a collision of 75-round SHA-1 with complexity 2^{78}, and find a collision of 70-round SHA-1 with complexity 268.

Some strategies of the attack can be utilized to further improve the attacks on MD5 and SHA-0 etc. For example, applying the new technique of combining near-collision paths into a collision path, we can improve the successful probability of the attack on MD5 from 2^{-37} to 2^{-32}.

At this point, it is worth comparing the security of the MD4 family of hash functions against the best known attacks today. We can see that more complicated message preprocessing does provide more security. However, even for SHA-1, the message expansion does not seem to offer enough avalanche effect in terms of spreading the input differences. Furthermore, there seem to be some unexpected weaknesses in the structure of all the step updating functions. In particular, because of the simple step operation, the certain properties of some Boolean functions combined with the carry effect actually facilitate, rather than prevent, differential attacks.

We hope that the analysis on SHA-1 as well as other hash functions will provide useful insight on design criteria for more security hash functions. We anticipate that the design and analysis of new hash functions will be an important research topic in the coming years.

Acknowledgements

It is a pleasure to acknowledge Arjen K. Lenstra for his important suggestions, 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, pringer-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, 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.* 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. H. B. Yu, Y. Lisa Yin, *Efficient Collision Search Attacks on SHA-0.* These proceedings. 2005.

Appendix: Tables

index i	*index*	vector x_{i-1}	index i	*index*	vector x_{i-1}	index i	*index*	vector x_{i-1}
1		e0000000	33	*17*	80000002	65	*49*	2
2		2	34	*18*	0	66	*50*	0
3		2	35	*19*	2	67	*51*	0
4		80000000	36	*20*	0	68	*52*	0
5		1	37	*21*	3	69	*53*	0
6		0	38	*22*	0	70	*54*	0
7		80000001	39	*23*	2	71	*55*	0
8		2	40	*24*	2	72	*56*	0
9		40000002	41	*25*	1	73	*57*	0
10		2	42	*26*	0	74	*58*	0
11		2	43	*27*	2	75	*59*	0
12		80000000	44	*28*	2	76	*60*	0
13		2	45	*29*	1	77	*61*	0
14		0	46	*30*	0	78	*62*	0
15		80000001	47	*31*	0	79	*63*	0
16		0	48	*32*	2	80	*64*	0
17	*1*	40000001	49	*33*	3	81	*65*	4
18	*2*	2	50	*34*	0	82	*66*	0
19	*3*	2	51	*35*	2	83	*67*	0
20	*4*	80000002	52	*36*	2	84	*68*	8
21	*5*	1	53	*37*	0	85	*69*	0
22	*6*	0	54	*38*	0	86	*70*	0
23	*7*	80000001	55	*39*	2	87	*71*	10
24	*8*	2	56	*40*	0	88	*72*	0
25	*9*	2	57	*41*	0	89	*73*	8
26	*10*	2	58	*42*	0	90	*74*	20
27	*11*	0	59	*43*	2	91	*75*	0
28	*12*	0	60	*44*	0	92	*76*	0
29	*13*	1	61	*45*	2	93	*77*	40
30	*14*	0	62	*46*	0	94	*78*	0
31	*15*	80000002	63	*47*	2	95	*79*	28
32	*16*	2	64	*48*	0	96	*80*	80

Table 5. Disturbance vectors of SHA-1. The 96 vectors x_i $(i = 0, ..., 95)$ satisfy the SHA-1 message expansion recursion, but no other conditions. The second *italicized* index is only needed for numbering the 80 vectors that are chosen for constructing the best 80-step near collision.

t-step SHA-1	start & end index of DV	HW in ro.2-4	# conditions in ro.2-4	complexity NC	2BC
80	17, 96	27	71	2^{68}	2^{69}
79	17, 95	26	71	2^{68}	2^{69}
78	17, 94	24	71	2^{68}	2^{69}
77	16, 92	23	71	2^{68}	2^{69}
76	19, 94	22	69	2^{66}	2^{67}
75	20, 94	21	65	2^{62}	2^{63}
74	21, 94	20	63	2^{60}	2^{61}
73	20, 92	20	61	2^{58}	2^{59}
72	23, 94	19	59	2^{56}	2^{57}
71	24, 94	18	55	2^{52}	2^{53}
70	25, 94	17	52	2^{49}	2^{50}
69	26, 94	16	50	2^{48}	2^{49}
68	27, 94	16	48	2^{46}	2^{47}
67	28, 94	16	45	2^{43}	2^{44}
66	29, 94	15	41	2^{39}	2^{40}
65	30, 94	13	40	2^{38}	2^{39}
64	29, 92	14	37	2^{35}	2^{36}
63	32, 94	12	35	2^{33}	2^{34}
62	33, 94	11	34	2^{32}	2^{33}
61	32, 92	11	31	2^{29}	2^{30}
60	29, 88	12	29	2^{27}	2^{28}
59	30, 88	10	28	2^{26}	2^{27}
58	29, 86	11	25	2^{23}	2^{24}
57	32, 88	9	23	2^{21}	2^{22}
56	33, 88	8	22	2^{20}	2^{21}
55	32, 86	8	19	2^{17}	2^{18}
54	33, 86	7	18	2^{16}	2^{17}
53	34, 86	7	18	2^{16}	2^{17}
52	32, 83	7	15	2^{13}	2^{14}
51	33, 83	6	14	2^{12}	2^{13}
50	34, 83	6	14	2^{12}	2^{13}

Table 6. Search complexity for near collisions (NC) and two-block collisions (2BC) of SHA-1 reduced to t steps. "Start & end index" refers to the index for disturbance vectors in Table 5. The complexity estimation takes into account the speedup using early stopping techniques (see Section 4.8), and the estimation for 78-80 steps also takes into accounts the speedup by advanced modification techniques (see Section 4.5).

SHA-1 reduced to t steps	start & end point of DV	HW in rounds 2-4	# conditions in rounds 2-4	search complexity
80	1, 80	31	96	2^{93}
79	2, 80	30	95	2^{92}
78	3, 80	30	90	2^{87}
77	4, 80	28	88	2^{85}
76	5, 80	27	83	2^{80}
75	6, 80	26	81	2^{78}
74	7, 80	25	79	2^{76}
73	8, 80	25	77	2^{74}
72	9, 80	25	77	2^{74}
71	10, 80	24	74	2^{71}
70	11, 80	24	71	2^{68}
69	12, 80	22	68	2^{66}
68	13, 80	21	62	2^{60}
67	14, 80	19	58	2^{56}
66	15, 80	19	55	2^{53}
65	16, 80	18	51	2^{49}
64	17, 80	18	48	2^{46}
63	18, 80	16	48	2^{46}
62	19, 80	16	45	2^{43}
61	20, 80	15	41	2^{39}
60	21, 80	14	39	2^{37}
59	22, 80	13	38	2^{36}
58	23, 80	13	35	2^{33}
57	24, 80	12	31	2^{29}
56	25, 80	11	28	2^{26}
55	26, 80	10	26	2^{24}
54	27, 80	10	24	2^{22}
53	28, 80	10	21	2^{19}
52	29, 80	9	17	2^{15}
51	30, 80	7	16	2^{14}
50	31, 80	7	14	2^{12}

Table 7. Search complexity for one-block collisions of SHA-1 reduced to t steps. Explanation of the table is the same as that for 6.

step	disturb in bit 2	disturb in other bits	comments
19	0	1	For a_{21}
20	0	2	For a_{21}, a_{22}
21	1	3	Condition a_{20} is "truncated"
22-36	2	4	
37	3	4	
38-40	4	4	
41-60	4	4	
61-76	2	4	
77	2	3	Conditions are "truncated"
78	2	2	starting at step 77.
79	(1)	(1)	Conditions for step 79,80
80	(1)	(1)	can be ignored in analysis

Special counting rules:

1. If two disturbances start in both bit 2 and bit 1 in the same step, then they only result in 4 conditions (see Section 4.8).
2. For Round 3, two consecutive disturbances in the same bit position only account for 6 conditions (rather than 8). This is due to the property of the MAJ function.

Table 8. Rules for counting the number of conditions in rounds 2-4

index	number of conditions	comments
21	$4-1-1=2$	4 cond's: $a_{20}, a_{21}, a_{22}, a_{23}$ $-$ a_{20} due to truncation $-$ a_{21} using modification
23,24,27,28 32,35,36	$2 \times 7 = 14$	
25,29,33,39	$4 \times 4 = 16$	
43,45,47,49	$4 \times 4 = 16$	
65,68,71,73,74	$4 \times 5 = 20$	
77	3	Truncation
79	0	2 conditions ignored
80	0	1 condition ignored
Total	71	

Table 9. Example: Counting the number of conditions for the 80-step near collision. The "index" refers to the second italicized index in Table 5.

step			Δa_i					
i	x_{i-1}	Δm_{i-1}	no carry	with carry	Δb_i	Δc_i	Δd_i	Δe_i
1	80000001	30	30	$-30, 31$				
2	2	-2 $-5, 6$ -30 31	2 5 -30 31	$-2, 3$ 5 -30 $-31, 32$	Δa_1			
3	2	$-1, -2$ -7 $30, -31$	1 10	1 10	Δa_2	$\Delta a_1^{\lll 30}$		
4	2	-7 30	-2 15 -5	$2, -3$ $-15, 16$ $5, -6$	...	$\Delta a_2^{\lll 30}$	$\Delta a_1^{\lll 30}$	
5	0	$-2, 7$ $30, 31$ 32	20 28 -1 -10	$-20, 21$ $-28, 29$ -1 $10, 11, -12$		...	$\Delta a_2^{\lll 30}$	$\Delta a_1^{\lll 30}$
6	0	-2 $-30, -31$	25 15	25 $-15, 16$			...	$\Delta a_2^{\lll 30}$
7	1	$1, 32$	1 8 $4, -21$	1 $-8, -9, 10$ $4, -21$				...
8	0	-6	-18	$18, ..., -26$	...			
9	80000002	$1, 2$	$-2, 32$ -9	$-2, 32$ $9, ..., -19$		...		
10	2	-2 $-5, 7$ 31					...	
11	80000002	$7, 31$	$2, -32$ 9	$2, -32$ 9	...			...
12	0	-2 $-5, -7$ -30 $31, -32$			Δa_{11}	...		
13	2	$-30, -32$	-2	-2		$\Delta a_{11}^{\lll 30}$	...	
14	0	$7, 32$			Δa_{13}		$\Delta a_{11}^{\lll 30}$	...
15	3	$1, 30$	1	1		$\Delta a_{13}^{\lll 30}$		$\Delta a_{11}^{\lll 30}$
16	0	$-6, -7$ 30			Δa_{15}		$\Delta a_{13}^{\lll 30}$	

Table 10. The differential path for the 58-step SHA-1 collision. Note that x_i $(i = 0..15)$ are the disturbance vector for the first 16 steps, which correspond to the 16 vectors indexed by 23 through 38 in Table 5. The Δ entries list the positions of the differences and their signs. For example, the difference 2^j is listed as $(j+1)$ and -2^j as $-(j+1)$.

step i	x_{i-1}	Δm_{i-1}	Δa_i no carry	Δa_i with carry	Δb_i	Δc_i	Δd_i	Δe_i
1	40000001	30	30, 31	30, 31				
2	2	$-2, -4$ 6 $-30, -31, 32$	2 6 30	$-2, 3$ $-6, -7, 8$ $-30, -31, 32$	Δa_1			
3	2	1, 2 -7 30	-1 4 11	-1 4 $-11, -12, -13, 14$	Δa_2	$\Delta a_1^{\lll 30}$		
4	80000002	7 $29, -30$ -32	$-2, 9$ 16 -32	$-2, 9$ $-16, -17, -18, 19$ -32	...	$\Delta a_2^{\lll 30}$	$\Delta a_1^{\lll 30}$	
5	1	$1, -2$ $-5, 7$ 29, 31, 32	-5 21 28	$5, -6$ $-21, 22$ 28		...	$\Delta a_2^{\lll 30}$	$\Delta a_1^{\lll 30}$
6	0	$-2, -6$ 29, 31 32	11 16 26	$-11, -12, 13$ $-16, 17$ $-26, 27$			...	$\Delta a_2^{\lll 30}$
7	80000001	30	1 $-4, -6$ 32	1 $-4, 6, -7$ 32				...
8	2	$-2, -5, -6$ 30, 31	-19	$19, ..., -26$	...			
9	2	$1, -2, -7$ $-30, -31$	-2 -10	-2 $10, ..., -20$		...		
10	2	7 -30	2	2			...	
11	0	$2, -7$ $-30, 31, -32$	9	$-9, 10$	...			...
12	0	2 $-30, -31$	-4	-4		...		
13	1	1 32	1	1			...	
14	0	-6						...
15	80000002	$-1, 2$	-32	-32				
16	2	$2, 5, -7$ -31	2	2	Δa_{15}			
17	80000002	-7 31	-2 32	-2 32	Δa_{16}	$\Delta a_{15}^{\lll 30}$		
18	0	$-2, -5, 7$ 30, 31, 32				$\Delta a_{16}^{\lll 30}$	$\Delta a_{15}^{\lll 30}$	
19	2	30 32	2	2			$\Delta a_{16}^{\lll 30}$	$\Delta a_{15}^{\lll 30}$
20	0	-7 32			Δa_{19}			$\Delta a_{16}^{\lll 30}$

Table 11. The differential path for the 80-step SHA1 collision. Note that x_i $(i = 0..19)$ are the disturbance vector for the first 20 steps, which correspond to the 20 vectors indexed by 1 through 20 in Table 5. The Δ entries list the positions of the differences and their signs. For example, the difference 2^j is listed as $(j+1)$ and -2^j as $-(j+1)$.

chaining variable	conditions on bits 32 − 25	24 − 17	16 − 9	8 − 1
a_1	a00-----	--------	1-----aa	1-0a11aa
a_2	01110---	------1-	0aaa-0--	011-001-
a_3	0-100---	-0-aaa0-	--0111--	01110-01
a_4	10010---	a1---011	10011010	10011-10
a_5	001a0---	--01-000	10001111	-010-11-
a_6	1-0-0011	1-1001-0	111011-1	a10-00a-
a_7	0---1011	1a0111--	101--010	-10-11-0
a_8	-01---10	000000aa	001aa111	---01-1-
a_9	-00-----	10001000	0000000-	---11-1-
a_{10}	0-------	1111111-	11100000	0-----0-
a_{11}	--------	------10	11111101	1-a--0--
a_{12}	0-------	--------	--------	10--11--
a_{13}	--------	--------	--------	11----10
a_{14}	-0------	--------	--------	----0-1-
a_{15}	10------	--------	--------	----1-0-
a_{16}	--1-----	--------	--------	----0-0-
a_{17}	0-0-----	--------	--------	------1-
a_{18}	--1-----	--------	--------	----a---
a_{19}	--b-----	--------	--------	------0-
a_{20}	--------	--------	--------	-----a--
a_{21}	--------	--------	--------	-------1

Table 12. A set of sufficient conditions on a_i for the differential path given in Table 11. The notation 'a' stands for the condition $a_{i,j} = a_{i-1,j}$ and 'b' denotes the condition $a_{19,30} = a_{18,32}$.