Collisions of SHA-0 and Reduced SHA-1*

Eli Biham[1]**, Rafi Chen[1],
Antoine Joux[2,3,]***,
Patrick Carribault[3], Christophe Lemuet[3], and William Jalby[3]

[1] Computer Science Department
Technion – Israel Institute of Technology
Haifa 32000, Israel
Email: {biham,rafi_hen}@cs.technion.ac.il
WWW: http://www.cs.technion.ac.il/~biham/
[2] DGA
Email: antoine.joux@m4x.org
[3] Laboratoire PRISM†
Université de Versailles St-Quentin-en-Yvelines
45, avenue des Etats-Unis
78035 Versailles Cedex
FRANCE
Email: {Patrick.Carribault,Christophe.Lemuet,William.Jalby}@prism.uvsq.fr

Abstract. In this paper we describe improvements to the techniques used to cryptanalyze SHA-0 and introduce the first results on SHA-1. The results include a generic multi-block technique that uses near-collisions in order to find collisions, and a four-block collision of SHA-0 found using this technique with complexity 2^{51}. Then, extension of this and prior techniques are presented, that allow us to find collisions of reduced versions of SHA-1. We give collisions of variants with up to 40 rounds, and show the complexities of longer variants. These techniques show that collisions up to about 53–58 rounds can still be found faster than by birthday attacks.

1 Introduction

The hash function SHA was designed by the National Security Agency (NSA) and issued by NIST in 1993 as a Federal Information Processing Standard (FIPS-180) [3]. A revised version called SHA-1, which specifies an additional rotate operation to the message expansion, was later issued in 1995 as FIPS-180-1 [4]. The revised version is aimed to be a more secure replacement, that improves the security provided by the hash function. No details of the weaknesses found

* Part of the results of this paper were given by the first author in an invited talk in SAC 2004, Waterloo, Canada.

** Part of this work was done while visiting École normale supérieure, Paris, France.

*** This work was mostly done while the author was at DCSSI Crypto Lab

† CNRS UMR-8144

in SHA-0 were provided. In order to refer more clearly to the first version, we denote it as SHA-0, which is a widely used but non standardized name.

SHA-0 and SHA-1 are based on the principles of MD4 [5] and MD5 [6]. They take messages of any length (up to 2^{64} bits) and compute 160-bit hash values.

At CRYPTO'98 Chabaud and Joux [2] proposed a theoretical attack on the full SHA-0 with a complexity of 2^{61}. It is a differential attack that uses a weakness of the expansion algorithm of SHA-0. Their attack is faster than the generic birthday paradox attack and partially explain the withdrawal of SHA-0 by NSA. It is interesting to note that they count the complexity in term of the number of message pairs to be tried and not in term of the number of SHA-0 calls. At first, it may seem to be an artificial way to reduce the claimed complexity by 2. However, due to the use of an early abort strategy in the implementation, the effective complexity in term of SHA-0 calls is roughly 1/4 of the announced value. For the sake of clarity, we continue this tradition and announce all the complexity results by giving the average number of necessary message pairs.

In [1] Biham and Chen discussed near-collisions of SHA-0. By using some of the ideas that originally appeared in [2], they showed that in SHA-0 near-collisions are easier to find than full collisions, and proposed an efficient searching algorithm that eliminates the probabilistic behavior of more than 20 rounds of the algorithm, using the notion of *neutral bits*. When applied to the attack of Chabaud and Joux, this improves the complexity by an approximate factor of 32.

In our current research we improve over the results of [1] in several directions: we first present a tool that uses near-collisions in order to find collisions using a multi-block technique. This tool can be used to attack variants that cannot be attacked by the original technique, as well as to reduce complexities of attacking other variants. With some additional refinements, it also improves the attack on full SHA-0, reducing the complexity down to 2^{51}. Then we present our attacks on reduced-round SHA-1, which can find collisions of up to 53–58 rounds faster than the birthday attack, and show new techniques to attack SHA-1.

In parallel to this paper, Rijmen and Oswald also recently studies reduced versions of SHA-1 [9].

This paper is organized as follows: In Section 2 we describe how near-collisions can be used to find collisions by a *multi-block* technique. In section 3, we show how the multi-block technique can be refined in order to work on the full SHA-0, this leads to a full collision on SHA-0 using messages of four blocks. In Section 4 we describe how the attack on SHA-0 is expanded to attack SHA-1. This section presents various attacks on reduced versions of SHA-1, where each attack emphasizes different aspects and techniques. A 34-round SHA-1 collision that can be found with relatively low complexity is introduced. With this reduced version we show how collisions can be found with messages that have only ASCII letters and even messages with some meaningful words. We continue with a collision of 36-round SHA-1 that uses a message of two blocks, where the first block changes the initial value to a value that is convenient for the attack, and the collision is found in the second block. This attack also shows some differences between the attack of SHA-0 and SHA-1, where the non-linearity of SHA is used in the attack. We then discuss how to bypass the consecutive disturbances problem in the IF rounds. The last attack in this section is a two-block collision of 40-round SHA-1 that uses the same characteristic in both blocks. All the collisions of reduced SHA-1 that we present were found within a few seconds of computation on a PC. Section 5 analyzes the complexity of attacking various reduced versions of SHA-1 with more rounds, and shows that SHA-1 up to 53–58 rounds can be attacked faster than the birthday attack. The assessments are based on the best characteristics we could find for each reduced version. Section 6 summarizes the paper.

Due to lack of space, we removed the descriptions of SHA-0 [3] and SHA-1 [4], and the description of prior techniques related to this paper, e.g., the original technique for analysis of SHA-0 [2], the improved technique and neutral bits [1]. For descriptions of SHA and these techniques, see the respective references. We also shortened the descriptions of some results and removed some detailed explanations about the attack complexity. The full details will appear in the full version of the paper.

Note This paper is the result of the merge of two papers by non-intersecting groups of authors. The first group consists of the Technion authors, and the other consists of the DGA and PRISM authors. The multi-block technique as a generic tool including the 50-rounds SHA-0 application and the results on SHA-1 are due to the first group. Motivated by their work, the first author within the second group restarted searching on old, non-working results about iterated collisions in SHA-0. It resulted in an improved multi-block cryptanalysis for full SHA-0, which was then ported and optimized for the supercomputer by the other authors within the group.

2 The Multi-Block Tool

SHA uses an iterative process in which each block M_j along with an intermediate value h_{j-1} is subjected to a compression function, whose output is the value of the next intermediate value h_j. Previous works on hash functions, and in particular on SHA, use only one block for the attack. Those attacks start with the initial value h_0 and construct a pair of messages M_1 and M_1^* that output the same h_1 to find a collision, or h_1, h_1^* with a small difference h_1' for near-collisions.

The tool we present in this section uses the iterative process of SHA to find collisions. The idea of this technique is to start with a pair of blocks M_1 and M_1^* that create a near-collision h_1', and continue with a construction of a second block. In the first block we base the message on a characteristic that has a zero input difference h_0', and a non-zero output difference h_1', with some message difference M_1'. In the second block we use a characteristic with a non-zero input difference h_1', and a zero output difference h_2'.

The attack proceeds as follows: Given messages M_1, M_1^* that conform to the first characteristic, we receive the pair of intermediate hash values h_1 and h_1^*. Using these values, we search for a second block M_2, M_2^* whose input values are

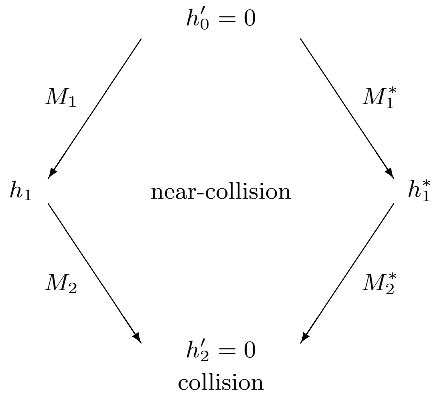

Fig. 1. Using Intermediate Near-Collisions to Find Collisions with Two Blocks

h_1, h_1^*, and which conforms to the second characteristic. Such a pair will then have $h'_2 = 0$, which means a collision after the second block.

As a result we succeed in finding a near-collision in the first block, and then finding a second block, constructed in a similar way, but in which each message starts with a different input value (rather than same value as is usually done in hash functions) in order to find a collision. An illustration of a two-block attack is given in Figure 1.

The multi-block tool is particularly useful when there is no characteristic that predicts a full collision in one block, and to reduce the complexity of an attack when a single-block collision is more complex.

It should be noted that Wang [7, 8] independently used two message blocks to find the collision of MD5, using a first block that creates a near-collision, and a second block that restarts from this near-collision and ends with a collision.

Applications In order to illustrate the multi-block technique, we can apply to SHA-0 reduced to 50-rounds. This example is interesting, since this reduced version does not have any characteristic (i.e., any disturbance vector) that predicts a collision with a single block. However, it is very easy to find near-collisions with complexity of about 2^{17}. Using the multi-block technique, we can restart from this near-collision in order to find a longer message pair that collides after the second block. The total complexity remains about 2^{17}.

Collisions with More than Two Blocks This technique can be generalized to several blocks. In the case of two blocks the first block of the messages M, M^* is constructed by using a characteristic that has a zero input difference h'_0 and, a non-zero output difference h'_1. In the second block we use a characteristic whose input difference is h'_1, and which has a non-zero output difference h'_2. In the case of two blocks $h'_2 = 0$, which means a collision. However, in case $h'_2 \neq 0$, it is

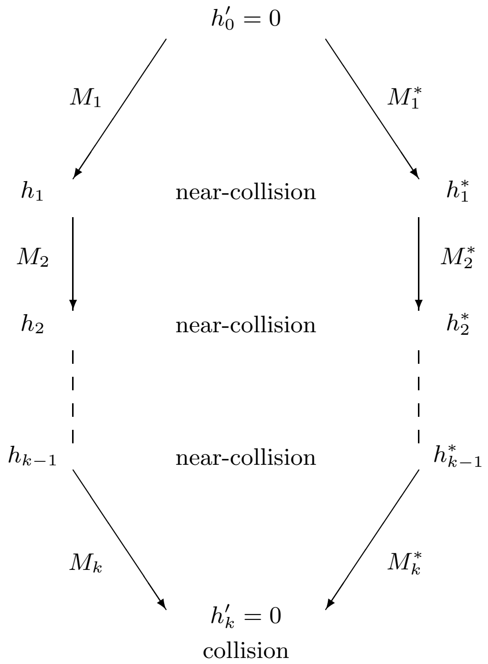

Fig. 2. The Multi-Block Technique—Using Intermediate Near-Collisions to Find Collisions

possible to use h'_2 as the input difference of a third block which leads to a collision (see Figure 2). Alternatively the third block can lead to another near-collision that may later be converted to a collision of the fourth block. In general the technique can find k-block collisions, where the first block starts with $h'_0 = 0$, with $k-1$ intermediate near-collisions $h'_i \neq 0$ ($i = 1, \ldots, k-1$), which lead to a collision with $h'_k = 0$ after k blocks. The complexity of finding the k-block collision is the sum of the complexities of finding the $k-1$ near-collisions and the final collision. More information on usage of multi-block collisions will appear in the full paper.

3 A Multi-Block Collision of SHA-0

Since the multi-block technique described above is very promising, it is extremely tempting to apply it to the full 80 rounds SHA-0. Unfortunately, contrarily to what happens with the 50-rounds version, there is no attack of this type which behaves better than the single block attack proposed by Chabaud and Joux. All the other paths that use near collisions happen to be dead-ends.

In order to remove this obstruction, another key idea is necessary. We should note that in the early rounds of SHA-0, an IF function is used. This means, that during the early rounds, SHA-0 may in some case behave differently than the linearized model of [2]. This misbehavior might allow us to connect differentials which do not belong together in the linearized model of SHA-0. In order to make this idea precise, we first introduce some notations to describe the differences before and after each block. First, remark that in each register A to E, after a successful application of a one block differential, a difference may occur at a single, fixed, position. In A and B a difference may occur at bit 1, in C, D and E at bit 31. As a consequence, to describe an initial or final difference, a 5-bit number suffices. We assign the high order bit to A and the low order bit to E. Thus, a state with a single difference D will be referred to as state 2. The second step is to compare the expected behavior of a reference state in the linearized model with the possible behaviors of a given state when the IF function is used, i.e., in real-life SHA-0. This is done by examining how the initial difference propagates in the five first rounds.[4] To start with a simple example, assume that reference state 2 is considered in the linearized model. In that case, we have a single initial difference on bit 31 of D. Due to the XOR function, this difference propagates in the update formula for the next value of A. Thanks to the disturbance vector, it will be adequately corrected, however, this is not relevant for this part of the discussion, we just need to know that it propagates in the formula. Then, the registers are shifted and the initial difference moves to E. In the next update formula, it will also propagate, again on bit 31. After that round, the initial difference has vanished and no longer propagates. Now, consider that state 3 enters the real SHA-0. Then, in the first formula, both D and E have a difference on bit 31, however, depending on the result of the IF function the difference on D may either propagate or not. More precisely, if bit 31 of register B (which is the same in both messages) is a 1, the difference on D does not propagate. On the other hand, the difference on E always propagates. The gross result is that a single difference propagates on bit 31, thus at this point state 3 behaves as reference state 2 in the linearized model. After the registers shift, a difference remains on E and it propagates in the second update formula. As a consequence, we see that real state 3 may behave as reference state 2. Thus, we may start a differential attack from state 3 by using a disturbance vector that "expects" state 2. Moreover, state 3 may also behave like reference state 3. This implies, that it is possible to connect together much more differentials than initially expected. Thus, the graph of possible paths is considerably richer than first predicted and we expect to find a better attack.

With this translation table in mind, we now try to assemble several differentials with different disturbance vectors into a global attack. For any disturbance vector, we add five extra bits, the "negative" bits which indicate the starting reference state. Similarly, the value of the last five bits indicate the expected

[4] We further remark that this representation can be extended to a general kind of characteristics describing the evolution of differences in registers $A, \ldots, E$, and in the expanded message, in a similar way to the characteristics used in related-key differential cryptanalysis. In such a case, the intermediate differences can be very different than predicted by the model of [2], while still leading to collisions.

state after the block cipher part of the compression function. To incorporate such a disturbance vector into the global attack, we proceed as follows: Assume that the current state is a and that we are given a disturbance vector $a' \to b$, i.e., a disturbance that goes from reference state a' to expected state b, then if a is compatible with a', we have a differential that goes from state a to next state $a \oplus b$ after the final addition. Thus, we can build a transition graph, where each possible state is a node, and each differential, with good enough probability, is an edge. In this graph, we now search for a path from state 0 to itself, with low expected complexity. The best path we could find has length 4, it starts from state 0, goes to 3, 25, 8 and finally comes back to zero. It is build on the following disturbance vectors:

Ref (DV) States	DV	Actual States	Compatible With
$0 \to 3$	00000 00010000101001000111 10010110000011100000 00000011000000110110 00000110001011011000	$0 \to 3$	**2** 3
$2 \to 26$	01000 10000000010000101001 00011110010110000011 10000000000011000000 11011000000110001011	$3 \to 25$	**17** 18 28 31
$17 \to 17$	10001 00100101000100101111 11000010000100001100 00101100100000000001 11010011101000010001	$25 \to 8$	8 **11** 13 14
$11 \to 8$	11010 00100000000100001010 01000111100101100000 11100000000000110000 00110110000001100010	$8 \to 0$	collision

One can easily check that this sequence of block differences can possibly lead to a full collision. Initially, the difference between the two messages of a pair corresponds to state 0. After the first block, we intend to reach state 3. Of course, for this block the final additions add equal values on each branch, thus the difference is expected to remain at state 3. Since state 3 is compatible with reference state 2, we restart from there and go to state 26. For this second block, the additions change state 26 into state 25. Again, the compatibility of 25 with reference state 17 allows us to restart with the third block difference. The expected state is 17 before the final additions and 8 after them. Thanks to the compatibility of 8 with 11, we use the fourth difference and expect a state of 8 before the additions. Since the two states 8 correspond to differences on the same bits, we expect that they cancel each other. Thus, we finally reach a full collision. Evaluating the exact complexity of this attack requires a detailed analysis that, for lack of space, is not given here. The total cost is 2^{51} message pairs, as confirmed by our implementation.

3.1 Implementation and Optimization

The theoretical complexity of our collision search algorithm is 2^{51}. This complexity is expressed in term of the number of pairs of messages to test. As is the case with the original attack of Chabaud and Joux, this roughly corresponds to the cost of 2^{49} evaluations of the SHA-0 compression function.

In order to demonstrate feasibility of this collision search, we implemented this algorithm on an Intel Itanium 2 processor. This processor allows a wide degree of instruction level parallelism (ILP). More precisely, it is able to execute up to six instructions per cycle, and a wide variety of combinations is possible (e.g., 6 arithmetical operations, or 4 memory operations and 2 floating point multiply add, or 3 logical operations and 3 branches, etc.). Furthermore, this wide ILP capability is enhanced by a large register file and many duplicated functional units. The processor also offers several mechanisms to implement control/branch structures with speculative execution, predication, and multi-way branches (up to three branches per cycle). Due to the complex nature of the processor, the performance of programs running on it heavily relies on the capability of the compiler to produce efficient code. Our algorithm was compiled by the Intel compiler (ICC) whose performance in this respect is usually above average.

To optimize our code, a *profiling* step was performed to detect the most time-consuming code sections. This study revealed that the main function, which enumerates pairs of messages derived from a reference pair and its' neutral set represented a large majority of the execution time. Focusing on this part, we checked the behavior of the code at the hardware level during execution through the use of *hardware counters*.

We, thus, determined that the main performance limiting factors were:

- *Limited amount of parallelism*: All rounds of SHA-0 contain chains of bitwise operations ($+$, ROL_x, $\ldots$) depending on each other, which limited the effect of the internal parallelism.
- *Complex control flow*: Due to the probabilistic nature of the collision search, the control flow is quite complex and statistically (at compile time) unpredictable.
- *Cached memory access*: Despite being in a very favorable case where all data fits in the first level data cache (L1D) of the Itanium 2 (16KB), the number of accesses to the cached memory is very high, when arrays are used to represent the intermediate values during the computation. As a consequence, memory access in L1D was a bottleneck in our basic implementation.

Due to the complex control flow, the Intel compiler could not determine a good way to execute branches. Even the use of advanced optimization tools such as *profile guided optimization*, did not help much. The compiler still used speculative execution, which led to bad performance. A first step in our tuning process was to make the compiler avoid speculation, by writing each round differently, depending on the probability of success at this point.

Since the number of L1D memory accesses was critical, the second step consisted in reducing them. This was done by replacing all arrays by registers thus avoiding many memory stores and loads. This optimization makes good use of the large number of registers of the Itanium 2. Such a technique is called *register promotion* and is usually performed by the compilers. However, in this example, this had to be done on a large number of source lines and the compiler was unable to deal with this. Moreover, we had to extend the technique to deal with the complex control structure.

All the fine tuning techniques allowed to gain an additional 20% of performance compared to the best compiler options (which are not the standard `O3` options and had to be determined through exhaustive search). On average, 4 instructions per cycle were effectively executed, out of a maximum 6.

3.2 A Full Collision of SHA-0

Once the program was ported to the supercomputer, it processed a large number of messages pairs for each block. Very precisely, the total number of trial pairs was:

First block	$796\,682\,307\,091\,035 \approx 2^{49.5}$
Second block	$1\,572\,177\,940\,314\,628 \approx 2^{50.5}$
Third block	$1\,712\,558\,626\,669\,268 \approx 2^{50.6}$
Fourth block	$17\,049\,400\,703\,749 \approx 2^{44}$

We can remark that the number of computations is higher than expected for the first two blocks. At first, we simply assumed that we had been unlucky, however, a deeper investigation revealed a subtle bug in the neutral bits identification code. Due to this bug, some messages pairs were processed more than once, and up to four times, by the program. These useless computations explain the mismatch between the predicted complexity of the first two blocks and the effective numbers of messages pairs processed. Luckily, the bug did not affect the computation of block 3, thus the total slowdown was limited. Finally, we reached the following messages (written in hexadecimal):

```
a766a602 b65cffe7 73bcf258 26b322b3 d01b1a97 2684ef53 3e3b4b7f 53fe3762
24c08e47 e959b2bc 3b519880 b9286568 247d110f 70f5c5e2 b4590ca3 f55f52fe
-----------------------------------------------------------------------
effd4c8f e68de835 329e603c c51e7f02 545410d1 671d108d f5a4000d cf20a439
4949d72c d14fbb03 45cf3a29 5dcda89f 998f8755 2c9a58b1 bdc38483 5e477185
-----------------------------------------------------------------------
f96e68be bb0025d2 d2b69edf 21724198 f688b41d eb9b4913 fbe696b5 457ab399
21e1d759 1f89de84 57e8613c 6c9e3b24 2879d4d8 783b2d9c a9935ea5 26a729c0
-----------------------------------------------------------------------
6edfc501 37e69330 be976012 cc5dfe1c 14c4c68b d1db3ecb 24438a59 a09b5db4
35563e0d 8bdf572f 77b53065 cef31f32 dc9dbaa0 4146261e 9994bd5c d0758e3d
```

and

```
a766a602 b65cffe7 73bcf258 26b322b1 d01b1ad7 2684ef51 be3b4b7f d3fe3762
a4c08e45 e959b2fc 3b519880 39286528 a47d110d 70f5c5e0 34590ce3 755f52fc
-----------------------------------------------------------------------
6ffd4c8d 668de875 329e603e 451e7f02 d45410d1 e71d108d f5a4000d cf20a439
4949d72c d14fbb01 45cf3a69 5dcda89d 198f8755 ac9a58b1 3dc38481 5e4771c5
-----------------------------------------------------------------------
796e68fe bb0025d0 52b69edd a17241d8 7688b41f 6b9b4911 7be696f5 c57ab399
a1e1d719 9f89de86 57e8613c ec9e3b26 a879d498 783b2d9e 29935ea7 a6a72980
-----------------------------------------------------------------------
6edfc503 37e69330 3e976010 4c5dfe5c 14c4c689 51db3ecb a4438a59 209b5db4
35563e0d 8bdf572f 77b53065 cef31f30 dc9dbae0 4146261c 1994bd5c 50758e3d
```

which have the same hash values. More precisely, the intermediate hashes for both messages are compatible with the predictions of our differential attack and their precise values are:

IV	67452301	EFCDAB89	98BADCFE	10325476	C3D2E1F0
Block 1	83C1CE2D	C5BF5480	C2AF2358	104B337B	9E78A1E7
Block 2	27AE025A	9D36F7B6	29FA88E7	87B70063	984119F3
Block 3	4DD120B4	D6EC801F	468628A7	0CC26464	371F36B2
Block 4	81FB4643	08FDF1F4	A3C4F3A3	6188FED3	FD2378E6
Padding	C9F16077	7D4086FE	8095FBA5	8B7E20C2	28A4006B
IV	67452301	EFCDAB89	98BADCFE	10325476	C3D2E1F0
Block 1	83C1CE2D	C5BF5480	C2AF2358	904B337B	1E78A1E7
Block 2	27AE0258	9D36F7B4	29FA88E7	87B70063	184119F3
Block 3	4DD120B4	D6EC801D	468628A7	0CC26464	371F36B2
Block 4	81FB4643	08FDF1F4	A3C4F3A3	6188FED3	FD2378E6
Padding	C9F16077	7D4086FE	8095FBA5	8B7E20C2	28A4006B

In this table, the underlined values highlight the difference between the two hash processes. These differences are as predicted by our differential attack. After the fourth blocks, the two messages collide. Of course, since the two messages have the same length, the padding blocks are identical. Thus, the final values inherit from the fourth block collision.

4 SHA-1 Results

Our attack on SHA-1 extends the techniques of [1] designed for SHA-0. The only difference between SHA-1 and SHA-0 is an additional rotation operation in the expansion process. Due to this rotation SHA-1 mixes the bits in the expanded message in a more efficient way than SHA-0 does, thus making the attack much less efficient against SHA-1 (as was already noted in [1]). In this section we observe that with some modifications, the attack can be applied to reduced versions of SHA-1. In the next subsections we present collisions of 34–40 rounds SHA-1 that we found using this application.

4.1 A Collision of 34-Round SHA-1

The attacks of SHA-0 use only bit 1 as the location of disturbances. This bit is selected to eliminate the probabilistic behavior of the carry when corrections are applied to bit 31, thus increasing the total probability of the characteristic. Since the expansion process in SHA-0 does not mix bits in different locations in the 32-bit word, all the disturbances in the expanded message are in bit 1, but this is not the case in SHA-1. Therefore, other bits can be used as disturbances. With this change in the selection, a disturbance vector in SHA-1 is not boolean, in which each entry tells whether there is a disturbance in bit 1, but instead a 32-bit word that represents all the disturbances in a round. Following this change, the corrections associated with a disturbance vector are derivated slightly differently than in [1] (i.e., corrections are applied to each disturbance relative to its location).

We observe that for 34-round reduced SHA-1 (unlike longer versions) there is a disturbance vector with a very low Hamming weight, which is given in Table 1. In this table D.Vec column shows the expected values of A'_{i+1} for $i = 0, \ldots, 33$,

Rnd	D.Vec	D&C	Rnd	D.Vec	D&C	Rnd	D.Vec	D&C
−5	00000000		8	00000000	80000003	21	00000000	00000040
−4	00000000		9	00000002	40000002	22	00000002	00000000
−3	00000000		10	00000000	C0000040	23	00000000	80000040
−2	00000000		11	00000000	C0000002	24	00000000	80000002
−1	00000000		12	00000000	80000000	25	00000000	00000000
0	00000002	00000002	13	00000000	80000000	26	00000000	80000000
1	00000000	00000040	14	00000002	80000002	**27**	**00000000**	80000000
2	00000002	00000000	15	00000000	00000040	**28**	**00000000**	00000000
3	00000000	80000040	16	00000000	00000002	**29**	**00000000**	00000000
4	00000002	80000000	17	00000000	80000000	**30**	**00000000**	00000000
5	00000000	00000040	18	00000000	80000000	**31**	**00000000**	00000000
6	00000003	80000001	**19**	**00000000**	80000000	**32**	**00000000**	00000000
7	00000000	00000060	20	00000002	00000002	**33**	**00000000**	00000000

Table 1. The Disturbance Vector Used for 34-round SHA-1 (in 32-bit hex words)

```
Message 1:
F1641C2B 242BFDB5 EAE01E30 F4BBBA6F 18D45E8E DE68AEBA 74EC8CF9 FC204957
45AAA8BF 1CD3AE7D D845C2F3 AC737464 F25BEBBB BE5FFF1D 2ADB2818 0B1D13FB
Message 2:
F1641C29 242BFDF5 EAE01E30 74BBBA2F 98D45E8E DE68AEFA F4EC8CF8 FC204937
C5AAA8BC 5CD3AE7F 1845C2B3 6C737466 725BEBBB 3E5FFF1D AADB281A 0B1D13BB
```

Table 2. Collision of SHA-1 Reduced to 34 Round (in 32-bit hex words)

which we denote by δ_{i+1} (note that the indices of δ here are 1, …, 34, rather than 0, …, 33, as δ_i is the difference at the input of round i). The D&C column shows the message difference (which is underlined and shown in rounds $0, \ldots, 15$), and the differences of the expanded messages in rounds $16, \ldots, 33$. $\delta_{-4}, \ldots, \delta_0$ are also shown in this table (in rounds $-5, \ldots, -1$), and their values are related to the initial value differences. Intermediate rounds after which collisions are predicted are marked in boldface. This disturbance vector is unique in that almost all the disturbances are in one location (bit 1), and in all of the 34 rounds there is only a single disturbance in a different location, which is bit 0 in round 6. This disturbance succeeds to cancel the avalanche effect that is expected in SHA-1 due to the additional rotate operation, and that does exist in other disturbance vectors.

By using the neutral bits method of [1], the complexity of an attack can be estimated based on the number of disturbances after round 20. Thus, using this disturbance vector, that has only two disturbances after round 20, we easily found millions of collisions, one of which is given in Table 2.

Since the complexity of finding collisions in this 34-round attack is so low, we were able to generate collisions with additional constraints, which caused

Message 1:
`IkGDqVMwISGGcBMpNHMYavPTsmUlykPTzokJOkwnrSgJSfDmlpeqsmDzWbAjmNxP`
Message 2:
`IkgDqRMwISGGcFEpNHEYarPTsmMlymPTzoSJOkSnrWkJSfhmlpmqsmLzWbijmJxP`

Table 3. Two Messages in ASCII Letters that Collide Under 34-Round SHA-1

Message 1:
`I Am OilMANgujnPay916472136314$USAkNOWwTkjepMFXGlmfHNGcpodElGfvL`
Message 2:
`I am KilMANgunfPay11607213.312$USASNOWSTknipMFtGlmnHNGkpodmlGbvL`
Message 1:
`OhG, not this mess,age notThat onenot U, oh noHRtBMTkKllLlIluvpB`
Message 2:
`Ohg, jot this$eess$aga notLhar oneVot q, kd nodRtBETkKdlLlalurpB`

Table 4. Two Examples of Partially Meaningful Messages that Collide Under 34-Round SHA-1

some increase in the complexity. This way we found collisions whose all bytes are formed of ASCII letters. The disturbance vector of Table 2 does not allow that, as some bytes of the message differ in the most significant bits. However, by rotating the locations of the disturbances (by the same number of bits in all the rounds) we can move the differences to lower bits, while increasing the complexity of the attack by a small factor. A colliding pair of messages consisting entirely of letters in ASCII is given in Table 3.

With some additional creativity, and some additional increase in the complexity, it was also possible to force some of the bytes into partial English text. Table 4 lists two examples. The first example is an attempt to force the two colliding messages to contain meaningful text. However, there are still many constraints on the possible text, thus it can be seen that some letters are capitalized, while other are not, that some spaces appear between words, while they do not appear between other words, and that some random letters must be allowed in some locations in order to allow more text afterwards. The second example in Table 4 is an attempt to further improve the text of one message in the expense of the text of the other message.

4.2 A Collision of 36-Round SHA-1

In this section we present a collision of 36-round reduced SHA-1 along with several techniques that were used to find it.

In our attack on 36-round SHA-1 we use the best characteristic that predicts a collision after one block. We show this disturbance vector in Table 5.

It should be noted that this characteristic cannot be used with the standard initial value of SHA-1, i.e., with

$$h_0 = (67452301_x,\ EFCDAB89_x,\ 98BADCFE_x,\ 10325476_x,\ C3D2E1F0_x),$$

Rnd	D.Vec	D&C	Rnd	D.Vec	D&C	Rnd	D.Vec	D&C
-5	00000000		9	00000002	00000008	23	00000000	80000050
-4	00000000		10	00000002	00000042	24	80000003	80000001
-3	00000000		11	80000000	50000042	25	00000000	A0000070
-2	00000000		12	00000002	10000010	26	00000000	20000003
-1	00000000		13	00000000	90000040	27	00000002	40000002
0	80000000	80000000	14	00000002	20000000	28	00000000	E0000040
1	00000000	00000010	15	00000000	20000040	29	00000000	E0000002
2	00000001	80000001	16	80000003	20000001	30	00000002	80000002
3	00000000	20000020	17	00000000	00000070	31	00000000	80000040
4	00000003	20000002	18	00000000	00000003	32	00000000	80000002
5	00000002	60000062	19	00000002	60000002	33	00000000	80000000
6	00000001	40000042	20	00000000	E0000040	34	00000000	80000000
7	00000002	80000020	21	00000000	E0000002	35	**00000000**	80000000
8	40000000	00000041	22	80000002	00000002			

Table 5. The Disturbance Vector Used for 36-Round SHA-1 (in 32-bit hex words)

Common block 1: sixteen 00000000 words							
Message 1, block 2:							
9F29DE8D	BBD58270	1F11EB22	A6637C3E	7E6FB0C0	63E9BF5E	C4FF7010	073174B3
3133689A	579A753E	2D17124D	7D37E853	5B5BBB01	F0371FBB	025A725C	8FB9FE33
Message 2, block 2:							
1F29DE8D	BBD58260	9F11EB23	86637C1E	5E6FB0C2	03E9BF3C	84FF7052	87317493
313368DB	579A7536	2D17120F	2D37E811	4B5BBB11	60371FFB	225A725C	AFB9FE73

Table 6. The Second Block of the Collision of 36-Round SHA-1 (in 32-bit hex words)

due to the observation that in round 2 there is a difference in the most significant bit of register B ($B' = 80000000_x$), but both most significant bits of C and D are zero (where $C = 67452301_x \lll 30$ and $D = EFCDAB89_x \lll 30$). Thus, considering the differences of the messages ($W_2' = 80000001_x$) in that bit, the new content of A must have a difference in this bit, in contrary to the prediction of the disturbance vector.

In order to be able to use the disturbance vector of Table 5, the initial value is replaced by another value by adding an additional first block, which in this case is the whole zero block ($M_1 = M_1^* = 0$). The resultant intermediate hash value is

$$h_1 = (37970DFF_x,\ 5E912289_x,\ C78B3705_x,\ 923B82E9_x,\ CC36E948_x).$$

With this intermediate value h_1, we can now proceed to the next block with the disturbance vector of Table 5. The second block of the collision of the 36-round SHA-1 is presented in Table 6.

Round	$D\&C$	δ_{i+1}	A'_{i+1}	B'_{i+1}	C'_{i+1}	D'_{i+1}
0	80000000	80000000	80000000	00000000	00000000	00000000
1	00000010	00000000	00000000	80000000	00000000	00000000
2	80000001	00000001	00000001	00000000	20000000	00000000
3	20000020	00000000	00000000	00000001	00000000	20000000
4	20000002	**00000003**	**00000001**	00000000	40000000	00000000
5	60000062	00000002	00000002	00000001	00000000	40000000
6	40000042	00000001	00000001	00000002	40000000	00000000
7	80000020	00000002	00000002	00000001	80000000	40000000
8	00000041	40000000	40000000	00000002	40000000	80000000
9	00000008	00000002	00000002	40000000	80000000	40000000
10	00000042	00000002	00000002	00000002	10000000	80000000
11	50000042	80000000	80000000	00000002	80000000	10000000
12	10000010	00000002	00000002	80000000	80000000	80000000
13	90000040	00000000	00000000	00000002	20000000	80000000
14	20000000	00000002	00000002	00000000	80000000	20000000
15	20000040	00000000	00000000	00000002	00000000	80000000

Table 7. Comparison of δ_i and A'_i in the 36-Round Collision (in 32-bit hex words)

A Generalized Test for Conformance The 36-round collision of Table 6 presents an additional change in respect to the attack of SHA-0. In the attack on SHA-0, the intermediate differences A'_i are necessarily equal to δ_i for $i = 1, \ldots, r$, where r is the number of rounds of the analyzed compression function. In SHA-1 this is not the case, since more than a single location of a bit are selected for the disturbances. In cases where there are two or more disturbances or corrections in adjacent bits, it may happen that the more significant bit is not correctly approximated, e.g., the IF function does not output the XOR of its inputs for the particular values of the registers. However, it may happen that the carry of the less significant bit cancels this wrong approximation, resulting with the expected difference $A'_i = \delta_i$. In other cases, a wrong approximation of the less significant bit cancels the correct approximation of the more significant bit, e.g., the addition modulo 2^{32} of the less significant bit changes the carry. In these cases $A'_i \neq \delta_i$, and the difference is in this more significant bit. The difference that the more significant bit expects to create in A'_i is now canceled, but the corrections for this expected difference still exist in the following five rounds. These corrections are now used to correct wrong approximations of the less significant bit which change the carries in the next five rounds. If we are lucky, the less significant bit creates additional differences in the carry, thus corrects the differences in A'_i in the next rounds.

Table 7 shows the differences of first 16 rounds of the compression function in the second block of the 36-round collision (shown in Table 6). In this table the D&C column shows the message difference M', δ_{i+1} shows the expected difference in A'_{i+1}, and the other four columns show the actual difference A'_{i+1}, B'_{i+1}, C'_{i+1}, and D'_{i+1}. The table shows a situation where two disturbances are applied to bit 0 and 1, and the carry change of bit 0 cancel the disturbance of bit 1. The entry of round 4 in the table shows (in boldface) that the expected difference δ_5 is different from the actual value of A'_5. This difference between the expected and actual values is due to a carry change of the disturbance of bit 0 that cancels the difference in bit 1. The five corrections in the next five rounds do not have a disturbance in registers A, B, C, D, nor E, but other properties of the IF and carry overcome the missing difference and ensure correct differences in the following rounds.

We call bits whose difference may differ from the expected value of the characteristic, but whose effect can be canceled immediately afterwards, by the name *T bits*. In some cases a simultaneous modification of a few bits makes a similar effect. We can view T bits as extending the notion of characteristics into differentials in which most information on the intermediate differences is fixed, but a few can have any value, describing several different paths leading to the same differential. There are several T bits in the intermediate differences characteristic of 36-round SHA-1, and also in other characteristics used in this paper.

Due to such cases we extended our program to check for conformance by testing for a generalized kind of differences instead of testing exactly whether $A'_i = \delta_i$.

Consecutive Disturbances in the IF Rounds In the attack on SHA-0 two consecutive disturbances in the first 17 rounds (i.e., rounds $0, \ldots, 16$) have a probability zero to be corrected (see [2]). This limitation forces a higher Hamming weight to occur in the expanded disturbance vector, but an attack is still feasible (i.e., there are still few disturbance vectors that predict collisions, and do not have two consecutive disturbances in the first 17 rounds). We observed that all the disturbance vectors that we could find that predict one-block collisions of SHA-1 reduced to 35 or more rounds have consecutive disturbances, i.e., two disturbances at the same bit locations in two consecutive rounds. Thus, this limitation seems to be much more restrictive in SHA-1. However, this stronger limitation comes with the ability to bypass it by various techniques in some fraction of the cases. The characteristic we use for the collision of 36-round SHA-1 is an example for such a case.

In the following discussion, we first explain the limitation of the two consecutive disturbances in SHA-0, and then we show how they behave in SHA-1. In SHA-0, two consecutive disturbances in rounds i and $i+1$ (in bit 1) create differences in $D^{31'}_{i+4}$ and $C^{31'}_{i+4}$, respectively. The two corrections to these differences are applied to the same bit, thus cancel each other in the approximation leading to no difference in δ_{i+4}. On the other hand, the IF function applied on these two differences, where the difference of $B^{31'}_{i+4}$ is zero, causes the result to be complemented always. Thus, in A'_{i+4} we have a difference with no corrections. With SHA-1 the same arguments apply, but we allow disturbances at any bit location. Thus, we can use the carry bit from another disturbance (or correction) as an additional source of corrections.

The following two examples, which are taken from our 36-round attack, should clarify the above: In the first example we show how a carry can be used as follows: At rounds 4 and 5 there are disturbances in bit 1, from which we expect to get A'_5 and A'_6 equal to $\delta_5 = 00000003_x$ and $\delta_6 = 00000002_x$ respectively, which lead after three rounds to the differences $D'_8 = C0000000_x$ and $C'_8 = 80000000_x$. With these differences the IF function applied on $D_8^{31'}$ and $C_8^{31'}$ always complement the output, but it is never complemented in the approximation. Thus, we have a difference that cannot be corrected. However, in the messages we use the carry from bit 0 at round 4 cancels the disturbance at bit 1 of this round, and therefore the created differences are $A'_5 = 00000001_x$ and $A'_6 = 00000002_x$ (see Table 7). Thus, in round 8 the differences are $C'_8 = 80000000_x$ and $D'_8 = 40000000_x$, which can be corrected by the non-linear behavior of the IF function to fit the approximation.

In the second example we show how the problem of two consecutive disturbances can be bypassed when there is another disturbance in one of a few different locations. In rounds 9 and 10 (see Table 7) we have two consecutive disturbances in bit 1 ($\delta_{10} = 00000002_x$ and $\delta_{11} = 00000002_x$), but in this case there is also a disturbance in round 11 in bit 31 ($\delta_{12} = 80000000_x$). Thus, in round 13 we have $B'_{13} = C'_{13} = D'_{13} = 80000000_x$, which fit the approximation with probability 1/2.

In general, consecutive disturbances in bit j of rounds i and $i+1$ can be corrected, if there is a correction or disturbance in a less significant bit that may change the carry to bit $j-2$ in round $i+4$ (i.e., in bit $j-8$ of δ_{i+3}, bit $j-1$ of δ_{i+2}, or bit $j-1$ of δ_{i+1}, δ_i or of δ_{i-1} where the bit numbers are mod 32), leaving the rest of the differences behave as expected.

4.3 A Two-Block Collision of 40-Round SHA-1

In this section we present a collision of 40-round reduced SHA-1. The best (one-block) characteristic that we could find has 19 disturbances from round 20 to round 39, so the complexity of the attack is expected to be around 2^{57}. However, it is easy to find near-collisions of 40 rounds with only five disturbances from round 20 to 39. Thus, we construct a two-block attack where the first block generate such a near-collision, and the second block uses the difference of the initial value that are created by the first block and generate a collision.

We observe that the hash values of multi-block messages are computed as the sum of the initial value and the states g_i of the compression function before the final addition operations, i.e.,

$$h_n = h_0 + \sum_{i=1}^{n} g_i.$$

Therefore, for colliding pairs of messages the following equation holds

$$\sum_{i=1}^{n} (g_i - g_i^*) = 0,$$

which when the addition is approximated by XOR becomes

$$\sum_{i=1}^{n} g'_i = 0.$$

Therefore, when searching for multi-block collisions it may be best to find characteristics for which this sum is zero, and verify that all the other requirements are satisfied, rather than vice versa.

In the particular case of a two-block collision this equation means that $g'_1 = g'_2$, i.e., the two disturbance vectors should have same differences in the last five rounds. This leads to the question why should we use different disturbance vectors for both blocks. The answer would be that the initial value difference of the second block is necessarily different than of the first block (as $h'_0 = 0$ and $h'_1 \neq 0$), where the initial value is related to the difference of the first five rounds of the disturbance vector (rounds $-5, \ldots, -1$). But this is only a partial answer, as we can extend the technique (using for example T bits, with similarities to the extension of Section 3 in the case of SHA-0, but with much more flexibility), and use a disturbance vector whose first five rounds are different than the initial value difference (in the second block). Once we say that, we observe that in the case of the disturbance vector that we use for the first round, the intermediate value h'_1 fits as a replacement initial difference for the same disturbance vector, i.e., if we replace rounds $-5, \ldots, -1$ of the disturbance vector by the last five rounds from the first block, we still get differences that can be corrected later by the disturbance vector. In terms of characteristics, this means that we have two characteristics with different input differences, but same message differences and output differences (and that in most of the rounds they have the same intermediate differences).

Table 8 describes the disturbance vector we use for this attack. This disturbance vector is the same vector used in our 34-round collision (Table 1) rotated by 28 bits to the left and expanded to 40 rounds. In the first five rounds ($-5, \ldots, -1$) of the disturbance vector the differences are zero, and in the last five rounds they have two active bits (these rounds are marked in parentheses). Therefore, we expect that h'_1 will have two active bits in these locations (up to the rotation by 30 bits), so the disturbance vector for the next block should have the first five rounds with the same differences as given in parentheses in the table. Now, we observe that when we replace the first five rounds of the same disturbance vector with the values in parentheses (see Table 9) we still receive a correctable result. The disturbance vector itself, from round 0 to round 39 is unchanged, thus the modified five rounds do not fit to the expansion function of SHA-1, but as these difference come from the initial value, they are not calculated anyway by this expansion. These values should only ensure that the probability of the rounds in which they participate (as A, B, C, D, or E) is greater than zero, and this is the case with these replaced differences.

We would also wish to add that the change of the initial rounds of the disturbance vector can be even extended to a few additional rounds, as long as the message differences remain unchanged, i.e., it would be possible to expect for

Rnd	D.Vec	D&C	Rnd	D.Vec	D&C	Rnd	D.Vec	D&C
−5	00000000		10	00000000	0C000004	25	00000000	00000000
−4	00000000		11	00000000	2C000000	26	00000000	08000000
−3	00000000		12	00000000	08000000	27	00000000	08000000
−2	00000000		13	00000000	08000000	28	00000000	00000000
−1	00000000		14	20000000	28000000	29	00000000	00000000
0	20000000	20000000	15	00000000	00000004	30	00000000	00000000
1	00000000	00000004	16	00000000	20000000	31	00000000	00000000
2	20000000	00000000	17	00000000	08000000	32	00000000	00000000
3	00000000	08000004	18	00000000	08000000	33	00000000	00000000
4	20000000	08000000	19	00000000	08000000	34	40000000	40000000
5	00000000	00000004	20	20000000	20000000	35	(00000000)	00000008
6	30000000	18000000	21	00000000	00000004	36	(00000000)	40000000
7	00000000	00000006	22	20000000	00000000	37	(80000000)	90000000
8	00000000	38000000	23	00000000	08000004	38	(40000000)	50000010
9	20000000	24000000	24	00000000	28000000	39	(00000000)	90000008

Table 8. The Disturbance Vector Used for the Two-Blocks Collision of 40-round SHA-1 (in 32-bit hex words)

Round	First Block D.Vec	Second Block D.Vec	Common D&C
−5	00000000	(00000000)	
−4	00000000	(00000000)	
−3	00000000	(80000000)	
−2	00000000	(40000000)	
−1	00000000	(00000000)	
0	20000000	20000000	20000000
1	00000000	00000000	00000004
2	20000000	20000000	00000000
3	00000000	00000000	08000004
4	20000000	20000000	08000000
5	00000000	00000000	00000004
6	30000000	30000000	18000000
7	00000000	00000000	00000006
8	00000000	00000000	38000000
9	20000000	20000000	24000000
⋮	⋮	⋮	⋮

Table 9. The Beginning of Both Blocks of the Disturbance Vector Used for 40-round SHA-1 (in 32-bit hex words)

Message 1, block 1:							
404B674C	B70CB385	D2DDAC0D	3A0E9BD3	CA7F1780	7FEFDA17	05E43AF2	444344C2
641A2CB6	86C2CFE6	EBCDEF67	6577E095	1A9CAD10	CFE48484	78639157	B13B759A
Message 2, block 1:							
604B674C	B70CB381	D2DDAC0D	320E9BD7	C27F1780	7FEFDA13	1DE43AF2	444344C4
5C1A2CB6	A2C2CFE6	E7CDEF63	4977E095	129CAD10	C7E48484	50639157	B13B759E
Message 1, block 2:							
E63C47F7	0AB5F259	47DE1E6B	09E06877	6229CC42	604CF1AB	9B14B8F3	7261186C
1A5370F9	822E13EB	FB7157EF	6B0919C5	1F3D744B	FA4DE198	FBB10C06	FDA3C3E9
Message 2, block 2:							
C63C47F7	0AB5F25D	47DE1E6B	01E06873	6A29CC42	604CF1AF	8314B8F3	7261186A
225370F9	A62E13EB	F77157EB	470919C5	173D744B	F24DE198	D3B10C06	FDA3C3ED

Table 10. The Two-Block Collision of 40-Round SHA-1 (in 32-bit hex words)

different values in round 0 (or even 1) of the disturbance vector when changing the initial five rounds, but without changing the message differences. Also, it is possible to make replacements in the last few rounds. This phenomena is similar to the usual technique of differential cryptanalysis, where iterative characteristics are used with modified first and last rounds, allowing even larger probabilities than in the full iterative case.

The messages of the 40-round collision are presented in Table 10. The output difference h_1' of the compression function of the first block becomes the input difference entering the second application. These intermediate differences can be corrected by the same message difference that we use in the first block. Thus, by using the same message difference in the second block the difference of the intermediate value is corrected. We expect to get $g_2' = h_1'$ (i.e., the differences in the registers after the last round of the compression function are equal to the intermediate value differences), which with probability 1/4 cancels the differences after the final addition of $h_2 = g_2 + h_1$.

5 Strength of Reduced Versions of SHA-1 with More Rounds

SHA-1 with more than 40 rounds is also vulnerable to the attacks described in this paper. Though all the disturbance vectors that we found have consecutive disturbances in the first 17 rounds, many of them contain correctable consecutive disturbances. We therefore list here two set of results: the first is the results for SHA-1 reduced to fewer rounds, where these rounds are set at the first rounds of SHA-1, i.e., the first 20 rounds use the IF function. This case is denoted later by SHA-1. The second set of results, denoted later by NO-IF, have consecutive disturbances, so if the reduced version starts with 20 IF rounds, the probability of success is reduced to 0, but if the reduced version of SHA-1 starts at a different

Rounds	SHA-1			NO-IF		
	HW	2B	NC	HW	2B	NC
34	**2**			2		
35	7	6	3	**4**	5	3
36	**7**	**3**	3	5	3	3
37	11	9	3	**5**	5	3
38	12	7	4	8	6	3
39	12	11	5	8	8	4
40	19	**5**	5	11	5	5
41	17	14	6	12	10	6
42	17	14	7	13	11	7
43	17	15	8	17	13	7
44	19	17	9	15	15	8
45	25	16	10	15	15	10
46	25	18	10	23	13	10
47	26	23	12	24	21	11
48	28	25	13	14	14	13
49	32	22	15	14	14	14
50	35	29	16	14	14	14
51	38	26	19	15	15	15
52	42	32	19	16	16	15
53	42	32	20	16	16	16
54	39	42	24	36	34	16
55	39	48	27	39	38	16
56	41	39	28	41	29	16
57	61	56	29	42	23	17
58	58	52	29	42	17	17
59	64	53	29	51		17
60	45	45		29		18
61	45	38		30		19

Table 11. The Hamming Weights of the Best Disturbance Vectors that We Found (Counted from Round 20)

location, the attack is still possible (such as when the reduced version contains the last rounds of SHA-1, rather than the first ones).

Table 11 lists the results for 34 up to 61 rounds. For each number of rounds, and each set of results (SHA-1 or NO-IF) the table lists the Hamming weight of the disturbance vector from rounds 20 and on for three cases: the first, marked by HW, is the Hamming weight of the best disturbance vector predicting a one-block collision we found. The second, marked by 2B, is the best disturbance vector predicting a two-block collision, and the last, marked by NC is the best disturbance vector predicting a near-collision. Entries that we used to actually find a collision are marked in boldface.

The complexities of the attacks that use the mentioned disturbance vectors can be approximated by $2^{3\mathrm{HW}}$, where HW is the Hamming weight of the disturbance vector from round 20 and on (i.e., the value in the table). The exact complexity may vary (between 2^{2HW} to 2^{4HW}) by some factor which depends on the exact functions (IF, MAJ, XOR) used, by the rounds where the disturbances occur, and by a few additional details.

We can thus see that entries with up to about 26 Hamming weight predict a collision with complexity (slightly) faster than the generic birthday attack (as $2^{3\cdot 26} = 2^{78} < 2^{80}$). We marked the location of this threshold by underlines. Hamming weights much smaller than 26 predict much more practical complexities, and as can be seen from the table, Hamming weights up to about 10 require only a short computation on a personal computer (all the found collisions marked in boldface were found within a few seconds of computation).

It is especially interesting to see the huge increase of the Hamming weight in the case of NO-IF after 53 rounds, where the Hamming weight of 53 rounds is 16 and of 54 rounds is 36. Similarly in the two-block attack the Hamming weight is 17 for 58 rounds. Thus, we expect that one-block collisions of 53-round reduced SHA-1 can be found with complexity about 2^{60}, and two-block collisions of 58-round SHA-1 can be found with complexity about 2^{75} (this is a more accurate approximation than 2^{3HW} for this case), where the reduction is to the last 53 (respectively 58) rounds of SHA-1, but we have no hope according to the table to find one-block collisions of 54-round reductions. In the case of the first rounds of SHA-1, the maximal number of rounds according to the table is 51 using the two-block technique, but the complexity of this attack would be only marginally faster than the birthday attack (though much easier to parallelize).

We are now working on improvements for further rounds, some of them are by applications of the techniques described in this paper in more complex ways, and some using new ideas. Note that the NC column is a lower bound for any multiple-block attack, thus we see that there is still some hope for the attacker to find better results.

In particular, we succeed to show that the NO-IF figures hold also for the case of the first rounds of SHA-1 (starting with IF rounds) by using different characteristic paths for the first rounds, but leaving the same input, output, and message differences.

6 Summary

This paper presents various attacks on reduced versions of SHA-0 and SHA-1 along with various techniques for the analysis of hash functions. These techniques, along with the neutral bit technique and other prior techniques, form a set of tools that enable practical attacks on the full SHA-0, and reduces the complexity of attacking SHA-1 reduced to 58 or fewer rounds to less than the complexity of the birthday attack.

As this work is still in progress, we expect to further improve some of the attacks presented in this paper, and to incorporate several new ideas that may increase the total number of rounds that we can attack, such as three-block attacks and attacks with more than three blocks. In particular, it is possible to use the 53-round and the 58-round attacks on SHA-1 even against the first 53 and 58 rounds.

Finally we observe that a search for one-block near-collisions is easier than search for one-block collisions, as when searching for near-collisions, there is no need to fix the initial value of the compression function, but instead it is possible to fix an intermediate value, and search backwards in the direction of the initial value, and then forward for the output. In such a search, we found that the number of neutral bits is much larger than in the regular case, thus allowing to increase the number of rounds that we get for free from about 20–22 rounds to about 30 rounds, thus decreasing the number of rounds that should be analyzed by the probabilistic stage. Moreover, it is possible to select the 30 rounds to be the 30 consecutive rounds with the lowest probability in the characteristic, thus increasing the probability even further. For example, with such a technique it is possible to find pseudo-collisions of the full SHA-0 with probability about 2^{30}–2^{33}.

References

1. Eli Biham, Rafi Chen, *Near-Collisions of SHA-0*, Advances in Cryptology, proceedings of CRYPTO 2004, LNCS 3152, pp. 290–305, Springer Verlag, 2004.
2. Florent Chabaud, Antoine Joux, *Differential Collisions in SHA-0*, Advanced in Cryptology, proceedings of CRYPTO '98, LNCS 1462, pp. 56–71, Springer Verlag, 1999.
3. National Institute of Standards and Technologies, *Secure Hash Standard*, Federal Information Processing Standards Publication, FIPS-180, May 1993.
4. National Institute of Standards and Technologies, *Secure Hash Standard*, Federal Information Processing Standards, Publication FIPS-180-1, April 1995.
5. Ron Rivest,*The MD4 Message-Digest Algorithm*, Network Working Group, Request for Comments:1186, October 1990.
6. Ron Rivest, *The MD5 Message-Digest Algorithm*, Network Working Group, Request for Comments:1321, April 1992.
7. Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu, *Cryptanalysis for Hash Functions MD4 and RIPEMD*, these proceedings.
8. Xiaoyun Wang, Hongbo Yu, *How to Break MD5 and Other Hash Functions*, these proceedings.
9. V. Rijmen, E. Oswald. Update on SHA-1. In *RSA Crypto Track 2005*, LNCS 3376, 2005.