Near-Collisions of SHA-0

Eli Biham Rafi Chen

Computer Science Department
Technion – Israel Institute of Technology
Haifa 32000, Israel
Email: biham@cs.technion.ac.il, rafi_hen@cs.technion.ac.il
WWW: http://www.cs.technion.ac.il/~biham/

Abstract. In this paper we find two near-collisions of the full compression function of SHA-0, in which up to 142 of the 160 bits of the output are equal. We also find many full collisions of 65-round reduced SHA-0, which is a large improvement to the best previous result of 35 rounds. We use the very surprising fact that the messages have many neutral bits, some of which do not affect the differences for about 15–20 rounds. We also show that 82-round SHA-0 is much weaker than the (80-round) SHA-0, although it has more rounds. This fact demonstrates that the strength of SHA-0 is not monotonous in the number of rounds.

Breaking News: Forthcoming Results on SHA-0 and SHA-1

After this paper was written, we made various improvements to the techniques mentioned in this paper. These new results will appear in forthcoming papers.

Among other things, the new results show that SHA-0 should not be used in applications where collisions are a threat. They also include a method to use near-collisions as a tool in order to find collisions. This method is especially useful in cases where the original technique cannot find collisions (such as 50-round SHA-0).

Our new improvements also allowed us to analyze reduced variants of SHA-1. We successfully found millions of collisions of 34-round reduced SHA-1, some of which are formed of ASCII letters only, or contain partial English text. We also found a collision of 36-round reduced SHA-1, and near-collisions of 45-round reduced SHA-1. We expect to be able to find collisions and near-collisions of longer variants.

At the current state of research, our attacks are not expected to break the full SHA-1.

1 Introduction

SHA-0 is a cryptographic hash function, which was issued as a Federal Information Processing Standard (FIPS-180) by NIST in 1993 [8]. It is based on the principles of MD4 [12] and MD5 [13]. The algorithm takes a message of any length up to 2^{64} bits and computes a 160-bit hash value. A technical revision, called SHA-1, which specifies an additional rotate operation to the algorithm, was issued as FIPS-180-1 [9] in 1995. The purpose of the revision according to NIST is to improve the security provided by the hash function.

Finding collisions of hash functions is not an easy task. The known cases of successful finding of collisions (such as the attack on Snefru [14, 2], and the attack on MD4 [12, 4]) are rare, and use detailed weaknesses of the broken functions. It is widely believed that finding near-collisions (i.e., two messages that hash to almost the same value, with a difference of only a few bits) are as difficult, or almost as difficult, as finding a full collision. The Handbook of Applied Cryptography [7] defines near-collision resistance by

> *near-collision resistance.* It should be hard to find any two inputs x, x' such that $h(x)$ and $h(x')$ differ in only a small number of bits.

and states that it may serve as a certificational property. In some designs of hash functions, such as SHA-2/224 [10], SHA-2/384 [11], and Tiger [1], the designers that wish to allow several hash sizes for their design, base the version with the smaller size on the one with the larger size, and discard some of the output bits, thus showing the confidence of the designers in the difficulty of finding near-collisions. Near-collisions were also used in the cryptanalysis of MD4 [15, 4]. Near-collisions are the simplest example of forbidden relations between outputs of the hash function. Another proposed forbidden relation of the hash results is *division intractability* [5] where finding messages hashed to a divisor of other hashes should be difficult.

In [3] Chabaud and Joux proposed a theoretical attack on the full SHA-0 with complexity of 2^{61}. Using their technique they found a collision of SHA-0 reduced to 35 rounds.

In this paper we improve over the results of [3], and present attacks with lower complexities. We present collisions of 65-round reduced SHA-0, and near-collisions of the full compression function of SHA-0 in which up to 142 of the 160 bits of the hash value are equal. We use the very surprising observation that many bits of the message are neutral bits, i.e., they do not affect the differences of the intermediate data for 15–20 rounds. We observe that the strength of SHA-0 is not monotonous, i.e., collisions of 82 rounds are easier to find than of 80 rounds, and use it in our search for near-collisions. We also present several observations on variants of SHA-0.

A comparison of Chabaud and Joux' results with our results is given in Table 1.

Table 2 shows the complexity of finding collisions of reduced and extended SHA-0, as a function of the number of rounds. The table demonstrates that the strength of SHA-0 is not monotonous with the number of rounds. In the complexity calculations we assume that for the extended SHA-0, the additional rounds after the original 80 rounds are performed with the f_i function being XOR, like in rounds $60, \ldots, 79$ that preceed them. We also assume that the first 22 rounds can be gained for free by using the neutral bits.

	Chabaud and Joux		Our Results	
	Rounds	Complexity	Rounds	Complexity
Optimized for	80	2^{61}	82	2^{43}
Best collision found	35	2^{14}	65	2^{29} (*)
Conforming rounds found	≈ 56 [6]	–	76	2^{40} (**)
Near collisions (18-bit diff)	–	–	80	2^{40} (**)

(*) About half an hour on a PC
(**) Our actual search took less than a day on a PC which is equivalent to a complexity of 2^{35}

Table 1. Comparison of Chabaud and Joux' Results to Our Results

Number of Rounds	Complexity	Number of Rounds	Complexity
64	2^{29}	80	2^{56}
65	2^{29}	81	2^{43}
68	2^{43}	82	2^{43}
74	2^{50}	83	2^{65}
75	2^{52}	84	2^{64}
76	– (*)	85	2^{71}
77	2^{66}	86	2^{72}
78	2^{56}	87	– (*)
79	2^{56}	92	2^{74}

(*) There is no disturbance vector for which the differences of the five registers after 50, 76 or 87 rounds are zero and which no do not have consequent disturbances in the first 17 rounds

Table 2. The Complexity of Finding Collisions of Reduced/Extended SHA-0

A comparison between finding near-collisions using a generic attack and our attack is given in Table 3. Note that the generic attack hashes a large number of random messages, all of them are then kept in memory. Due to the birthday paradox, it is expected to have a collision or near-collision with complexity (number of messages) about

$$1.17\sqrt{2^{160} \Big/ \binom{160}{k}},$$

Number of Diff. Bits	0	1	2	3	4	5	18
Generic (time & memory)	2^{80}	2^{76}	2^{73}	2^{70}	2^{68}	2^{65}	2^{41}
Ours (time, negligible memory)	2^{56}	2^{43}	2^{43}	2^{42}	2^{42}	2^{42}	2^{40}

Table 3. The Complexities of Finding Near-Collisions of the Compression Function of SHA-0 by a Generic Attack and by Our Attack (the number of different bits is the Hamming distance of the five registers before the feed-forward)

where k is the Hamming weight of the difference. As this attack is generic, it uses no special properties on SHA-0, and thus cannot be used to gain insight on its design.

This paper is organized as follows: Section 2 describes the SHA-0 algorithm, and a few notations. Section 3 describes the attack of Chabaud and Joux. Our improved attack is presented in Section 4. Two pairs of near-collisions of the compression function of SHA-0 and full collision of 65-round reduced SHA-0 are given in section 5. Section 6 describes small variations of SHA-0 that largely affect its security. Finally, Section 7 summarizes the paper.

2 Description of SHA-0

SHA-0 hashes messages of any length in blocks of 512 bits, and produces a message digest of 160 bits.

1. The message is padded with a single bit '1', followed by 0–511 bits '0', followed by a 64-bit representation of the message length, where the number of zeroes is selected to ensure the total length of the padded message is a multiple of 512 bits. The padded message is divided to 512-bit blocks $M_1, \ldots, M_n$.
2. A 5-word buffer h_0 is initialized to

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

3. Each block M_j in turn is subjected to the compression function, along with the current value of the buffer h_{j-1}. The output is a new value for h_j:

$$h_j = \text{compress}(M_j, h_{j-1}).$$

4. h_n is the output of the hash function.

The compression function is:

1. Divide the 512-bit block M_j to 16 32-bit words $W_0, W_1, \ldots, W_{15}$.
2. Expand the 16 words to 80 words by the recurrence equation:

$$W_i = W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}, \qquad i = 16, \ \ldots, \ 79. \qquad (1)$$

We denote expansion of a block to 80 words by this equation by $\exp(\cdot)$, and note that $W = \exp(M_j)$.

3. Divide h_{j-1} to the five registers A, B, C, D, and E by

$$(A_0, \ B_0, \ C_0, \ D_0, \ E_0) = h_{j-1}$$

4. Iterate the following round function 80 times ($i = 0, \ldots, 79$)

$$A_{i+1} = (W_i + \text{ROL5}(A_i) + f_i(B_i, C_i, D_i) + E_i + K_i) \bmod 2^{32}, \qquad (2)$$

$$B_{i+1} = A_i, \quad C_{i+1} = \text{ROL30}(B_i), \quad D_{i+1} = C_i, \quad E_{i+1} = D_i,$$

where the functions and constants used in each round are described in Table 4.

Rounds	$f_i(B,C,D)$	K_i
$0 \leq i \leq 19$	$BC \vee \bar{B}D$	$5A827999_x$
$20 \leq i \leq 39$	$B \oplus C \oplus D$	$6ED9EBA1_x$
$40 \leq i \leq 59$	$BC \vee BD \vee CD$	$8F1BBCDC_x$
$60 \leq i \leq 79$	$B \oplus C \oplus D$	$CA62C1D6_x$

Table 4. Functions and Constants

5. The output of the compression function is

$$h_j = (A_0 + A_{80},\quad B_0 + B_{80},\quad C_0 + C_{80},\quad D_0 + D_{80},\quad E_0 + E_{80}).$$

In the remainder of the paper we consider only 512-bit messages and only the first application of the compression function. We denote the j'th bit of W_i by W_i^j, and similarly we denote the j'th bits of A_i, B_i, C_i, D_i, and E_i by A_i^j, B_i^j, C_i^j, D_i^j, and E_i^j. We also use the notation f_i to denote the output of $f_i(B_i, C_i, D_i)$ in round i, and f_i^j denotes the j'th bit of f_i.

3 Description of Chabaud and Joux Attack

In the attack of Chabaud and Joux [3] messages are constructed with specific differences, such that the effect of the differences of the messages on the difference of the registers A, ..., E can be canceled within a few rounds. The cancellation is performed by applying correcting patterns by additional differences in the messages.

The attack is initiated by a selection of a difference Δ, that is later used as the difference of the two colliding messages. The difference is selected with various disturbances and corrections, where the corrections are additional differences used to correct the differences caused by the disturbances. The disturbances are always selected in bit 1 of the message words. Due to the rotations by 5 and 30 bits in the round function, corrections are made in bits 1, 6, and 31 of the words. These disturbances and corrections are aimed to limit the evolution of differences to other bits. The result is that in an expected run, A_i and A'_i can only differ in bit 1 (i.e., $A_i \oplus A'_i \in \{0, 00000002_x\}$), and each time they differ, they cause differences in the other registers in the following rounds, which are then corrected by differences of the messages (or W's).

A disturbance starts by setting bit 1 in one of the input words of M' as the complement of the corresponding bit of M. We now show how applying a correction sequence on bits 6, 1, 31, 31, 31 on the following words may cancel the differences at the end of the sequence. Suppose the initial disturbance is in $W_i^1 \neq W_i'^1$. This input difference causes registers A and A' to differ at bit 1. On each consequent round the difference moves to the next register (B, C, D or E), while the corrections of bits 6, 1, 31, 31, 31 in the input words $W'_{i+1}, \ldots W'_{i+5}$, respectively, keep registers A and A' equal in these rounds. After this sequence of a single disturbance and five corrections, the registers' contents are equal. By

Round		Disturbance i	Correction Rounds $i+1$	$i+2$	$i+3$	$i+4$	$i+5$
Input	W^1	$0 \to 1$		$1 \to 0$			
Words	W^6		$1 \to 0$				
	W^{31}				$0 \leftrightarrow 1$	$0 \leftrightarrow 1$	$0 \leftrightarrow 1$
Desired	A^6		$0 \to 1$				
Results	f^1			$0 \to 1$			
	f^{31}				$0 \leftrightarrow 1$	$0 \leftrightarrow 1$	
	E^{31}						$0 \leftrightarrow 1$
Registers		A^1	B^1	C^{31}	D^{31}	E^{31}	
Differences		$0 \to 1$	$0 \to 1$	$0 \to 1$	$0 \to 1$	$0 \to 1$	None

Table 5. Single Difference and Corrections

generating M' from M by applying this mask, and calculating the difference of A and A' at each round we can get the differences described in Table 5 with a non negligible probability. The table describes a disturbance with $W_i^1 = 0$ and $W_i'^1 = 1$, and the required corrections. A similar disturbance and corrections can be applied for a '1' to '0' difference. The notation $0 \to 1$ refer to a change where a bit is '0' in W and '1' in W'. The notation $0 \leftrightarrow 1$ means that there is a change either from '0' to '1' or from '1' to '0'.

Let $\mathcal{D}$ be a vector of 80 words, which correspond to the 80 rounds of the compression function. Each word in the vector is set to '1' if there is a disturbance in the corresponding round, and is set to '0' otherwise. We call this vector the *disturbance vector*. Since getting a collision for the full function requires five correcting rounds, full collisions require the last five words of the disturbance vector to be zero (but for near-collisions this property is not required). Let $\mathrm{SR}^l(\mathcal{D})$ be the vector of 80 words received by prepending l zero words to the first $80 - l$ words of $\mathcal{D}$ (i.e., a non-cyclic shift operation of the words). Then, the corrections are made in bit 6 in the rounds which correspond to non-zero words in $\mathrm{SR}^1(\mathcal{D})$, in bit 1 in $\mathrm{SR}^2(\mathcal{D})$, and in bit 31 in $\mathrm{SR}^3(\mathcal{D})$ and $\mathrm{SR}^4(\mathcal{D})$ and $\mathrm{SR}^5(\mathcal{D})$. Thus, the expansion of $\varDelta$ to 80 round can be written in the form

$$\begin{aligned}\exp(\varDelta) = &\big(\big(\mathcal{D} \oplus \mathrm{SR}^2(\mathcal{D})\big) \lll 1\big) \oplus \\ &\big(\ \mathrm{SR}^1(\mathcal{D}) \lll 6\big) \oplus \\ &\big(\big(\mathrm{SR}^3(\mathcal{D}) \oplus \mathrm{SR}^4(\mathcal{D}) \oplus \mathrm{SR}^5(\mathcal{D})\big) \lll 31\big)\,,\end{aligned}$$

where $\lll$ denotes shift of each word of the vector separately. In addition, since $\exp(\varDelta)$ is expanded by the linear feedback shift register of Equation (1), the disturbance vector $\mathcal{D}$ is also generatable by this linear feedback shift register. See [3] for additional details on the attack, and the additional required constraints.

We expect that the value of $A_{i+1} \oplus A'_{i+1}$ be $\mathcal{D}_i \lll 1$ if all the corrections succeed (i.e., only disturbances in the current round affect the difference after the round). Thus, the vector of the expected values of $(A_{i+1} \oplus A'_{i+1})_{i=0,\ldots,79}$, which we denote by δ is

$$\delta = \mathcal{D} \lll 1$$

(note that the indices of δ are 1, ..., 80, rather than 0, ..., 79).

As the correction process is probabilistic, and assuming each disturbance has the same probability for correction, we are interested in the disturbance vector with the least Hamming weight for getting the least search complexity (but note that the correction probabilities vary, and depend on the f_i's used in the correction rounds).

4 Our Improved Attack

Our attack is based on the attack of Chabaud and Joux with enhancements that increase the probability of finding collisions and near-collisions.

The main idea is to start the collision search from some intermediate round, thus eliminating the probabilistic behavior of prior rounds. In order to start the collision search from round r, we build a pair of messages M and M' with a difference $M \oplus M' = \Delta$, and with the two additional properties described below. Before we describe these properties we wish to make the following definitions:

Definition 1. Given the difference Δ of two messages, the attack of Chabaud and Joux defines the expected differences δ of the values of register A in each round. We say that a pair of messages *conforms to* δ_r if $A_i \oplus A'_i = \delta_i$ for every $i \in \{1, \ldots, r\}$ (which means that the differences at the output of the first r rounds 0, ..., $r-1$ are as expected).

Definition 2. Let M and M' be a pair of messages that conforms to δ_r for some $r \geq 16$. We say that the i'th bit of the messages ($i \in \{0, \ldots, 511\}$) is a *neutral bit* with respect to M and M' if the pair of messages received by complementing the i'th bits of M and M' also conform to δ_r. We say that the pair of the i'th and j'th bits is *neutral* with respect to M and M' if all the pairs of messages received by complementing *any* subset of these bits ($\{i\}$, $\{j\}$, or $\{i, j\}$) in both messages M and M' also conform to δ_r. We say that a set of bits $S \subseteq \{0, \ldots, 511\}$ is *neutral* with respect to M and M' if all pairs of messages received by complementing *any* subset of the bits in S in both messages M and M' also conform to δ_r. We say that a subset $S \subseteq \{0, \ldots, 511\}$ of the bits of the messages is a *2-neutral set* with respect to M and M' if every bit in S is neutral, and every pair of bits in S is neutral.

We denote the size of the maximal 2-neutral set (for given messages and r) by $k(r)$. We are now ready to describe the two additional properties:

1. The message pair conforms to δ_r. Having the required sequence of $A \oplus A'$ implies that all other differences (i.e., $B \oplus B'$, $C \oplus C'$, $D \oplus D'$, $E \oplus E'$) are also as required.
2. The message pair has a large-enough 2-neutral set of bits. We expect that a large fraction of the subsets of the bits in the 2-neutral set are also neutral.

Given a pair of messages with these properties, we can construct a set of $2^{k(r)}$ message pairs by complementing subsets of the bits of the 2-neutral set. Since a large fraction of these pairs conform to δ_r, while the probability of random pairs is much smaller, it is advisable to use these pairs for the attack.

How r and $k(r)$ are determined? Starting the search from round r we can calculate the probability

$$p(r) = \prod_{i=r}^{79} p_i$$

of successful corrections in all the rounds given messages that conform to δ_r (where p_i is the probability of successful corrections in round i, or 1 if no correction is performed). When the disturbance vector has zeroes at the last five rounds, $p(r)$ is the probability for getting a collision (otherwise, a near-collision is expected). The number of conforming pairs we need to test is expected to be about $1/p(r)$. Since every subset of $k(r)$ neutral bits can be used, we can try $2^{k(r)}$ pairs using with these bits. Thus, we should select r that satisfies $2^{k(r)} \geq 1/p(r)$. In fact, we select the largest r that satisfies this inequality.

4.1 Finding 2-Neutral Sets of Bits of a Given Pair

The following algorithm finds a 2-neutral set of bits. The input to the algorithm is a pair of messages M, M' with a difference Δ that conforms to δ_r. The algorithm generates 512 candidate pairs by complementing single bits in M, M' (leaving their difference unchanged). Let e_i, $i \in \{0, \ldots, 511\}$, denote a message whose value has a single bit '1', and 511 bits '0', where the bit '1' is in the i'th location. The candidate pairs can be written by

$$(M \oplus e^i, M' \oplus e^i), \qquad\qquad i \in \{0, \ldots, 511\}.$$

Each candidate pair is tested to conform to δ_r. If a candidate pair conforms to δ_r, then bit i is a neutral bit.

In order to find a 2-neutral set of bits we define a graph whose vertices correspond to the neutral bits. We then add an edge for each pair of bits whose simultaneous complementations does not affect conformance. This graph describes all the bits whose complementation does not affect conformance, and all the pairs of these bits whose simultaneous complementations does not affect conformance. We are now interested to find the maximal clique (or an almost maximal clique) in this graph, i.e., the maximal subset of vertices for which any vertex in the subset is connected to any other vertex in the subset by an edge. Although in general finding a maximal clique is an NP-complete problem, in our case finding a large enough clique is not difficult, as many vertices are connected to all other vertices by edges.

We are now ready to make some very important observations, on which the success of our attack is based:

Observation 1. When we perform a search with the set of $2^{k(r)}$ message pairs, about $1/8$ of the pairs (i.e., about $2^{k(r)-3}$ pairs) conform to δ_r.

M=	$19EF75A8_x$	$D2F24D9A_x$	$8F179A7D_x$	$1A295690_x$
	$2E84C143_x$	$D74B9DDC_x$	$18C10577_x$	$8107056E_x$
	$5B1A47ED_x$	$6212C3F2_x$	$3B2D04F8_x$	$F5581AB0_x$
	$26D8CDBC_x$	$AB3A3248_x$	$F347E871_x$	$46278F39_x$
M'=	$19EF75A8_x$	$D2F24D9A_x$	$8F179A7D_x$	$1A295692_x$
	$2E84C103_x$	$D74B9DDE_x$	$98C10577_x$	$0107056E_x$
	$DB1A47EF_x$	$6212C3B2_x$	$3B2D04F8_x$	$75581AF0_x$
	$A6D8CDBE_x$	$AB3A324A_x$	$7347E831_x$	$C6278F3B_x$
Singles:	388 457	458 459	464 484	485 489
	490 491	494 495	496 499	501 506
	507			
Pairs:	301 264	461 424	493 456	497 460
	500 463	502 428		
Triplets:	296 175 138	341 220 183	376 255 218	386 265 228
	391 270 233	462 426 425	466 429 393	488 483 478
	492 334 297			
Quadru-	229 137 108 71	331 210 116 79	364 338 337 300	
plets:	455 435 434 397	505 437 431 400		
Quintu-	471 470 469 433 395		487 465 344 343 306	
plets:	504 480 451 438 420			

Table 6. A Pair of Messages with 40 Neutral Bits and Simultaneous Neutral Bits for $r = 22$ (the bits are numbered in the range $0, \ldots, 511$)

Let

$$p(r \to r') = \prod_{j=r}^{r'-1} p_i$$

be the probability that a pair that conforms to δ_r also conforms to $\delta_{r'}$, and notice that $p(r) = p(r \to 80)$.

Observation 2. Let r and r' be some rounds where $p(r \to r') \approx 2^{-k(r)}$. By trying the $2^{k(r)}$ generated message pairs, we get the expected number of pairs conforming to $\delta_{r'}$, but surprisingly a fraction of the pairs that conform to $\delta_{r'}$ also conform to $\delta_{r'+l}$, which we would expect to get with a larger set of about $2^{k(r)+\alpha}$, where $2 \le l \le 4$ and $3 \le \alpha \le 8$.

In the actual attack we improve the algorithm further by searching for pairs of non-neutral bits whose simultaneous complementation create pairs that also conform to δ_r (and similarly search for triplets of bits, or larger sets of bits). Using this method we receive a larger number of neutral "bits" that can be used for our analysis with higher rounds.

An example of a pair of messages with its neutral set of bits, is given in Table 6. In this example $r = 22$ and the size of the neutral set is $k(22) = 40$. In particular, the quadruplet 229 137 108 71 consists of bits of rounds 7, 4, 3, and 2, so the changes at round 2 are successfully corrected by the changes in the other rounds so the difference is unaffected for 20 rounds, and even from round 7 there are 15 additional rounds whose difference is not affected.

Observation 3. In many cases pairs of bits that are simultaneously neutral, but each bit is not, are of the form W_i^j, $W_{i-l}^{(j-5l) \bmod 32}$ for small l's. Similarly triplets (and quartets, etc.) of non-neutral bits, whose simultaneous complementation is neutral are of the same form, i.e., W_i^j, and $W_{i-l}^{(j-5l) \bmod 32}$ for two different small l's. We call such sets of bits *simultaneous-neutral sets*, and in case of pairs of bits *simultaneous-neutral pairs.*

4.2 Finding a Pair With a Larger 2-Neutral Set

For the attack, we are interested in finding a message pair with a maximal 2-neutral set of bits. Assume that we are already given a pair conforming to δ_r. We are now modifying this pair slightly in order to get another pair that conforms to δ_r with a larger 2-neutral set of bits.

This algorithm takes the given message pair as a base, modifies it in a certain way that we describe later, and calls the algorithm that finds the 2-neutral set of the new pair. If the size of this set is larger than the set of the base pair, the base pair is replaced by the new pair, and the algorithm restarts with the new pair as the base.

By modifying the current message pair we create a new pair that hopefully conforms to δ_r. The modifications are made in bits that maximize the probability of success. In order to create a new conforming pair, we modify several neutral bits (and simultaneously-neutral sets of bits), and check whether the resultant pair conforms to δ_r.

In some cases we can improve further. In rounds where bit 1 differs, i.e., $W_i^1 \neq W_i'^1$, the carry from bit 1 to the next can create a difference in the next bit. The probability for this carry to make this difference is 1/2. In such case $A_{i+1} \oplus A'_{i+1} \neq 00000002_x$, and thus the new pair does not conform to δ_r.

Observation 4. If the differences of the carry is changed, the change can be canceled by complementing W_i^0 and $W_i'^0$, or by complementing other bits in the message that affect A_{i+1}^0 indirectly.

Such bits are also W_{i-1}^{27} and $W_{i-1}'^{27}$ (which affect A_i^{27}, and then A_{i+1}^0 after the rotate operation), or W_{i-2}^{22} and $W_{i-2}'^{22}$, or $W_{i-l}^{(32-5l) \bmod 32}$ and $W_{i-l}'^{(32-5l) \bmod 32}$ for other small l's. Each such complementation has probability 1/2 to cancel the difference in the carry.

This algorithm can be simplified as follows: The algorithm takes as an input a message and modifies a few subsequent bits in several subsequent words, with the shift of five bits as mentioned above. For example, the modified bits cover all $2^{24}-1$ (non-empty) subsets of $\{W_0^0, \ldots, W_0^3\} \cup \{W_1^5, \ldots, W_1^8\} \cup \ldots \cup \{W_5^{25}, \ldots, W_5^{28}\}$. Then, the pattern of modification is shifted by all 31 possible rotations. Finally, we proceed and make the same analysis starting from W_1,

Round	Function	$-\log P_i$
0, ..., 19	IF	25
20, ..., 39	XOR	16
40, ..., 59	MAJ	15
60, ..., 79	XOR	15

Table 7. Probability Summary

then W_2, etc. The modification process ends when the algorithm starts with W_{10}. This simplification lacks consideration of some optimizations and details given earlier, whose incorporation is vital for an optimized implementation.

4.3 Increasing the Number of Conforming Rounds

In order to start the search at a higher round we need to construct a pair that conforms to $\delta_{r'}$, where $r' > r$. This pair is constructed using the last pair with the maximal number of neutral bit we have. The pair undergoes small modifications of the form described above. Once a message conforms to $\delta_{r'}$ is found, we use the algorithms described in Subsections 4.1 and 4.2 to find a 2-neutral set, and then to find a pair with the largest 2-neutral set.

4.4 Final Search

After computing the 2-neutral set, we start the final search by complementing sequentially every subset of the bits in the 2-neutral set (a total of $2^{k(r)}-1$ trials). Since a large fraction of the resulting pairs of messages conform to δ_r, then the search effectively starts at round r. If in addition $2^{k(r)} > 1/p(r)$, then we expect to find a collision or a near-collision, depending on the expected difference after r rounds. If $2^{k(r)} > 1/p(r \to r')$ for some r', then we expect to find a collision (or a near-collision) of r' rounds reduced (or extended) SHA-0.

5 Results

In our search we used Δ that is optimized for finding 82-round collisions (thus also near-collisions of 80 rounds). This Δ is not suitable for finding full collisions of 80 rounds, as it has two disturbances at the last five rounds. However, its corresponding 80-round probability is much higher than the probability of a Δ that allows a full collision. Although this Δ cannot provide full collisions, it can lead to collisions of 65-round reduced SHA-0 and of 82-round extended SHA-0. The overall probability of successful corrections in 82-round SHA-0 is $p(0 \to 82) = 2^{-71}$. A probability summary for each set of 20 consecutive rounds (i.e., the IF, XOR, MAJ, XOR rounds) is described in Table 7 (in rounds 80 and 81 the probability is 1 if $f_{80} = f_{81} =$ XOR). Using our technique with $r = 22$ the overall probability is reduced to 2^{-43}. Our algorithm finds a 2-neutral set with

	Difference (in hex)	Weight
M_1 and M_1':		
Before:	00401FA0 00060184 00000400 80000020 80000000	17
After:	01C061A0 00020084 00000C00 800001E0 80000000	19
M_2 and M_2':		
Before:	00C030A4 000E0304 00000403 80000060 80000000	20
After:	004070A4 00020104 00000C07 80000020 80000000	18

Table 8. Difference of the Hash Results Before and After the Feed-forward (i.e., $A_{80} \oplus A'_{80}, \ldots, E_{80} \oplus E'_{80}$ and $(A_0 + A_{80}) \oplus (A'_0 + A'_{80}), \ldots, (E_0 + E_{80}) \oplus (E'_0 + E'_{80})$), and Their Hamming Weights

40 neutral and simultaneous-neutral bits (see Table 6), thus we expect to find near-collisions of the compression function after 73 rounds in two computation days on a PC. Our actual findings (using an earlier set of neutral bits) are near-collisions of the compression function with a difference of only three bits (of $A \oplus A', \ldots, E \oplus E'$) after 76 rounds (that still conform to δ_{76}), which are also near-collisions of the full compression function (but do not conform to δ_{80}), and full collisions of 65-round reduced SHA-0. The near collisions were found after about a day of computation for each pair, which is equivalent to a search with a complexity of 2^{35}. Finding 65-round near-collisions take about half an hour. Two such pairs of messages (in 32-bit hex words) are:

1. M_1 = 310EEB32 AC418FC2 415D5A54 6FFA5AA9
   5EE5A5F5 7621F42D 8AE2F4CA F7ACF74B
   B144B4E1 5164DF45 C61AD50C D5833699
   6F0BB389 B6468AC5 4D4323F9 86088694

   M_1' = 310EEB32 AC418FC2 415D5A54 6FFA5AAB
   5EE5A5B5 7621F42F 0AE2F4CA 77ACF74B
   3144B4E3 5164DF05 C61AD50C 558336D9
   EF0BB38B B6468AC7 CD4323B9 06088696
2. M_2 = EF567055 F0722904 009D8999 5AFB3337
   37D5D6A8 9E843D80 69229FB9 06D589AA
   4AD89B67 CFCCCD2C A9BAE20D 6F18C150
   43F89DA4 2E54FE2E AE7B7A15 80A09D3D

   M_2' = EF567055 F0722904 009D8999 5AFB3335
   37D5D6E8 9E843D82 E9229FB9 86D589AA
   CAD89B65 CFCCCD6C A9BAE20D EF18C110
   C3F89DA6 2E54FE2C 2E7B7A55 00A09D3F

The differences of the results of hashing M_1 and M_2 with the full SHA-0 are described in Table 8 along with the number of differing bits. Tables 9 and 10 show detailed information of the evolution of differences in each round of the compression function, including the expanded messages, their differences, the differences $A_{i+1} \oplus A'_{i+1}$, the probability of conformance of each round (in log form), and the rounds where the values collide, or the number of differing bits of the five registers. Both messages collide after 65 rounds, and have only small differences afterwards. If we consider SHA-0 reduced to 76 rounds, our results show a near collision with difference of only three bits before the feed forward and three and four bits difference after the feed forward when using M_1 and M_2.

6 SHA-0 variants

In this section we analyze some variants of SHA-0 that show strengths and weaknesses of the hash function.

6.1 Increasing the Number of Rounds

There are Δ's that lead to collision after 82 rounds, whose probability $p(0 \rightarrow 82)$ is considerably larger than the probability $p(0 \rightarrow 80)$ of the best Δ that leads to an 80-round collision. Therefore, increasing the number of rounds of SHA-0 from 80 to 82 would make it much easier to find collisions.

6.2 Different Order of Functions

Modifying the order of the f_i functions can reduce the complexity of the attack. For example, if the order would be IF, XOR, MAJ, XOR, ..., IF, XOR, MAJ, XOR, where in each round the function changes, the restrictions caused by two consecutive IF round would be removed, and thus Δ's with much higher probabilities could be chosen.

6.3 SHA-1

Since in SHA-1 Equation (1) is replaced by

$$W_i = \mathrm{ROL1}(W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}), \qquad i = 16, \ \ldots, \ 79. \tag{3}$$

which makes the mixing of the message bits much more effective, and since the techniques used in this paper uses the properties inherited from equation (1), the presented attacks are not applicable to SHA-1.

7 Summary

In this paper we described how to find near-collisions of SHA-0 using the surprising existence of many neutral bits. The near-collisions were found within a day on our PC. Our technique also improves the complexity of finding full collisions of SHA-0, but we concentrated on near-collisions due to the very low complexity of finding them. The observation that the strength of SHA-0 is not monotonous with the number of rounds is used here to find near-collisions of 80 rounds by applying the much more efficient attack on SHA-0 extended to 82 rounds. We expect that finding full collisions will take a month of computation time, and intend to check it in the continuation of our research. Due to the additional rotate operation, the results of this paper are not applicable to SHA-1.

References

[1] Ross Anderson, Eli Biham, *Tiger: a Fast New Hash Function*, proceedings of Fast Software Encryption, LNCS 1039, pp. 89–97, Springer Verlag, 1996.

[2] Eli Biham, Adi Shamir, *Differential Cryptanalysis of Snefru, Khafre, REDOC-II, LOKI and Lucifer*, Advances in Cryptology, proceedings of CRYPTO '91, LNCS 576, pp. 156–171, 1992.

[3] Florent Chabaud, Antoine Joux, *Differential Collisions in SHA-0*, Advanced in Cryptology, proceedings of CRYPTO '98, LNCS 1462, pp. 56–71, Springer Verlag, 1999.

[4] Hans Dobbertin, *Cryptanalysis of MD4*, Journal of Cryptology, Vol. 11 pp. 253–271, 1998.

[5] Rosario Genaro, Shai Halevi, Tal Rabin, *Secure Hash-and-Sign Signatures Without the Random Oracle*, Advanced in Cryptology, proceedings of EUROCRYPT'99, LNCS 1592, pp. 123–139, 1999.

[6] Antoine Joux, private communications, 2004.

[7] Alfred Menezes, Paul van Oorschot, Scott Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

[8] National Institute of Standards and Technologies, *Secure Hash Standard*, Federal Information Processing Standards Publication, FIPS-180, May 1993.

[9] National Institute of Standards and Technologies, *Secure Hash Standard*, Federal Information Processing Standards, Publication FIPS-180-1, April 1995.

[10] National Institute of Standards and Technologies, *FIPS 180-2 Secure Hash Standard, Change Notice 1*, Federal Information Processing Standards Publication, FIPS-180-2, December, 2003.

[11] National Institute of Standards and Technologies, *Secure Hash Standard*, Federal Information Processing Standards Publication, FIPS-180-2, August 2002.

[12] Ron Rivest, *The MD4 Message-Digest Algorithm*, Network Working Group Request for Comments:1186, October 1990.

[13] Ron Rivest, *The MD5 Message-Digest Algorithm*, Network Working Group Request for Comments:1321, April 1992.

[14] Ralph Merkle, *A Fast Software One-Way Hash Function*, Journal of Cryptology, Vol. 3, No. 1, pp. 43–58, 1990.

[15] Serge Vaudenay, *On the Need for Multipermutation: Cryptanalysis of MD4 and SAFER*, proceedings of Fast Software Encryption, Second International Workshop, LNCS 1008, pp. 286–297, Springer-Verlag, 1995.

Round (i)	W_i	W'_i	$\exp(\Delta)$	$A_{i+1} \oplus A'_{i+1}$	$-\log P_i$	Diff Bits
0	$310EEB32_x$	$310EEB32_x$	00000000_x	00000000_x	0	collision
1	$AC418FC2_x$	$AC418FC2_x$	00000000_x	00000000_x	0	collision
2	$415D5A54_x$	$415D5A54_x$	00000000_x	00000000_x	0	collision
3	$6FFA5AA9_x$	$6FFA5AAB_x$	00000002_x	00000002_x	1	1
4	$5EE5A5F5_x$	$5EE5A5B5_x$	00000040_x	00000000_x	0	1
5	$7621F42D_x$	$7621F42F_x$	00000002_x	00000000_x	2	1
6	$8AE2F4CA_x$	$0AE2F4CA_x$	80000000_x	00000000_x	1	1
7	$F7ACF74B_x$	$77ACF74B_x$	80000000_x	00000000_x	1	1
8	$B144B4E1_x$	$3144B4E3_x$	80000002_x	00000002_x	1	1
9	$5164DF45_x$	$5164DF05_x$	00000040_x	00000000_x	0	1
10	$C61AD50C_x$	$C61AD50C_x$	00000000_x	00000002_x	3	2
11	$D5833699_x$	$558336D9_x$	80000040_x	00000000_x	1	2
12	$6F0BB389_x$	$EF0BB38B_x$	80000002_x	00000000_x	3	2
13	$B6468AC5_x$	$B6468AC7_x$	00000002_x	00000002_x	2	2
14	$4D4323F9_x$	$CD4323B9_x$	80000040_x	00000000_x	1	2
15	86088694_x	06088696_x	80000002_x	00000000_x	2	1
16	$77518F42_x$	$F7518F42_x$	80000000_x	00000000_x	1	1
17	$DF9C29D7_x$	$5F9C29D5_x$	80000002_x	00000002_x	2	2
18	$5FAAAC39_x$	$DFAAAC7B_x$	80000042_x	00000002_x	1	2
19	$BB09175F_x$	$BB09171F_x$	00000040_x	00000002_x	3	3
20	$6490CB61_x$	$E490CB21_x$	80000040_x	00000002_x	2	4
21	$6861259A_x$	$686125D8_x$	00000042_x	00000000_x	1	4
22	$CDEC748D_x$	$4DEC748F_x$	80000002_x	00000000_x	1	3
23	$445065FB_x$	$C45065F9_x$	80000002_x	00000002_x	1	3
24	$686ECB35_x$	$686ECB75_x$	00000040_x	00000000_x	0	2
25	$9697B486_x$	$1697B486_x$	80000000_x	00000002_x	2	2
26	$B2EBAF47_x$	$32EBAF05_x$	80000042_x	00000002_x	1	3
27	$B0A26036_x$	$30A26074_x$	80000042_x	00000000_x	1	3
28	$D04FEF97_x$	$D04FEF95_x$	00000002_x	00000000_x	1	2
29	$EAC4868C_x$	$EAC4868C_x$	00000000_x	00000000_x	0	2
30	$475CB800_x$	$475CB800_x$	00000000_x	00000000_x	0	1
31	$CD8B252F_x$	$4D8B252F_x$	80000000_x	00000000_x	0	collision
32	$AA516EC2_x$	$AA516EC0_x$	00000002_x	00000002_x	1	1
33	$B55E320E_x$	$B55E324C_x$	00000042_x	00000002_x	1	2
34	$445AED30_x$	$445AED70_x$	00000040_x	00000002_x	2	3
35	$C99B3C31_x$	$499B3C73_x$	80000042_x	00000000_x	1	3
36	$CC6D6275_x$	$CC6D6277_x$	00000002_x	00000000_x	1	3
37	$82AF2BDD_x$	$02AF2BDD_x$	80000000_x	00000000_x	0	2
38	$2B453B89_x$	$2B453B89_x$	00000000_x	00000000_x	0	1
39	$D3219627_x$	53219627_x	80000000_x	00000000_x	0	collision
40	$F27B216D_x$	$F27B216D_x$	00000000_x	00000000_x	0	collision
41	$B82EDD37_x$	$B82EDD37_x$	00000000_x	00000000_x	0	collision
42	$F5DF3BC7_x$	$F5DF3BC7_x$	00000000_x	00000000_x	0	collision
43	$6186FBE6_x$	$6186FBE6_x$	00000000_x	00000000_x	0	collision
44	$E350E8D5_x$	$E350E8D5_x$	00000000_x	00000000_x	0	collision
45	$503FB3B9_x$	$503FB3B9_x$	00000000_x	00000000_x	0	collision
46	$A7CE16AD_x$	$A7CE16AF_x$	00000002_x	00000002_x	1	1
47	$48A469D3_x$	$48A46991_x$	00000042_x	00000002_x	1	2
48	$4C4F1126_x$	$4C4F1164_x$	00000042_x	00000000_x	1	2
49	$6325C5A5_x$	$E325C5A7_x$	80000002_x	00000000_x	2	2
50	$354CDD51_x$	$354CDD51_x$	00000000_x	00000000_x	0	2
51	$66FDFD2C_x$	$66FDFD2C_x$	00000000_x	00000000_x	1	1
52	$675D748C_x$	$E75D748C_x$	80000000_x	00000000_x	0	collision
53	$34FDD312_x$	$34FDD312_x$	00000000_x	00000000_x	0	collision
54	$180DF165_x$	$180DF167_x$	00000002_x	00000002_x	1	1
55	$44F6564F_x$	$44F6560D_x$	00000042_x	00000002_x	1	2
56	$7F16D89E_x$	$7F16D8DC_x$	00000042_x	00000000_x	1	2
57	$A2801211_x$	22801211_x	80000000_x	00000002_x	3	3
58	$6735580C_x$	$6735584E_x$	00000042_x	00000002_x	1	4
59	$28526DED_x$	$28526DAF_x$	00000042_x	00000000_x	2	3
60	$814398E5_x$	$814398E7_x$	00000002_x	00000000_x	1	2
61	$4B535174_x$	$4B535174_x$	00000000_x	00000000_x	0	2
62	$DBDE9B03_x$	$DBDE9B03_x$	00000000_x	00000000_x	0	1
63	$EE3462DC_x$	$6E3462DC_x$	80000000_x	00000000_x	0	collision
64	$4D46459D_x$	$4D46459D_x$	00000000_x	00000000_x	0	collision
65	$7C86B19B_x$	$7C86B199_x$	00000002_x	00000002_x	1	1
66	$DB10930D_x$	$DB10934F_x$	00000042_x	00000002_x	1	2
67	$3714064E_x$	$3714060C_x$	00000042_x	00000000_x	1	2
68	$8295AC97_x$	$0295AC95_x$	80000002_x	00000000_x	1	2
69	$E0484724_x$	$E0484724_x$	00000000_x	00000000_x	0	2
70	$8BD1B4B6_x$	$8BD1B4B4_x$	00000002_x	00000002_x	1	2
71	$8AD78A15_x$	$0AD78A55_x$	80000040_x	00000000_x	0	1
72	$B52D822B_x$	$B52D822B_x$	00000000_x	00000002_x	2	2
73	$7D857AD1_x$	$FD857A93_x$	80000042_x	00000002_x	1	3
74	$B7B1D9F1_x$	$37B1D9B3_x$	80000042_x	00000000_x	1	3
75	$E138B8FC_x$	$E138B8FC_x$	00000000_x	00000002_x	2	3
76	$A58DD5A0_x$	$A58DD5E2_x$	00000042_x	00000082_x	1	5
77	$F29EAD7D_x$	$F29EAD3F_x$	00000042_x	00001000_x	1	5
78	$FC71D2D4_x$	$FC71D2D6_x$	00000002_x	00060184_x	1	9
79	$BDE88CF2_x$	$BDE88CF2_x$	00000000_x	$00401FA0_x$	0	17

Table 9. A Near-Collision and its Differences in the Various Rounds (M_1 and M'_1 are formed by the first 16 words of W and W')

Round (i)	W_i	W'_i	$\exp(\Delta)$	$A_{i+1} \oplus A'_{i+1}$	$-\log P_i$	Diff Bits
0	$EF567055_x$	$EF567055_x$	00000000_x	00000000_x	0	collision
1	$F0722904_x$	$F0722904_x$	00000000_x	00000000_x	0	collision
2	$009D8999_x$	$009D8999_x$	00000000_x	00000000_x	0	collision
3	$5AFB3337_x$	$5AFB3335_x$	00000002_x	00000002_x	1	1
4	$37D5D6A8_x$	$37D5D6E8_x$	00000040_x	00000000_x	0	1
5	$9E843D80_x$	$9E843D82_x$	00000002_x	00000000_x	2	1
6	$69229FB9_x$	$E9229FB9_x$	80000000_x	00000000_x	1	1
7	$06D589AA_x$	$86D589AA_x$	80000000_x	00000000_x	1	1
8	$4AD89B67_x$	$CAD89B65_x$	80000002_x	00000002_x	1	1
9	$CFCCCD2C_x$	$CFCCCD6C_x$	00000040_x	00000000_x	0	1
10	$A9BAE20D_x$	$A9BAE20D_x$	00000000_x	00000002_x	3	2
11	$6F18C150_x$	$EF18C110_x$	80000040_x	00000000_x	1	2
12	$43F89DA4_x$	$C3F89DA6_x$	80000002_x	00000000_x	3	2
13	$2E54FE2E_x$	$2E54FE2C_x$	00000002_x	00000002_x	2	2
14	$AE7B7A15_x$	$2E7B7A55_x$	80000040_x	00000000_x	1	2
15	$80A09D3D_x$	$00A09D3F_x$	80000002_x	00000000_x	2	1
16	$8B479C85_x$	$0B479C85_x$	80000000_x	00000000_x	1	1
17	$CB3EAD0A_x$	$4B3EAD08_x$	80000002_x	00000002_x	2	2
18	$1E522001_x$	$9E522043_x$	80000042_x	00000002_x	1	2
19	20205362_x	20205322_x	00000040_x	00000002_x	3	3
20	$D63179BF_x$	$563179FF_x$	80000040_x	00000002_x	2	4
21	$A8576A05_x$	$A8576A47_x$	00000042_x	00000000_x	1	4
22	$ADA12DA9_x$	$2DA12DAB_x$	80000002_x	00000000_x	1	3
23	$9F88A004_x$	$1F88A006_x$	80000002_x	00000002_x	1	3
24	$C0728FEA_x$	$C0728FAA_x$	00000040_x	00000000_x	0	2
25	$C64B8CDF_x$	$464B8CDF_x$	80000000_x	00000002_x	2	2
26	$6B98FFAC_x$	$EB98FFEE_x$	80000042_x	00000002_x	1	3
27	$A11EE3F6_x$	$211EE3B4_x$	80000042_x	00000000_x	1	3
28	$FDF912D1_x$	$FDF912D3_x$	00000002_x	00000000_x	1	2
29	$6D3BF6BA_x$	$6D3BF6BA_x$	00000000_x	00000000_x	0	2
30	$298328CF_x$	$298328CF_x$	00000000_x	00000000_x	0	1
31	$29EF82E2_x$	$A9EF82E2_x$	80000000_x	00000000_x	0	collision
32	$385CC5D4_x$	$385CC5D6_x$	00000002_x	00000002_x	1	1
33	$04D65A78_x$	$04D65A3A_x$	00000042_x	00000002_x	1	2
34	$8A1424F0_x$	$8A1424B0_x$	00000040_x	00000002_x	2	3
35	$11351F45_x$	$91351F07_x$	80000042_x	00000000_x	1	3
36	$82BF1CBF_x$	$82BF1CBD_x$	00000002_x	00000000_x	1	3
37	$D0F0184B_x$	$50F0184B_x$	80000000_x	00000000_x	0	2
38	$556595C9_x$	$556595C9_x$	00000000_x	00000000_x	0	1
39	$F293B286_x$	$7293B286_x$	80000000_x	00000000_x	0	collision
40	$4346ADD9_x$	$4346ADD9_x$	00000000_x	00000000_x	0	collision
41	$36E6A098_x$	$36E6A098_x$	00000000_x	00000000_x	0	collision
42	$EEE67B0B_x$	$EEE67B0B_x$	00000000_x	00000000_x	0	collision
43	$9E56A7D0_x$	$9E56A7D0_x$	00000000_x	00000000_x	0	collision
44	60238639_x	60238639_x	00000000_x	00000000_x	0	collision
45	$7AC21718_x$	$7AC21718_x$	00000000_x	00000000_x	0	collision
46	$DAECDF02_x$	$DAECDF00_x$	00000002_x	00000002_x	1	1
47	$BF89EC25_x$	$BF89EC67_x$	00000042_x	00000002_x	1	2
48	$8BCC5BE5_x$	$8BCC5BA7_x$	00000042_x	00000000_x	1	2
49	$F9E93AA7_x$	$79E93AA5_x$	80000002_x	00000000_x	2	2
50	$59C4AF61_x$	$59C4AF61_x$	00000000_x	00000000_x	0	2
51	$D45FFB3B_x$	$D45FFB3B_x$	00000000_x	00000000_x	1	1
52	$4E1035E8_x$	$CE1035E8_x$	80000000_x	00000000_x	0	collision
53	$016512B4_x$	$016512B4_x$	00000000_x	00000000_x	0	collision
54	$18901C29_x$	$18901C2B_x$	00000002_x	00000002_x	1	1
55	$35ECCBD3_x$	$35ECCB91_x$	00000042_x	00000002_x	1	2
56	$27099F83_x$	$27099FC1_x$	00000042_x	00000000_x	1	2
57	$49C921C6_x$	$C9C921C6_x$	80000000_x	00000002_x	3	3
58	$E2ED9980_x$	$E2ED99C2_x$	00000042_x	00000002_x	1	4
59	$17C2D470_x$	$17C2D432_x$	00000042_x	00000000_x	2	3
60	$BD164D15_x$	$BD164D17_x$	00000002_x	00000000_x	1	2
61	$26C37009_x$	$26C37009_x$	00000000_x	00000000_x	0	2
62	$5E724CBE_x$	$5E724CBE_x$	00000000_x	00000000_x	0	1
63	$CE9A5044_x$	$4E9A5044_x$	80000000_x	00000000_x	0	collision
64	$D3C21B0E_x$	$D3C21B0E_x$	00000000_x	00000000_x	0	collision
65	$3A0DACE4_x$	$3A0DACE6_x$	00000002_x	00000002_x	1	1
66	$3BA3534D_x$	$3BA3530F_x$	00000042_x	00000002_x	1	2
67	$113A26F1_x$	$113A26B3_x$	00000042_x	00000000_x	1	2
68	$D19BC830_x$	$519BC832_x$	80000002_x	00000000_x	1	2
69	$29E9FA23_x$	$29E9FA23_x$	00000000_x	00000000_x	0	2
70	$70D1E9E5_x$	$70D1E9E7_x$	00000002_x	00000002_x	1	2
71	63247261_x	$E3247221_x$	80000040_x	00000000_x	0	1
72	$3FCFE72E_x$	$3FCFE72E_x$	00000000_x	00000002_x	2	2
73	$14D7B0B7_x$	$94D7B0F5_x$	80000042_x	00000002_x	1	3
74	$077CF5B9_x$	$877CF5FB_x$	80000042_x	00000000_x	1	3
75	$1FF465A6_x$	$1FF465A6_x$	00000000_x	00000002_x	2	3
76	$2628792C_x$	$2628796E_x$	00000042_x	00000182_x	1	6
77	$C6CC2FD7_x$	$C6CC2F95_x$	00000042_x	$0000100C_x$	1	8
78	$E295DBF3_x$	$E295DBF1_x$	00000002_x	$000E0304_x$	1	13
79	$B19BF7ED_x$	$B19BF7ED_x$	00000000_x	$00C030A4_x$	0	20

Table 10. A Near-Collision and its Differences in the Various Rounds (M_2 and M'_2 are formed by the first 16 words of W and W')