Differential Collisions in SHA-0

Florent Chabaud and Antoine Joux

Centre d'Électronique de l'Armement
CASSI/SCY/EC
F-35998 Rennes Armées, France
{chabaud,joux}@celar.fr

Abstract. In this paper we present a method for finding collisions in SHA-0 which is related to differential cryptanalysis of block ciphers. Using this method, we obtain a theoretical attack on the compression function SHA-0 with complexity 2^{61}, which is thus better than the birthday paradox attack. In the case of SHA-1, this method is unable to find collisions faster than the birthday paradox. This is a strong evidence that the transition to version 1 indeed raised the level of security of SHA.

1 Description of SHA

1.1 Historical Overview

The Secure Hash Standard (SHS) [7] was issued by the National Institute of Standards and Technology in 1993. It was largely inspired from Rivest's MD4 [5]. However, a certain number of basic blocks of this function were different from MD4 ones, but no explanation was given for the choices. Two years later, an addendum was made to the standard, slightly altering the function [8]. This change was claimed to correct a technical weakness in SHA but no justification was given. Yet, it was reported that a collision attack better than the birthday paradox had been found by the NSA.

Independantly, several attacks on the original MD4 function, and its MD5 improvement [6] have been published [2, 4]. However, these attacks couldn't be applied to the Secure Hash Algorithm (neither in the first nor in the second version) because of the expansion used.

1.2 Notation

The symbols we use in this paper are defined Table 1. Besides, we denote by capital letters 32-bits words, and $X^{(i)}$ stand for the value of X used at i-th round of SHA.

1.3 Description of SHA

Description of the Hash Function. The hash functions in the SHA family deal with 512 bits message blocks and output a 160 hash value. This hash value is formed by concatenating 5 registers of 32 bits each. In order to hash a message, several steps are performed:

Table 1. Notations

Notation	Definition
$\mathbb{F}_q$	Finite field with q elements.
$\langle X, Y, \ldots, Z \rangle$	Concatenation of 32-bits words.
$+$	Addition on 32-bits words modulo 2^{32}.
$\oplus$	*Exclusive or* on bits or 32-bits words.
$\vee$	*Inclusive or* on bits or 32-bits words.
$\wedge$	Logical *and* on bits or 32-bits words.
$ROL_\ell(X)$	Rotation by ℓ bits of a 32-bits word.
X_i	The ith bit of 32-bits word X, from the least significant 0 to the most significant 31.

1. Pad the message to be hashed by adding a 1, aa appropriate number of 0 and the 64 bits integer representing the length of the message. After this padding operation, the message is formed of a integral number of 512 blocks.
2. Initialize 5 registers of 32 bits A, B, C, D and E with fixed constants:
   - $A =$ 0x67452301
   - $B =$ 0xEFCDAB89
   - $C =$ 0x98BADCFE
   - $D =$ 0x10325476
   - $E =$ 0xC3D2E1F0
3. For each message block, copy A, B, C, D and E respectively in AA, BB, CC, DD and EE. Apply the compression function to AA, BB, CC, DD, EE and the message block. This yields AA', BB', CC', DD' and EE'. These 5 values are then added respectively to A, B, C, D and E.
4. Output the concatenation of A, B, C, D and E.

In the remaining of this paper, we try to find collisions on the compression function, from which collision on the hash function are trivial.

Description of the Compression Function. Following [7], we denote by $\langle W^{(0)}, \ldots, W^{(15)} \rangle$ the 512 bits input of SHA, constituted by 16 words of 32 bits. The first step of SHA-0 is to perform an expansion on these 512 bits. The result of this expansion is given by the following relation:

$$W^{(i)} = W^{(i-3)} \oplus W^{(i-8)} \oplus W^{(i-14)} \oplus W^{(i-16)}, \ \forall i, \ 16 \leq i < 80 \ . \qquad (1)$$

These 80 words of 32 bits are used to alter the five 32-bits words state denoted by $A^{(i)}$, $B^{(i)}$, $C^{(i)}$, $D^{(i)}$, $E^{(i)}$. The initial state is the input of the compression function. We now denote it $\langle A^{(0)}, B^{(0)}, C^{(0)}, D^{(0)}, E^{(0)} \rangle$.

The modification of $\langle A^{(i)}, B^{(i)}, C^{(i)}, D^{(i)}, E^{(i)} \rangle$ state is performed by the following transformation, where the function $f^{(i)}$ and the constant $K^{(i)}$ are set according to Table 2, and $ADD(U, V, W, X, Y) = U+V+W+X+Y \pmod{2^{32}}$:

for $i = 0$ to 79

$$A^{(i+1)} = ADD\left(W^{(i)}, ROL_5\left(A^{(i)}\right), f^{(i)}\left(B^{(i)}, C^{(i)}, D^{(i)}\right), E^{(i)}, K^{(i)}\right)$$
$$B^{(i+1)} = A^{(i)}$$
$$C^{(i+1)} = ROL_{30}\left(B^{(i)}\right)$$
$$D^{(i+1)} = C^{(i)}$$
$$E^{(i+1)} = D^{(i)}$$

Table 2. SHA definition of function $f^{(i)}(X, Y, Z)$, and constant $K^{(i)}$.

Round i	Function $f^{(i)}$		Constant $K^{(i)}$
	Name	Definition	
0–19	IF	$(X \wedge Y) \vee (X \wedge Z)$	0x5A827999
20–39	XOR	$(X \oplus Y \oplus Z)$	0x6ED9EBA1
40–59	MAJ	$(X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$	0x8F1BBCDC
60–79	XOR	$(X \oplus Y \oplus Z)$	0xCA62C1D6

The output of the compression function is the 160 bits word obtained in the final state $\langle A^{(80)}, B^{(80)}, C^{(80)}, D^{(80)}, E^{(80)} \rangle$. By collision, we understand the standard meaning of finding two input words $\langle W^{(0)} \ldots W^{(15)} \rangle$ and $\langle W'^{(0)} \ldots W'^{(15)} \rangle$ that gives the same 160-bits output $\langle A^{(80)}, B^{(80)}, C^{(80)}, D^{(80)}, E^{(80)} \rangle$, using the same initial value $\langle A^{(0)}, B^{(0)}, C^{(0)}, D^{(0)}, E^{(0)} \rangle$.

The basic architecture of SHA can be illustrated by Fig. 1. The expansion box can be considered as a linear application from $(\mathbb{F}_2)^{512}$ to $(\mathbb{F}_2)^{2560}$, that maps $\langle W^{(0)} \ldots W^{(15)} \rangle$ to $\langle W^{(0)} \ldots W^{(79)} \rangle$. This linear mapping is the only difference between first and second version of SHA. More precisely, the extension of SHA-1 is obtained by replacing (1) by the following equation, which differs from (1) by the one bit rotation to the left:

$$W^{(i)} = ROL_1\left(W^{(i-3)} \oplus W^{(i-8)} \oplus W^{(i-14)} \oplus W^{(i-16)}\right), \forall i,\ 16 \leq i < 80\ . \quad (2)$$

We will denote E_0 the initial expansion described by (1), and E_1 the modified expansion described by (2). This generic architecture defines a family of hash functions that could be derived by changing the expansion box.

2 Propagation of Local Perturbations in SHA-Like Hash Functions

2.1 Weakened SHA Variations

The Bare Architecture of SHA. We first want to study the propagation of local perturbations in a fully linear variation of SHA, in order to discriminate between

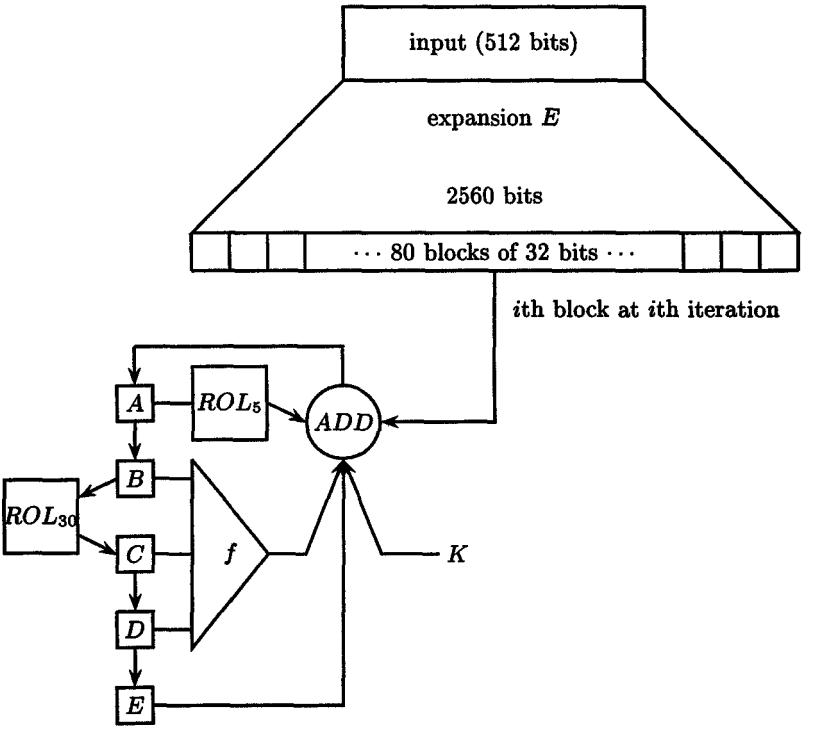


Fig. 1. SHA architecture

the roles of the bare architecture of the hash functions on one side and of the elementary building blocks on the other side. Within the compression function of a hash function in the SHA family, there are two sources of non-linearity, the $f^{(i)}$ functions and the addition function ADD.

Thus, the first hash function we consider is SHI1[1] the compression function in the SHA family built by starting from SHA-0 (thus using expansion E_0) and by replacing the ADD function by an *exclusive-or* on 5 variables, and all the $f^{(i)}$ by XOR functions.

We denote as usual by $W^{(i)}$ the ith word of the expansion ($0 \leq i < 80$), and the 32 bits of this word are numbered $W_0^{(i)}, \ldots, W_{31}^{(i)}$.

We now relax the constraints on the W vector and temporary forget that it results from an expansion process. Thus, we can apply any local perturbation on any bit of W. For example, we can negate the value of $W_1^{(i)}$. This change will modify bit 1 of $A^{(i+1)}$, bit 1 of $B^{(i+2)}$, bit 31 of $C^{(i+3)}$, bit 31 of $D^{(i+4)}$ and finally bit 31 of $E^{(i+5)}$. If we want to prevent further changes, we need to negate the values of bits $W_6^{(i+1)}$, $W_1^{(i+2)}$, $W_{31}^{(i+3)}$, $W_{31}^{(i+4)}$ and $W_{31}^{(i+5)}$. These new modifications prevent the change on bit 1 of $A^{(i+1)}$ to change bit 6 of $A^{(i+2)}$, the change on bit 1 of $B^{(i+2)}$ to change bit 1 of $A^{(i+3)}$, the change on bit 31 of $C^{(i+3)}$ to change bit 31 of $A^{(i+4)}$, the change on bit 31 of $D^{(i+4)}$ to change bit 31 of $A^{(i+5)}$ and the change on bit 31 of $E^{(i+5)}$ to change bit 31 of $A^{(i+6)}$. Thus negating $W_1^{(i)}$, $W_6^{(i+1)}$, $W_1^{(i+2)}$, $W_{31}^{(i+3)}$, $W_{31}^{(i+4)}$ and $W_{31}^{(i+5)}$ gives two different paths from $A^{(i)}$, $B^{(i)}$, $C^{(i)}$, $D^{(i)}$ and $E^{(i)}$ to $A^{(i+6)}$, $B^{(i+6)}$, $C^{(i+6)}$, $D^{(i+6)}$ and $E^{(i+6)}$, and yields a local collision. This is summarized in Fig. 2.

Note 1. It is clear that what we say for bit 1, can be generalized for any other bit from 0 to 31. However, it will become clear in the following (see Sect. 2.1), that this choice is the best one for our purpose. Hence, we focus on this value through the rest of this paper.

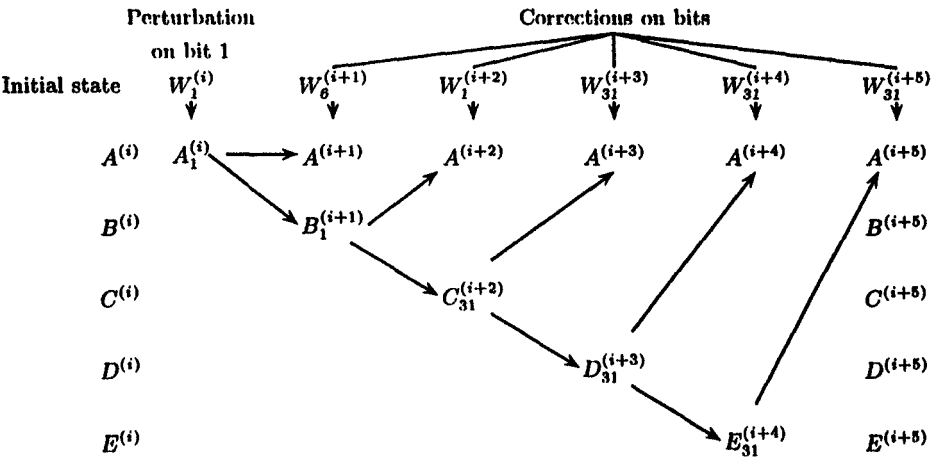
Since everything is linear, we can apply simultaneously as many local collisions as we want and get two different paths from $A^{(0)}$, $B^{(0)}$, $C^{(0)}$, $D^{(0)}$ and $E^{(0)}$ to $A^{(80)}$, $B^{(80)}$, $C^{(80)}$, $D^{(80)}$ and $E^{(80)}$, the first path using the original W and the second one using the modified one which we denote by W'. The question that now arises is "How to choose the local collisions to come back under the condition that both W and W' result from an expansion process ?"

Choosing the local collisions simply means to build an error vector m_0 of 80 bits (numbered from 0 to 79) with a 1 in position i if we want to negate $W_1^{(i)}$. However, we can't choose to negate $W_1^{(i)}$ for $i \geq 75$, since a perturbation in round i is never corrected before round $i + 6$, and since all perturbations must be corrected by round 80.

Let $\left(m_0^{(0)}, \ldots, m_0^{(79)}\right)$ be one of these error vectors. We deduce from it the perturbative mask on W, $M_0 = \left\langle M_0^{(-5)}, \ldots, M_0^{(79)} \right\rangle$ defined by:

$$\forall i,\ -5 \leq i \leq -1,\ M_0^{(i)} = 0$$

[1] SHI1 is a French pun involving cats and dogs.



Subscripts denote the perturbed bit of the state.

Fig. 2. SHI1 propagation of perturbation

$$\forall i,\ 0 \leq i \leq 79,\ M_{0,k}^{(i)} = 0 \text{ if } k \neq 1;$$
$$\forall i,\ 0 \leq i \leq 79,\ M_{0,1}^{(i)} = m_0^{(i)} \ .$$

This mask is completed by 5 zero-blocks, because the corrective masks are now deduced from this perturbative mask by translation and rotation.

The first corrective mask M_1 is deduced from M_0 by a translation by one round, and a rotation of 5 bits to left. This rotation comes from the description of the SHA transformation (see Sect. 1.3 and Fig. 2). Hence, it applies on bits numbered $k = 6$. We have:

$$\forall i,\ -4 \leq i \leq 79,\ M_1^{(i)} = ROL_5\left(M_0^{(i-1)}\right) \ . \tag{3}$$

The second corrective mask M_2 is deduced from M_0 by a translation by two rounds and no rotation (see Fig. 2).

$$\forall i,\ -3 \leq i \leq 79,\ M_2^{(i)} = M_0^{(i-2)} \ . \tag{4}$$

Similarly, M_3 (resp. M_4, M_5) are deduced from M_0 by translation by three (resp. four, five) rounds, and apply on bits numbered $k = 31$.

$$\forall i,\ -2 \leq i \leq 79,\ M_3^{(i)} = ROL_{30}\left(M_0^{(i-3)}\right) \ ; \tag{5}$$

$$\forall i,\ -1 \leq i \leq 79,\ M_4^{(i)} = ROL_{30}\left(M_0^{(i-4)}\right) \ ; \tag{6}$$

$$\forall i,\ 0 \leq i \leq 79,\ M_5^{(i)} = ROL_{30}\left(M_0^{(i-5)}\right) \ ; \tag{7}$$

Now, what we need is that the global differential mask M defined by

$$\forall i,\ 0 \leq i \leq 79,\ M^{(i)} = M_0^{(i)} \oplus M_1^{(i)} \oplus M_2^{(i)} \oplus M_3^{(i)} \oplus M_4^{(i)} \oplus M_5^{(i)}\ , \quad (8)$$

must be an output of E_0.

This condition holds if all masks M_k satisfy (1), which is ensured if the initial perturbative mask satisfies the following equation:

$$M_0^{(i)} = M_0^{(i-3)} \oplus M_0^{(i-8)} \oplus M_0^{(i-14)} \oplus M_0^{(i-16)},\ \forall i,\ 11 \leq i < 80\ . \quad (9)$$

Moreover, since E_0 does not interleave bits (see (1)), we can split the expansion in 32 identical boxes e_0 expanding 16 bits to 80 bits, and defined by (1) considered upon bits. The box e_0 is small enough to be exhaustively enumerated. The number of possible masks is in fact relatively small, as there are only 128 of the $2^{16} = 65536$ possible inputs, that satisfy (9), and the constraint of 5 zeroes on rounds 75 to 79, and thus give a mask m_0.

Given such a mask, one can obtain M, and, by reversing the linear application E_0, one can compute the corresponding 512 bits input mask μ such that $M = E_0(\mu)$. As the expansion boxes of the SHA functions are coded in a systematic way, it is clear that $\mu = \langle M^{(0)}, \ldots, M^{(15)} \rangle$. For all input $W = \langle W^{(0)} \ldots W^{(15)} \rangle$, $W' = W \oplus \mu$ has same output by the linear compression function SHI1.

Introducing Non Linear Functions.

From a Deterministic to a Probabilistic Method. We now want to study the impact of non-linear functions $f^{(i)}$ in the security of hash function from the SHA family. We consider a second function SHI2, the compression function in the SHA family built by starting from SHA-0 (thus using expansion E_0) and by replacing the ADD function by an *exclusive-or* on 5 variables. This can also be seen as SHI1 with added non-linear functions $f^{(i)}$. It can easily be seen that in some cases the $f^{(i)}$ behaves like a XOR. Thus, the previous attack may work. The questions that arise are "When does it work?" and "What is the probability of success?"

In order to compute the probability we need to make a detailed analysis of the IF and MAJ functions. Since these functions work in parallel on 32 bits, we need only study what happens on a single bit. Assuming that we study the behavior of the transition from $f^{(i)}(B^{(i)}, C^{(i)}, D^{(i)})$ to $f^{(i)}(B'^{(i)}, C'^{(i)}, D'^{(i)})$, by looking carefully at the rotations and at our perturbation model one can see that different cases can occur:

1. There is no change at all in the inputs, *i.e.* $B^{(i)} = B'^{(i)}$, $C^{(i)} = C'^{(i)}$ and $D^{(i)} = D'^{(i)}$. In that case the output $f(B'^{(i)}, C'^{(i)}, D'^{(i)}) = f(B^{(i)}, C^{(i)}, D^{(i)})$ does not change and $f^{(i)}$ behaves as XOR.
2. There is a single difference in the entries on bit 1 of $B^{(i)}$, *i.e.* $B'^{(i)} = B^{(i)} \oplus 2^1$. In that case, $f^{(i)}$ behaves as a XOR, if and only if $f^{(i)}(B'^{(i)}, C'^{(i)}, D'^{(i)}) = f^{(i)}(B^{(i)}, C^{(i)}, D^{(i)}) \oplus 2^1$.

3. There is a single difference in the entries on bit 31 of $C^{(i)}$ or $D^{(i)}$ (exclusive or). In that case, $f^{(i)}$ behaves as a XOR, if and only if $f^{(i)}(B', C'^{(i)}, D'^{(i)}) = f^{(i)}(B^{(i)}, C^{(i)}, D^{(i)}) \oplus 2^{31}$.
4. There are two differences in the entries on bits 31 of $C^{(i)}$ and $D^{(i)}$, that is to say $C'^{(i)} = C^{(i)} \oplus 2^{31}$ and $D'^{(i)} = D^{(i)} \oplus 2^{31}$. In that case, $f^{(i)}$ behaves as a XOR, if and only if the output of $f^{(i)}$ does not change $f^{(i)}(B'^{(i)}, C'^{(i)}, D'^{(i)}) = f^{(i)}(B^{(i)}, C^{(i)}, D^{(i)})$.

We can now look at the three last cases for the MAJ and IF function. For the MAJ function, Cases 2 and 3 behave identically, the change in the output occurs if and only if the two bits of input that do not change are opposite. This occur with probability $1/2$. In Case 4, the output does not change if and only if the two bits $C_{31}^{(i)}$ and $D_{31}^{(i)}$ change in opposite directions. This occurs with probability $1/2$.

For the IF function, in Case 2 the output changes if and only if bits $C_{31}^{(i)}$ and $D_{31}^{(i)}$ are opposite. This occurs with probability $1/2$. In Case 3, the output changes if and only if bit $B_{31}^{(i)}$ points on the changing bit (i.e. $B_{31}^{(i)} = 1$ if $C'^{(i)} = C^{(i)} \oplus 2^{31}$ changes and $B_{31}^{(i)} = 0$ if $D'^{(i)} = D^{(i)} \oplus 2^{31}$ changes), this occurs with probability $1/2$. In Case 4, the output will always change, so the probability of good behavior is 0. This implies, that we need to choose a perturbation pattern with no two adjacent perturbations in the IF rounds. More precisely, as the IF rounds occur from round 0 to 19 (see Table 2), and Case 4 involves states $C^{(i)}$ and $D^{(i)}$, no two adjacent perturbations can appear before round 16, but there may be two adjacent perturbations on rounds 16 and 17, because the propagation of the error will occur for $C^{(i)}$ and $D^{(i)}$ on round 20 (see Fig. 2).

Under all our constraints, we were able to find a pattern with a global probability of success of about $1/2^{24}$. We represent hereafter the corresponding 80 bits output of the e_0 box. The 5 preceding zeroes are just there to recall that this pattern satisfies the constraints developed in Sect. 2.1:

```
00000 00100010000000101111
      01100011100000010100
      01000100100100111011
      00110000111110000000
```

This pattern m_0 is ended and preceded by 5 zeroes, and has no two adjacent bits in the 16 first rounds.

By the same construction as described in Sect. 2.1, we obtain a differential mask that can be applied on input word, and gives a collision with non negligible probability. We reference this mask by $\mathcal{M}$.

Evaluating the probability of success is quite tricky, because the 16 first rounds must not be included in this evaluation. The reason for this appears when implementing the collision search.

Implementing the Collision Search. We now have the differential mask $\mathcal{M}$ that we can try to apply on any input word $\langle W^{(0)} \dots W^{(15)} \rangle$. In order to check

whether we have a collision or not, one has to verify for every perturbation, if the correction is done well, that is to say, if the function $f^{(i)}$ behaves like a XOR. Since each perturbation appears in 3 different (successive) $f^{(i)}$, we need to consider many elementary probabilities. In our example, there are perturbations in positions 2, 6, 14, 16, 17, 18, 19, 21, 22, 26, 27, 28, 35, 37, 41, 45, 48, 51, 54, 55, 56, 58, 59, 62, 63, 68, 69, 70, 71 and 72. Table 3 shows which case each perturbation is related to, for the three $f^{(i)}$ involved.

Note 2. In Table 3, Case 4 in MAJ case is counted for a probability $1/\sqrt{2}$ for each of the two perturbations involved. In this way, the global overall probability of $1/2$ seen above is obtained.

Table 3. Probability of success of mask $\mathcal{M}$ in SHI2 model

Perturbation in round i	$f^{(i+2)}$	case	$f^{(i+3)}$	case	$f^{(i+4)}$	case	overall probability	probability logarithm
2	IF	2	IF	3	IF	3	1/8	3
6	IF	2	IF	3	IF	3	1/8	3
14	IF	2	IF	3	IF	3	1/8	3 = 2 + 1 (see Note 3)
16	IF	2	IF	3	XOR	-	1/4	2
17	IF	2	XOR	-	XOR	-	1/2	1
18, 19, 21								
22, 26, 27	XOR	-	XOR	-	XOR	-	1	0
28, 35								
37	XOR	-	MAJ	3	MAJ	3	1/4	2
41	MAJ	2	MAJ	3	MAJ	3	1/8	3
45	MAJ	2	MAJ	3	MAJ	3	1/8	3
48	MAJ	2	MAJ	3	MAJ	3	1/8	3
51	MAJ	2	MAJ	3	MAJ	3	1/8	3
54	MAJ	2	MAJ	3	MAJ	4	$1/4\sqrt{2}$	2.5
55	MAJ	2	MAJ	4	MAJ	4	1/4	2
56	MAJ	2	MAJ	4	XOR	-	$1/2\sqrt{2}$	1.5
58, 59, 62								
63, 68, 69	XOR	-	XOR	-	XOR	-	1	0
70, 71, 72								

As the input word is transmitted with no modification through the expansion, it is possible to split the search in two. First, we search $W^{(0)} \dots W^{(14)}$ such that the function $f^{(i)}$ behaves like a XOR when the mask is applied. This occurs with probability $1/2^6$, as the two perturbations involved are in positions 2 and 6.

Then, $W^{(0)} \dots W^{(14)}$ being fixed, we try many values of $W^{(15)}$ (of course we must try less than 2^{32}, in practice any large number such as 10000 is satisfactory). Such a $W^{(15)}$ can lead to a collision after 80 rounds if all the other rounds behave

nicely. As can be seen on Table 3, this happen with probability $1/2^{26}$. Since the first part of the construction is done once for many $W^{(15)}$, the second probability gives the real cost of the enumeration.

Note 3. This first evaluation gives an overall probability of $1/2^{26}$ in place of the claimed probability. But we can further refine this approach and get rid of some of the probability coming from perturbation of round 14. The first function related to this perturbation is the IF function seen in round 16. This function behaves nicely if bits $C_1^{(16)}$ and $D_1^{(16)}$ differs. These bits are known in round 14, since they are copies of $A_3^{(14)}$ and $A_3^{(13)}$. This allows us to transfer a probability of $1/2$ from the second part of the enumeration to the first one. This reduces the probability to $1/2^{25}$.

The second function related to perturbation of round 14 is the IF function seen in round 17. This function behaves nicely if bit $B_{31}^{(17)}$ is a 1. Since this bit is a copy of $A_{31}^{(16)}$, one can check its correctness just after choosing $W^{(15)}$, and, if necessary, change bit 31 of $W^{(15)}$ before starting the testing process. This reduces the probability to the announced $1/2^{24}$.

Note 4. In the case of SHI2, the collision search is very fast and can be performed in less than a half minute. Here is a sample collision:

```
1a6191b0 3c4a331c 1f228ea2 403b7609
062ec496 48611ca8 583401bc 399879d0
2270fdbd 2a8090f0 4b12fd98 473cc7a1
002831a9 50fe1535 61ac0d3d f26700ec
```

and

```
1a6191b0 3c4a331c 1f228ea0 403b7649
062ec494 c8611ca8 d83401be b9987990
2270fdbf aa8090f0 cb12fd98 c73cc7a1
002831a9 50fe1535 61ac0d3f f26700ac
```

both give

```
1334f224 21a3efc9 b667d2b2 2890013b 56013ca9
```

after the 80 rounds of the SHI2 function.

Introducing Addition. Eventually, before dealing with SHA-0 and SHA-1 we want to study the influence of the addition ADD on our scheme of attack. We consider a third function SHI3, the compression function in the SHA family built by starting from SHA-0 (thus using expansion E_0) and by replacing the non-linear functions IF and MAJ by the function XOR. This can also be seen as SHI1 with the addition ADD put back.

The new point here is that a perturbation may lead to carries. If we can prevent this from happening, everything will behave nicely as before. At first, it seems that each perturbation bit and each correction bit may lead to carry. This

would imply an elementary probability of $1/2^6$ per perturbation, and therefore give no usable attack. However, remember that we choose to apply perturbation on bit 1 of $W^{(i)}$ thus getting three corrections on bits in position 31 ($W_{31}^{(i+3)}$, $W_{31}^{(i+4)}$, $W_{31}^{(i+5)}$). Since there is no possible carry from bit 31, this halves the logarithm of the elementary probability, and this explains our above choice .

We can reduce this even further, suppose that $W_1^{(i)}$ is a 0 and that it changes to a 1 in $W_1'^{(i)}$, if no carry occurs (probability 1/2) then $A_1^{(i+1)}$ is a 0 (and $A_1'^{(i+1)}$ is a 1). Following this change in the computation of $A^{(i+2)}$, we see that $W_6^{(i+1)}$ should be a 1 (and $W_6'^{(i+1)}$ should be a 0), otherwise the correction would lead to a carry. If this condition holds then the correction always occur without carry. The most difficult point is to correct the change in the computation of $A^{(i+3)}$. As before, we choose to fix $W_1^{(i+2)}$ to 1 (and $W_1'^{(i+2)}$ to 0). Then the correction behaves nicely if the first bit of the result of the XOR function is equal to $B_1^{(i+2)}$ (i.e $A_1^{(i+1)}$). This is true whenever $C_1^{(i+2)} = D_1^{(i+2)}$ (with probability 1/2).

The very same arguments show that the probabilities are the same when $W_1^{(i)}$ is a 1 (and changes to a 2 in $W_1'^{(i)}$). In fact, the important issue is that a change from 0 to 1 (an incrementation) must be corrected by a change from 1 to 0 (a decrementation) and that a change from 1 to 0 must be corrected by a change from 0 to 1. The elementary probability to consider is formed from a factor 1/2 to ensure that the initial perturbation engenders no carry, and another 1/2 to ensure that the XOR keeps the change in the same direction.

Two technical complications arise in this case, the first one is that we need to build W in such a way that $W_1^{(i)}$, $W_6^{(i+1)}$ and $W_1^{(i+2)}$ will satisfy the above (non-linear) constraints. Since E_0 does not interleave bits, we build W_1 and W_6 at the very beginning and keep them fixed for the rest of the attack. The second complication comes from the fact that nothing prevents us from getting a change in $W_1^{(i)}$, and another in $W_1^{(i+2)}$, in that case we get different conditions on W_1 and W_6 but the elementary probability of 1/4 still holds.

In practice, we were able to find a pattern with probability of $1/2^{44}$ (computed as in the SHI2 case)[2]. This pattern is:

```
00000 01000010100100011110
      01011000001110000000
      00001100000011011000
      00011000101101100000
```

and we will denote $\mathcal{M}'$ its associated differential mask.

Note 5. In this second pattern, we have no condition on adjacent perturbations, since we consider $f^{(i)}$ to always be the XOR function. Thus, one can note that this pattern has two adjacent perturbative bits on rounds 15 and 16.

[2] One can refine the enumeration process to force the perturbations of round 16 and 17 and their associated corrections to be successful. The details are too tricky to be explained here, but will appear in the journal version of this paper. This leads to a 2^{40} running time, which was confirmed by our implementation.

Associated to this pattern, the conditions on bits 1 and 6 of W and the expansion E_0 made us choose the following values for these bits:

```
Bit1: 01110010000000011000
      10101101011110000110
      11010101111101101010
      00001001111101010111
Bit6: 00010000000110100000
      10110001101001110011
      01101101011111000010
      00001011101101110111
```

Note 6. After a few days of computation, we were able to find an explicit collision for SHI3:

```
53c29e14 44fe051b 4a8ce882 576e1943
0c0abc30 3806260d 76cbeb2f 1b8379a8
0da433ac 6337b011 1041e2a9 20b44364
1a3f8b70 0e7a4620 25e81245 289acb2b
```

and

```
53c29e14 44fe0519 4a8ce8c2 576e1941
8c0abc30 b806260d f6cbeb2d 1b8379e8
0da433ac e337b051 9041e2ab 20b44366
9a3f8b30 8e7a4622 a5e81245 a89acb29
```

both give

```
983d1f8e e619f190 2e94fa09 0b0d479c 4c536e3e
```

after the 80 rounds of the SHI3 function.

2.2 True SHA-0 Case

Having studied SHI1, SHI2, and SHI3, we now come back to the SHA-0 case. In this case, all perturbations have to be inserted without any carry, as in SHI3 case. Moreover, we need to probe deeper into the analysis of the IF and MAJ functions, that we carried out to deal with SHI2.

Let us start with the IF function. As in SHI2, we must consider Cases 2, 3 and 4. Case 4 is always unacceptable in a pattern of attack. In case 3, everything remains the same: the change must go through the IF function, and it happens with probability 1/2. In case 2, the change must go through the function. Moreover, as in SHI3 case, its direction must be preserved. These two conditions are satisfied with probability 1/4.

For the MAJ function, we can remark that MAJ never reverses the direction of a change, so that cases 2 and 3 are left unchanged, and each one leads to an elementary probability of 1/2. However, case 4 undergoes an interesting change.

The new fact, as compared to SHI2, is that as in SHI3, we have the following additional properties:

$$C_{31}^{(i+3)} = A_1^{(i+1)} = W_1^{(i)} \ ,$$
$$D_{31}^{(i+4)} = A_1^{(i+2)} = W_1^{(i+1)} \ .$$

This means that in case 4, MAJ behaves as a XOR as soon as the following equation holds,

$$W_1^{(i)} \neq W_1^{(i+1)} \ , \tag{10}$$

because the result of MAJ does not change if and only if $C_{31}^{(i+3)}$ and $D_{31}^{(i+4)}$ change in opposite directions. Thus, when there are perturbations in round i and $i+1$ with $36 \leq i \leq 55$,, if we add the additional constraints (10) on W_1, then the elementary probability of case 4 for the MAJ function is 1. These conditions are added to the previous ones described for SHI3, when building W_1 and W_6.

Taking in account all these constraints, we were able to find two good patterns, with probability of success $1/2^{68}$ (resp. $1/2^{69}$). These patterns are:

```
00000 00010000000100100000
      00100001101101111110
      11010010000101010010
      10100010111001100000 c=68

00000 00100010000000101111
      01100011100000010100
      01000100100100111011
      00110000111110000000 c=69
```

We can now build the differential masks deduced from each pattern by the construction of Sect. 2.1. The second pattern was denoted $\mathcal{M}$ in Sect. 2.1. We denote the first one by $\mathcal{M}''$.

Note 7. The computation of the probabilities can be done from Tables 5 and 4. As explained in Note 3, the perturbation in round 14 is on the boundary between the two enumerations. It contributes to the overall probability of success by a single $1/2$.

Note 8. Given a pattern $\mathcal{M}''$ (resp. $\mathcal{M}$), once W_1 and W_6 are chosen according to the constraints, the collision search by itself remains unchanged (see Sect. 2.1). The expected running complexity is thus 2^{68} (resp. 2^{69}). However, being more careful when implementing the collision search, we can get rid of the remaining probability implied by the perturbation in round 14. We hence obtain a running complexity of 2^{67} (resp. 2^{68}). Moreover, in case of $\mathcal{M}$, one can also suppress the probabilities implied by the perturbations in round 16 and 17. This further decreases the probability of success of $\mathcal{M}$ to the claimed value of 2^{61}.

This ultimate trick can also be used in SHI2 model. Thus, instead of the probability $1/2^{24}$ obtained in Note 3, we can obtain a probability of $1/2^{20}$.

Table 4. Probability of success of mask $\mathcal{M}$ for SHA-0

Perturbation in round i	$f^{(i+2)}$	case	$f^{(i+3)}$	case	$f^{(i+4)}$	case	overall probability	probability logarithm
2	IF	2	IF	3	IF	3	1/32	5
6	IF	2	IF	3	IF	3	1/32	5
14	IF	2	IF	3	IF	3	1/32	4 + 1
16	IF	2	IF	3	XOR	-	1/16	4
17	IF	2	XOR	-	XOR	-	1/8	3
18, 19, 21								
22, 26, 27	XOR	-	XOR	-	XOR	-	1/4	2
28, 35								
37	XOR	-	MAJ	3	MAJ	3	1/16	4
41	MAJ	2	MAJ	3	MAJ	3	1/16	4
45	MAJ	2	MAJ	3	MAJ	3	1/16	4
48	MAJ	2	MAJ	3	MAJ	3	1/16	4
51	MAJ	2	MAJ	3	MAJ	3	1/16	4
54	MAJ	2	MAJ	3	MAJ	4	1/8	3
55	MAJ	2	MAJ	4	MAJ	4	1/4	2
56	MAJ	2	MAJ	4	XOR	-	1/4	2
58, 59, 62								
63, 68, 69	XOR	-	XOR	-	XOR	-	1/4	2
70, 71, 72								

Note 9. In the middle of the second 20-rounds block of pattern $\mathcal{M}$ with probability $1/2^{69}$ (basic search) or $1/2^{61}$ (improved search), we were lucky to find a group of 5 zeroes (in fact 6 but 5 is sufficient for our purpose). This allows us to stop the attack after this group, with a partial collision on 35 rounds of SHA. Here is such a partial collision:

```
78fb1285 77a2dc84 4035a90b b61f0b39
4a4d1c83 186e8429 74326988 7f220f79
a08e7920 16a3e469 2ed4213d 4a75b904
38bef788 2274a40c 4c14e934 cee12cec
```

and

```
78fb1285 77a2dc84 4035a909 b61f0b79
4a4d1c81 986e8429 f432698a ff220f39
a08e7922 96a3e469 aed4213d ca75b904
38bef788 2274a40c 4c14e936 cee12cac
```

both yield after 35 rounds of SHA-0:

```
7b907fb9 d050108b 88d6e6d6 5c70d4a3 7e06a692
```

The probability to find such a collision is $1/2^{22}$, using the basic collision search, or $1/2^{14}$, using the improved collision search.

Table 5. Probability of success of mask $\mathcal{M}''$ for SHA-0

Perturbation in round i	$f^{(i+2)}$	case	$f^{(i+3)}$	case	$f^{(i+4)}$	case	overall probability	probability logarithm
3	IF	2	IF	3	IF	3	1/32	5
11	IF	2	IF	3	IF	3	1/32	5
14	IF	2	IF	3	IF	3	1/32	4 + 1
22, 27, 28								
30, 31, 33	XOR	–	XOR	–	XOR	–	1/4	2
34, 35								
36	XOR	–	XOR	–	MAJ	4	1/4	2
37	XOR	–	MAJ	4	MAJ	4	1/4	2
38	XOR	–	MAJ	4	MAJ	3	1/8	3
40	MAJ	2	MAJ	3	MAJ	4	1/8	3
41	MAJ	2	MAJ	4	MAJ	3	1/8	3
43	MAJ	2	MAJ	3	MAJ	3	1/16	4
46	MAJ	2	MAJ	3	MAJ	3	1/16	4
51	MAJ	2	MAJ	3	MAJ	3	1/16	4
53	MAJ	2	MAJ	3	MAJ	3	1/16	4
55	MAJ	2	MAJ	3	MAJ	3	1/16	4
58, 60, 62								
66, 68, 69	XOR	–	XOR	–	XOR	–	1/4	2
70, 73, 74								

3 SHA-1 Case

In the SHA-1 case, the bits are interleaved and therefore it is no more possible to split the expansion in 32 little expansions. However, the invariance by translation is still true. Hence, it is still feasible to deduce the 5 corrective masks from a perturbative one, using the construction of Sect. 2.1.

More precisely, given a perturbative mask M_0 that is an output of E_1, Equ. (3) to (7) still hold, and the constructed mask M defined by (8) is again an output of E_1.

Finding the perturbative mask M_0 can be done using coding theory tools [3], because the mask can be considered as a low-weight codeword of the extension. Performing such a search on E_1 leads to some very short codewords as compared to the dimensions of the code. However, with very high probability, no codeword of weight less than 100 exists in E_1, that satisfies the constraints (see Sect. 2.1), whereas there exists 27 weighted codewords in E_0.

As every bit of the perturbative mask M_0 implies at least a factor 1/4 in the overall probability of success, our attack will therefore be totally inefficient on SHA-1.

However, it remains an open problem to see if differential masks exist in the SHA-1 case, because our attack builds very specific masks.

4 Conclusion

We have developed a new kind of attack on SHA functions that yields better results than the classical birthday-paradox attack on SHA-0. This attack is related to the well known differential cryptanalysis [1] in that it looks for some kind of characteristic masks that can be added to input word with non trivial probability of unchanging the output of the compression function. The expansion of SHA-1 seems to be designed to counter this kind of attack, which should increase the level of confidence in this standard.

5 Acknowledgments

We wish to thank Matthew Robshaw and the referees for their valuable remarks and improvements to this paper.

References

1. E. Biham, and A. Shamir. Cryptanalysis of the Full 16-Round DES, *CRYPTO'92* LNCS 740, pp 487–496, 1993.
2. B. den Boer, and A. Bosselaers. Collisions for the compression function of MD5, *EUROCRYPT'93* LNCS 773, pp 293–304, 1994.
3. A. Canteaut, and F. Chabaud. A new algorithm for finding minimum-weight words in a linear code: Application to primitive narrow-sense BCH codes of length 511, *IEEE Trans. Inform. Theory*, IT-44(1), pp 367–378, Jan. 1998.
4. H. Dobbertin. Cryptanalysis of MD4, *Fast Software Encryption* LNCS 1039, pp 53–69, 1996.
5. R. Rivest. The MD4 Message-Digest Algorithm, *CRYPTO'90* LNCS 537, pp 303–311, 1991.
6. R. Rivest. The MD5 Message-Digest Algorithm, *Network Working Group Request for Comments: 1321*, April 1992. `http://theory.lcs.mit.edu/~rivest/Rivest-MD5.txt`
7. Secure Hash Standard. *Federal Information Processing Standard Publication # 180*, U.S. Department of Commerce, National Institute of Standards and Technology, 1993.
8. Secure Hash Standard. *Federal Information Processing Standard Publication # 180-1*, U.S. Department of Commerce, National Institute of Standards and Technology, 1995 (addendum to [7]).