J Cryptol (2015) 28:110–160
DOI: 10.1007/s00145-014-9179-8

Journal of
CRYPTOLOGY

Cryptanalysis of SHA-0 and Reduced SHA-1

Eli Biham*, Rafi Chen*
Computer Science Department, Technion – Israel Institute of Technology, Haifa 32000, Israel
biham@cs.technion.ac.il; rafi_hen@inter.net.il; http://www.cs.technion.ac.il/~biham/

Antoine Joux
Laboratoire PRISM, CNRS UMR-8144, Université de Versailles St-Quentin-en-Yvelines, 45, avenue des Etats-Unis, 78035 Versailles Cedex, France
antoine.joux@m4x.org

Communicated by Preneel

Received 1 November 2008
Online publication 31 May 2014

Abstract. We present new techniques for the cryptanalysis of hash functions. Our contributions are two-fold: both on the search level of the compression function and on the meta-structure. The former led to the *neutral bits technique*, while the latter led to the *multi-block technique*. The usefulness of these techniques is demonstrated on SHA-0 and SHA-1, but they are applicable to other hash functions as well. We use these techniques to find a collision of the full SHA-0 which is the first published collision of this function, and very efficient collision attacks on reduced versions of SHA-1.

Keywords. Differential cryptanalysis, SHA-1, Hash functions.

1. Introduction

The cryptographic hash function SHA (called SHA-0 in this paper) was designed by the National Security Agency (NSA) and issued by NIST in 1993 as a Federal Information Processing Standard (FIPS-180) [28]. A revised version, called SHA-1, which introduces an additional rotate operation in the message expansion, was later issued in 1995 as FIPS-180-1 [29]. The revised version is intended to be a more secure replacement, that improves the security provided by the hash function. No details of the weaknesses found in SHA-0 were provided.

SHA-0 and SHA-1, as well as most hash functions currently in use, are based on Merkle–Damgård construction [12,25]. This construction is proved to be collision resistant if the underlying compression function is collision resistant. Our techniques use

* This work was supported in part by the Israel MOD Research and Technology Unit.

differential cryptanalysis [6] in a way that exploits the iterative nature of the Merkle–Damgård construction as well as weaknesses of the compression functions. The differences we use are in both inputs of the compression function, i.e., the initial values and the message blocks, whereas previous techniques were limited to differences in the message blocks. This extension allows us on one hand to use characteristics with much higher probabilities, but on the other hand they do not lead to collisions, but to near-collisions. In order to utilize the high probability characteristics that predict near-collisions, we concatenate them to a longer characteristic that predicts a collision in a technique we call *multi-block technique*. It should be noted that Wang et al. [36,37] independently used two-block messages 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.

The compression functions of SHA-0 and SHA-1 (as well as RIPMD-128 [17], RIPEMD-160 [18], HAVAL [42]) are based on the principles of MD4 [31] and MD5 [32]. These compression functions take a 512-bit message block, and output a pseudo-random string (128 bits in MD4 and MD5, and 160 bits in SHA-0 and SHA-1). The basic components of these functions are a message expansion, and an iterative round function that manipulates one of the expanded message words with some intermediate values. The round functions use the IF, XOR, MAJORITY, and addition modulo 2^{32} operations to manipulate the data.

1.1. *Previous Attacks on MD4/SHA Family of Hash Functions*

Shortly after Rivest introduced MD4 [31] in 1990, Merkle [26] showed in an unpublished work that it is possible to find a collision of MD4 reduced to the first 32 (out of the 48) rounds of MD4. Den Boer and Bosselaers [8] followed Merkle's work and presented an attack on the last 32 rounds of MD4 (rounds 16–47). Few years later, Dobbertin [15] presented a full collision of MD4 with a complexity of 2^{22}. His attack is differential, and the measure of difference is subtraction. The two colliding messages that he found differ in a single word by three consecutive bits, where the subtraction difference is 1. The attack is split into two parts: In the first part, a set of equations that describes the evolution of the differences is solved to achieve a predefined difference at some Round i. In the second part, a differential attack that starts with the difference of the first part and leads to a collision is performed. In the first part of the attack an attacker has a full control on the intermediate data, thus he reaches the predefined difference with a negligible complexity. The second part has a probability of about 2^{-22}, thus the overall complexity is about 2^{22}.

At CRYPTO'91 Rivest introduced MD5 [32] as a strengthened version of MD4. Two years later at EUROCRYPT'93 [9], den Boer and Bosselaers presented an attack that easily finds collisions of the compression function (also known as *pseudo-collisions*). Their attack finds two colliding inputs, each consists of an initial value and a message block, where the two initial values are different, but the two message blocks are identical. Though their attack showed a substantial weakness of the function, MD5 became the de facto standard of the industry in the following years. In the rump session of EUROCRYPT'96 [16] Dobbertin presented an attack on the compression function of MD5 that finds collisions of two different message blocks with a chosen (non-standard) initial value.

At CRYPTO'98, Chabaud and Joux [10] proposed a theoretical differential attack on the full SHA-0 with a complexity of 2^{61}, using a weakness of the expansion algorithm. Their attack is faster than the generic birthday attack, and partially explains the withdrawal of SHA-0 by NIST. This attack uses the XOR operation as a measure of difference, and the characteristic is determined by approximating the non-linear operations by XOR. Similarly to the attacks on MD4, the attack on SHA-0 is split into two parts. In the first 18 rounds, an attacker has almost full control on the conditions that a message should satisfy. From Round 19, the attack is probabilistic, and the chosen characteristic determines the complexity of the attack. Since our attack is based on this attack, we give a detailed description of it in Sect. 3.

For completeness we describe attacks that were published in parallel to or after our contribution in Table 1 and Sect. 9.

1.2. *Our Contribution*

This paper presents two cryptanalysis tools: the *neutral bits technique* (first described in [3]), and the *multi–block technique* (first described in [4,7]). The neutral bits technique is used to attack the compression function by using a poor avalanche of the round function, and the multi-block technique uses the iterative mode-of-operation of Merkle–Damgård to enable efficient attacks. The relevance of these techniques to attack SHA-0 and SHA-1 was presented in the rump session of CRYPTO 2004 in the sessions "New results on SHA-0 and SHA-1" by Biham and Chen [4] and "Collisions in SHA-0" by Joux [20].

We define the notion of neutral bits to describe many bits of a pair of messages that do not affect the differences and conditions that a pair should satisfy for a collision to occur. These neutral bits allow an attacker to start the attack from Round 22 or later[1] by eliminating the probabilistic behavior of prior rounds.

We analyze the complexity of attacking reduced and extended versions of SHA-0, and show that their complexities are not monotonous in the number of rounds. We then observe that characteristics that predict collisions of reduced and extended versions of SHA-0 may also be used to find small differences in the chaining values of the full 80-round SHA-0. Following this observation, we discuss the usefulness of characteristics that start with a zero or small difference in the chaining value and predict a collision or a small difference of the chaining value. We show that the complexity of finding a pair of message blocks that creates a small difference in the chaining values is much lower than a pair that creates a collision. The reason is that for the former we use characteristics with *any* differences at the message block, input chaining value, and output chaining value, while for the latter the differences are limited to the message block, and the differences of the input and output chaining values are zero. We then introduce the multi-block technique that links the different types of characteristics to produce a collision of a multi-block pair with much lower complexity than a collision of a single-block pair.

In the following subsection we present the results we achieved using these techniques. We note that although all of our examples are on the SHA family of hash functions, the techniques presented in this paper are general and may be used to cryptanalyze other

[1] SHA-0 and SHA-1 have 80 rounds in their compression functions.

hash functions. We also note that the neutral bits technique was found applicable for the cryptanalysis of stream ciphers as well [1,19,24].

1.3. *Results*

The applicability of the neutral bits and multi-block techniques is demonstrated on SHA-0 and SHA-1. In Table 1 we summarize the main results on attacking these functions in a chronological order. We start from the attack of Chabaud and Joux in 1998 then our results in 2004–2005 and other substantial results until 2013 (a short description of these later results is given in Sect. 9). The first column specifies the attacked hash function, and the next two columns specify the number of rounds of the attacked function and the number of message blocks used in the attack. The attack complexity is given in number of message pairs under the Pair column, and in number of SHA calls under the Time column. These two measures are given since some previous papers use the number of tested pairs as a measure of attack complexity, while others use number of SHA calls. In the remainder of this paper, we use number of SHA calls as a measure of attack complexity. In the Found column a "+" indicates that a colliding pair is found. The Cite column cites the publication of the result with our results in boldface. The last column specifies the year in which the result was published.

Table 1. Up-to-date results on SHA-0 and SHA-1.

	Rnd	Blocks	Pairs	Time	Found	Cite	Year
SHA-0	80	1	2^{61}	2^{58}		Chabaud and Joux [10]	1998
	80*	**1**	$\mathbf{2^{43}}$	$\mathbf{2^{40}}$		[2,3] **and Sect.** 4	**2004**
	82	**1**	$\mathbf{2^{43}}$	$\mathbf{2^{40}}$		[2,3] **and Sect.** 4	**2004**
	50	**2**	$\mathbf{2^{19}}$	$\mathbf{2^{16}}$	**+**	[2,11]	**2004**
	80	**4**	$\mathbf{2^{51}}$	$\mathbf{2^{46}}$	**+**	[7,20] **and Sect.** 7	**2004**
	80	2		2^{39}	+	Wang et al. [38]	2005
SHA-1	**34**	**1**	$\mathbf{2^{7}}$	$\mathbf{2^{4}}$	+	[2,4,7]	**2004**
	36	**2**	$\mathbf{2^{24}}$	$\mathbf{2^{21}}$	**+**	[2,4,7]	**2004**
	40	**2**	$\mathbf{2^{19}}$	$\mathbf{2^{16}}$	**+**	[4,7] **and Sect.** 8.3	**2004**
	53	1	2^{71}	2^{68}		Rijmen and Oswald [30]	2005
	53	**1**	$\mathbf{2^{60}}$	$\mathbf{2^{57}}$		[7]	**2005**
	58	**2**	$\mathbf{2^{75}}$	$\mathbf{2^{72}}$		[7]	**2005**
	58	1		2^{33}	+	Wang et al. [39]	2005
	80	2		2^{69}		Wang et al. [39]	2005
	80	2		2^{63}		Wang et al. [40]	2005
	64	2		2^{35}	+	De Cannière and Rechberger [13]	2006
	70	2		2^{44}	+	De Cannière et al. [14]	2007
	72	2		$2^{47.6}$	+	Grechnikov [21]	2010
	73	2		$2^{50.7}$	+	Grechnikov [21]	2010
	80	3		2^{58}		Chen [11]	2011
	75	2		2^{57}	+	Grechnikov and Adinetz [22]	2011
	80*	1		$2^{57.5}$		Stevens [33]	2013

Our results are in boldface
* Denotes near-collision

The full collision of SHA-0 is the first published collision of SHA-0, and it uses the neutral bits and multi-block techniques along with additional improvements. This result is an improvement by a factor of 2^{10} to the best previously known attack on SHA-0 [10].

Our attacks and collisions on reduced SHA-1 are the first published attacks on this function [4]. Each attack shows different weaknesses of the algorithm: SHA-1 reduced to 34 rounds is the easiest to attack, thus we were able to find colliding messages with ASCII letters and even meaningful words. The attack on SHA-1 reduced to 36 rounds shows a workaround for a limitation that was identified in [10] as "the consecutive disturbance problem in the IF rounds". It also demonstrates how the first block in the multi-block technique can be used to replace the standard initial values, in case they are incompatible with the characteristic. The attacks on SHA-1 reduced to 53 and 58 rounds were the highest reduced versions. we could attack with a complexity less than the generic birthday attack. In these attacks we use the technique mentioned above to resolve the consecutive disturbances problem. In parallel to our results, Rijmen and Oswald independently studied reduced versions of SHA-1, and found a characteristic of SHA-1 reduced to 53 rounds [30].

1.4. *Paper Organization*

This paper is organized as follows: Sect. 2 describes SHA-0 and SHA-1, and Sect. 3 reviews the original attack of [10] on SHA-0. Section 4 defines neutral bits, describes how to find and use them, and gives an example of such bits in SHA-0. Section 5 presents analysis of attacks on variants of SHA-0 with different number of rounds. In Sect. 6 we describe the multi-block technique, define near-collisions, pseudo-collisions and near-pseudo collisions, and show how they are used to construct an attack. In Sect. 7 we give a four-block collision of the full SHA-0, along with a further refinement of the prior techniques that make them applicable. Section 8 describes the extension to variants of SHA-1, and various attacks and results on reduced versions of SHA-1. Section 9 describes remarkable advances in the last few years. Finally, Sect. 10 summarizes the paper.

2. Description of SHA-0, SHA-1, and Notations

Throughout this paper, big-endian is used to convert words into bit strings, i.e., the first bit position is the most significant bit. A 32-bit word $\{b_{31}, \ldots, b_0\}$ is converted to an integer by $\sum_{i=0}^{i=31} b_i \cdot 2^i$. A right shift of a 32-bit word by r positions, where r zeroes are appended to the leftmost $32 - r$ bits of the shifted word, is denoted by $\gg r$. Similarly, a left shift is denoted by $\ll r$. $X^{\lll r}$ and $X^{\ggg r}$ denote a left and a right rotation of X by r positions, respectively.

2.1. *SHA-0 and SHA-1 Algorithms*

SHA-0 and SHA-1 are Merkle–Damgård iterative hash functions using specially designed compression functions. The Merkle–Damgård construction is outlined in Fig. 1. In this figure the sizes (in number of bits) of each message block M_i and chaining vari-

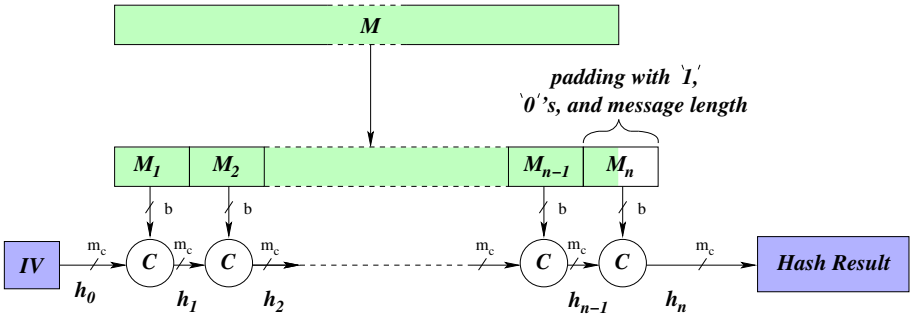

Fig. 1. Merkle–Damgård construction.

able h_i are denoted by b and m_c, respectively. The construction and the padding are as follows:

1. A single "1" bit is appended to the message M, followed by a variable number of "0" bits, followed by a 64-bit representation of the message length in bits. The number of zeroes is in the range $\{0, \ldots, 511\}$ such that the total length of the padded message is a multiple of 512 bits. The padded message is divided into blocks of 512 bits each: $M_1, \ldots, M_n$.
2. A chaining variable h_0 of five 32-bit words is initialized to

$$h_0 = (\texttt{67452301}, \texttt{EFCDAB89}, \texttt{98BADCFE}, \texttt{10325476}, \texttt{C3D2E1F0}).$$

3. For $k = 1$ to n, call the compression function with the current block M_k (512 bits) and the current chaining variable h_{k-1} (160 bits). In each iteration the output is a 160-bit chaining variable h_k:

$$h_k = \mathrm{C}(M_k, h_{k-1}).$$

4. h_n is the output of the hash function.

We give here a non-standard (but equivalent) description of the compression function of SHA-0 and SHA-1, which we found more convenient for the purpose of cryptanalysis. A traditional description of SHA-0 and SHA-1 is given in [29]. In the traditional description, five registers A_i, B_i, C_i, D_i, and E_i are used along with a word W_i from the *expanded message* $\bar{W}$ to compute the values of A_{i+1}, B_{i+1}, C_{i+1}, D_{i+1}, and E_{i+1}, respectively. In our description we use five entries A_{i-4}, A_{i-3}, A_{i-2}, A_{i-1}, and A_i of a *reduced state vector* A (that stores the values of A_i, B_i, C_i, D_i, and E_i up to a rotation), along with W_i to compute the value of the next entry A_{i+1}. We define a *state of the compression function* s_i, $i \in \{0, \ldots, 80\}$, and a transformation to the five registers representation by:

$$s_i = (A_i, A_{i-1}, A_{i-2}^{\lll 30}, A_{i-3}^{\lll 30}, A_{i-4}^{\lll 30}) = (A_i, B_i, C_i, D_i, E_i).$$

An illustration of this description is given in Fig. 2. The definition of the compression function of SHA-0 and SHA-1 and the computations of the entries of $\bar{W}$ and $\bar{A}$ are as follows:

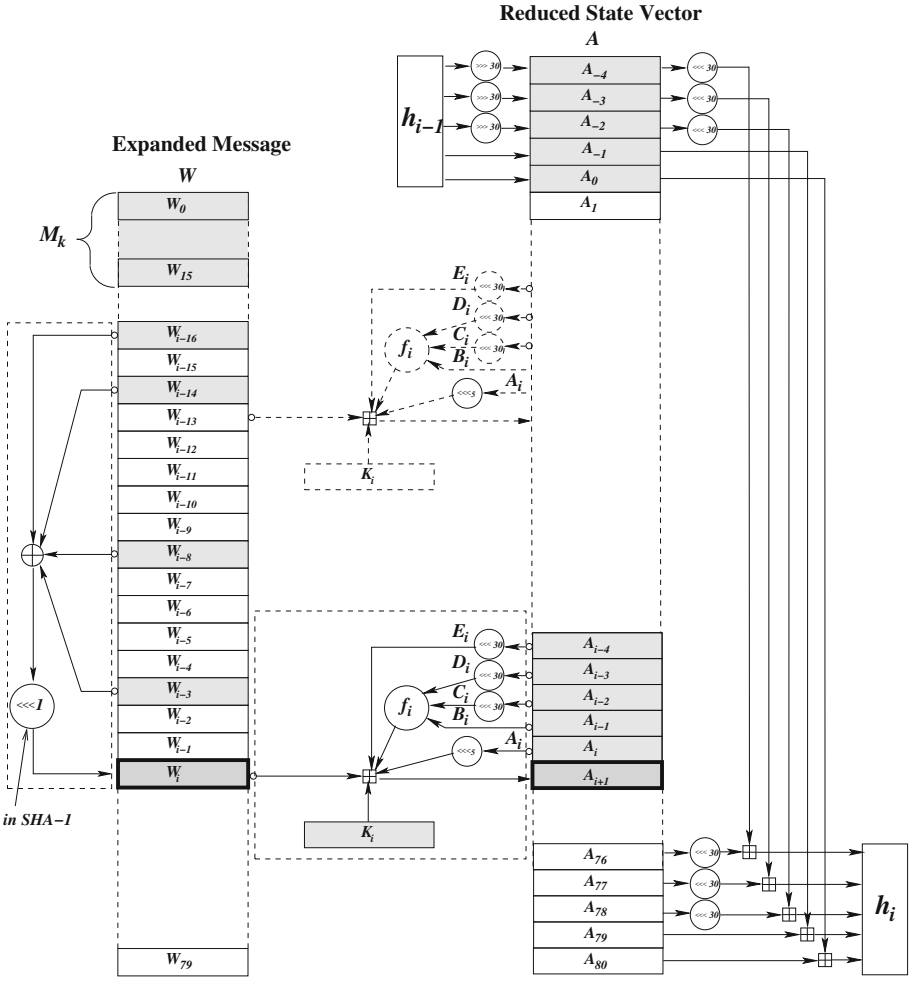

Fig. 2. The compression function of SHA-0 and SHA-1.

1. Divide the message block M_k into 16 words of 32 bits: $M_k = W_0, \ldots, W_{15}$.
2. Expand the 16 words to 80 words by the recurrence equation:

$$W_i = (W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16})^{\lll r}, \quad i = 16, \ldots, 79, \tag{1}$$

where r is 0 for SHA-0 and 1 for SHA-1. This rotate operation is the only difference between SHA-0 and SHA-1. We note that Eq. (1) represents a 16-word linear feedback shift register (LFSR). The LFSR is loaded with the initial value $W_0, \ldots, W_{15}$, and it is clocked 64 times to generate $W_{16}, \ldots, W_{79}$.

3. Divide h_{k-1} to the five first entries of the reduced state vector by

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

Table 2. Functions and constants.

Round	Function	$f_i(X, Y, Z)$	K_i
$0 \le i \le 19$	IF	$XY \vee \bar{X}Z$	5A827999
$20 \le i \le 39$	XOR	$X \oplus Y \oplus Z$	6ED9EBA1
$40 \le i \le 59$	MAJ	$XY \vee XZ \vee YZ$	8F1BBCDC
$60 \le i \le 79$	XOR	$X \oplus Y \oplus Z$	CA62C1D6

and

$$(A_0, A_{-1}, A_{-2}, A_{-3}, A_{-4}) = (A_0, B_0, C_0^{\ggg 30}, D_0^{\ggg 30}, E_0^{\ggg 30}). \quad (2)$$

4. For $i = 0$–79 compute the reduced state A_{i+1} by the following round function (Rounds $0, \ldots, 79$):

$$A_{i+1} = A_i^{\lll 5} + f_i(A_{i-1}, A_{i-2}^{\lll 30}, A_{i-3}^{\lll 30}) + A_{i-4}^{\lll 30} + W_i + K_i, \quad (3)$$

   where the functions f_i and the constants K_i are given in Table 2.
5. The output of the compression function is the sum of the final state s_{80} and the last chaining variable:

$$h_k = h_{k-1} + s_{80}, \quad (4)$$

   where the addition is word-wise modulo 2^{32}.

2.2. *Notations*

Unless it is explicitly written, an index in subscript denotes an index of a word in a vector or a round number. An index in superscript denotes a bit index, e.g., A_i^j is bit j of the reduced state vector word A_i. The function f_i denotes the 32-bit result of $f_i(A_{i-1}, A_{i-2}^{\lll 30}, A_{i-3}^{\lll 30})$ in Round i. In SHA-0 and SHA-1 f_i is a bit-wise function that processes each bit location independently, hence, we may use the notation $f_i^j(A_{i-1}^j, A_{i-2}^{j-30}, A_{i-3}^{j-30})$ to describe the j'th bit of its output. We sometimes use the explicit name of the function instead of f_i, i.e., instead of f_i we may write IF_i, XOR_i, or MAJ_i.

We use the notation CARRY_i^j to describe the value of the (single) carry bit from bit $j-1$ (and prior bits) to bit j in the computation of A_i by Eq. (3), i.e., the XOR difference of the output and all inputs of the addition operations in a SHA-round, which is

$$\mathrm{CARRY}_i = A_i \oplus A_{i-1}^{\lll 5} \oplus f_{i-1}(A_{i-2}, A_{i-3}^{\lll 30}, A_{i-4}^{\lll 30}) \oplus A_{i-5}^{\lll 30} \oplus W_{i-1} \oplus K_{i-1}. \quad (5)$$

We use the standard notations of differential cryptanalysis [5] to specify the values and differences of two messages and their parameters, e.g., M, M^* and M' describe the values and the difference of two messages, respectively. In addition, a bit is marked in boldface, e.g., $\mathbf{W}_i^1$, to indicate that the value of this bit is different in both messages, e.g., that $W_i'^1 = W_i^1 \oplus W_i^{*1} = 1$. We call such a bit an *active bit*.

3. The Basic Attack on SHA-0

In [10], Chabaud and Joux present a differential attack on SHA-0 that uses the XOR operation as a measure of difference. Their attack is aimed at finding a collision of a pair of single-block messages that are hashed with the standard initial value.

The basic idea of the attack is to generate a pair of messages with specific patterns of XOR differences in the first 16 words. Each pattern of differences that starts at Round i is constructed such that with a non-negligible probability it creates a difference of a single bit between the reduced states A_{i+1} and A^*_{i+1}. After five rounds, at Round $i+6$, the reduced states A_{i+1} and A^*_{i+1} do not affect the computations of succeeding reduced states, thus the states of the two messages are not affected by the difference that the pattern created. These patterns of differences are duplicated by the LFSR of Eq. (1) to the expanded messages. The result is a pair of expanded messages with patterns of differences such that each pattern creates with some probability a difference of a single bit between the reduced states. If each pattern in the expanded messages succeeds in creating a difference of a single bit, then the differences in the reduced states follow the patterns of differences in the expanded messages. Hence, if the difference that the last pattern creates is in A'_{75} (or before), then the last five reduced states are equal and a collision occurs.

The basic pattern of differences between the two messages that creates such equal states consists of six XOR differences. The first difference creates a minimal difference of one bit in the reduced state vector, and the other five differences avoid the propagation of this difference to the next words of the reduced state vector. We call such a pattern of differences, a *local collision sequence*. The next subsection describes a local collision sequence in detail.

3.1. *A Local Collision Sequence*

In the attacks on SHA-0 we concentrate only on local collision sequences that start at bit 1 of any of the 32-bit message words. Such a pattern that creates a *local collision* starts with a single-bit difference at $\mathbf{W}^1_i$, which we call a *disturbance* (or *perturbation*). With a probability 1/2 this disturbance creates a difference in $\mathbf{A}^1_{i+1}$ while leaving the carry to A^2_{i+1} unchanged. In order to avoid the propagation of the difference from $\mathbf{A^1_{i+1}}$ to the next rounds, five additional differences, called *corrections*, are inserted in the next five rounds. In Table 3 we demonstrate a disturbance and five corrections which form a local collision sequence. This table gives bit-wise computations of the round function for the bits that are affected by the active bits of the local collision sequence. The local collision sequence we show is: ($\mathbf{W^1_i}$, $\mathbf{W^6_{i+1}}$, $\mathbf{W^1_{i+2}}$, $\mathbf{W^{31}_{i+3}}$, $\mathbf{W^{31}_{i+4}}$, $\mathbf{W^{31}_{i+5}}$), and the desired resulting difference in the reduced state vector is $\mathbf{A^1_{i+1}}$ only, i.e., $\mathbf{W^1_i}$ activates $\mathbf{A^1_{i+1}}$, and $\mathbf{W^6_{i+1}}$, $\mathbf{W^1_{i+2}}$, $\mathbf{W^{31}_{i+3}}$, $\mathbf{W^{31}_{i+4}}$, $\mathbf{W^{31}_{i+5}}$ result in A^6_{i+2}, A^1_{i+3}, A^{31}_{i+4}, A^{31}_{i+5}, and A^{31}_{i+6} inactive. In terms of intermediate states differences, the state differences $s'_{i+1}, \ldots, s'_{i+5}$ show differences due to $\mathbf{A^1_{i+1}}$, while s'_{i+6} is not affected, i.e.,

$$s'_{i+1} = (00000002, 0, 0, 0, 0)$$
$$s'_{i+2} = (0, 00000002, 0, 0, 0)$$

Table 3. A description of a pattern of differences that creates a local collision.

A disturbance												
$\mathbf{A^1_{i+1}}$	$=$	A_i^{28}	$\oplus$	$\mathrm{f}_i^1(A_{i-1}^1, A_{i-2}^3, A_{i-3}^3)$	$\oplus$	A_{i-4}^3	$\oplus$	$\mathbf{W^1_i}$	$\oplus$	K_i^1	$\oplus$	CARRY_{i+1}^1
Five corrections												
A_{i+2}^6	$=$	$\mathbf{A^1_{i+1}}$	$\oplus$	$\mathrm{f}_{i+1}^6(A_i^6, A_{i-1}^8, A_{i-2}^8)$	$\oplus$	A_{i-3}^8	$\oplus$	$\mathbf{W^6_{i+1}}$	$\oplus$	K_{i+1}^6	$\oplus$	CARRY_{i+2}^6
A_{i+3}^1	$=$	A_{i+2}^{28}	$\oplus$	$\mathbf{f^1_{i+2}}(\mathbf{A^1_{i+1}}, A_i^3, A_{i-1}^3)$	$\oplus$	A_{i-2}^3	$\oplus$	$\mathbf{W^1_{i+2}}$	$\oplus$	K_{i+2}^1	$\oplus$	CARRY_{i+3}^1
A_{i+4}^{31}	$=$	A_{i+3}^{26}	$\oplus$	$\mathbf{f^{31}_{i+3}}(A_{i+2}^{31}, \mathbf{A^1_{i+1}}, A_i^1)$	$\oplus$	A_{i-1}^1	$\oplus$	$\mathbf{W^{31}_{i+3}}$	$\oplus$	K_{i+3}^{31}	$\oplus$	$\mathrm{CARRY}_{i+4}^{31}$
A_{i+5}^{31}	$=$	A_{i+4}^{26}	$\oplus$	$\mathbf{f^{31}_{i+4}}(A_{i+3}^{31}, A_{i+2}^1, \mathbf{A^1_{i+1}})$	$\oplus$	A_i^1	$\oplus$	$\mathbf{W^{31}_{i+4}}$	$\oplus$	K_{i+4}^{31}	$\oplus$	$\mathrm{CARRY}_{i+5}^{31}$
A_{i+6}^{31}	$=$	A_{i+5}^{26}	$\oplus$	$\mathrm{f}_{i+5}^{31}(A_{i+4}^{31}, A_{i+3}^1, A_{i+2}^1)$	$\oplus$	$\mathbf{A^1_{i+1}}$	$\oplus$	$\mathbf{W^{31}_{i+5}}$	$\oplus$	K_{i+5}^{31}	$\oplus$	$\mathrm{CARRY}_{i+6}^{31}$

$$
\begin{aligned}
s'_{i+3} &= (0, 0, 80000000, 0, 0)\\
s'_{i+4} &= (0, 0, 0, 80000000, 0)\\
s'_{i+5} &= (0, 0, 0, 0, 80000000)\\
s'_{i+6} &= (0, 0, 0, 0, 0).
\end{aligned}
$$

During a desired computation of a pair of messages that has a local collision sequence difference (as in Table 3), only the bits marked in boldface may be active. Thus, we require that the carries in the computations of A_{i+1}, A_{i+2}, and A_{i+3}, (at rounds i, $i+1$, and $i+2$) remain unchanged (inactive). We note that since the addition is modulo 2^{32}, there is no carry to bits next to bit 31. Thus, even though a disturbance may be located at any bit index, we prefer bit 1, since the probabilistic behavior of the carry after the rotation by 30 bits is eliminated at rounds $i+3$, $i+4$, and $i+5$.

An illustration of a local collision sequence is given in Fig. 3. In this figure the disturbance $\mathbf{W^1_i}$ and the difference of the reduced state $\mathbf{A^1_{i+1}}$ are marked in dark gray. The entries in light gray are the corrections, and entries with no background have

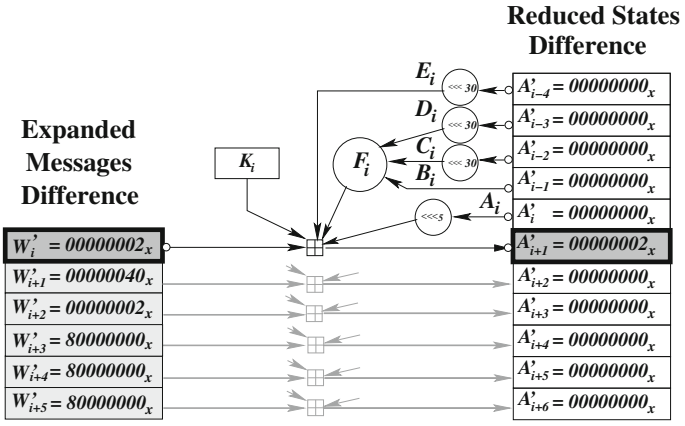

Fig. 3. A local collision sequence.

no difference. The middle part of this figure shows the computation of the reduced state difference A'_{i+1} with the inputs of the disturbance $\mathbf{W'_i} = \texttt{00000002}$ and the state differences $s'_i = (0, 0, 0, 0, 0)$. The desired result is $A'_{i+1} = \texttt{00000002}$, i.e., $s'_{i+1} = (\texttt{00000002}, 0, 0, 0, 0)$. In Round $i + 1$ the middle part of the figure is advanced by one entry such that its input differences are: $\mathbf{W'_{i+1}} = \texttt{40}$ and $s'_{i+1} = (\texttt{00000002}, 0, 0, 0, 0)$. The desired output difference of this round is $A'_{i+2} = 0$. In each consequent round the middle part in the figure is advanced by one entry, and the reduced state difference is computed. After five such computations, the state difference becomes $s'_{i+6} = (0, 0, 0, 0, 0)$, which forms a local collision.

A local collision sequence is a probabilistic process that depends on the function f_i and the carry. We now analyze each row of Table 3, and show how the desired differences are achieved, and what the probabilities are at each round.

A disturbance $\mathbf{W_i^1}$ turns $\mathbf{A_{i+1}^1}$ to be an active bit with probability 1. The carry from bit 1 remains unchanged if $\mathbf{W_i^1} = \mathbf{A_{i+1}^1}$, thus we assume a probability $1/2$ for this round. In the next five rounds we apply corrections that compensate the active bit $\mathbf{A_{i+1}}$, so that $A_{i+2}, \ldots, A_{i+6}$ remain inactive.

In the first correction at Round $i + 1$, the active bits $\mathbf{W_{i+1}^6}$ and $\mathbf{A_{i+1}^1}$ cancel each other if $\mathbf{W_{i+1}^6} = \overline{\mathbf{A_{i+1}^1}} = \overline{\mathbf{W_i^1}}$. Thus, by setting $\mathbf{W_i^1} = \overline{\mathbf{W_{i+1}^6}}$, both A_{i+2}^6 and the carry remain inactive. In the next three rounds the active bit $\mathbf{A_{i+1}^1}$ is an input to f_i. For simplicity and clarity we first consider f_i being XOR. We will later discuss the differences for the IF and MAJORITY operations.

At Round $i+2$ we need that A_{i+3}^1 and the carry from this bit remain inactive. The result of $\mathbf{XOR_{i+2}^1}(\mathbf{A_{i+1}^1}, A_i^3, A_{i-1}^3)$ is always active when $\mathbf{A_{i+1}^1}$ is active in the input of the XOR function, thus the correction $\mathbf{W_{i+2}^1}$ maintain A_{i+3}^1 inactive with probability 1. The carry from bit 1 remains inactive if $\mathbf{W_{i+2}^1} = \overline{\mathbf{XOR_{i+2}^1}}$. By writing the explicit terms of $\mathbf{XOR_{i+2}^1}$, and substituting $\overline{\mathbf{A_{i+1}^1}} = \mathbf{W_{i+1}^6}$, the carry remains inactive if $\mathbf{W_{i+2}^1} = \mathbf{W_{i+1}^6} \oplus A_i^3 \oplus A_{i-1}^3$, which occurs with probability $1/2$. We note that the condition of Round $i + 2$ can be tested at round $i - 1$ (when A_i^3 and A_{i-1}^3 are known). Thus, an attacker knows if he achieves the desired result three rounds in advance.

In the computations of A_{i+4}^{31}, A_{i+5}^{31}, and A_{i+6}^{31} we should not worry about changes of the carry as it is ignored anyway. Furthermore, $\mathbf{XOR_{i+3}^{31}}(A_{i+2}^{31}, \mathbf{A_{i+1}^1}, A_i^1)$ and $\mathbf{XOR_{i+4}^{31}}(A_{i+3}^{31}, A_{i+2}^1, \mathbf{A_{i+1}^1})$ are always active when only $\mathbf{A_{i+1}^1}$ is active in the input. Thus, in each of these three rounds the active bits $\mathbf{W}_{i+3}^{31}$, $\mathbf{W}_{i+4}^{31}$, $\mathbf{W}_{i+5}^{31}$ compensate the active bit $\mathbf{A_{i+1}^1}$ with probability 1.

The differences between the analysis of the XOR, IF, and MAJORITY operations are only in rounds $i + 2, i + 3$, and $i + 4$ where $\mathbf{A_{i+1}^1}$ is one of the inputs to the function. In these rounds we require $\mathbf{f_{i+2}^1}(\mathbf{A_{i+1}^1}, A_i^3, A_{i-1}^3)$, $\mathbf{f_{i+3}^{31}}(A_{i+2}^{31}, \mathbf{A_{i+1}^1}, \mathbf{A_i^1})$, and $\mathbf{f_{i+4}^{31}}(A_{i+3}^{31}, A_{i+2}^1, \mathbf{A_{i+1}^1})$ to be active, such that the corrections $\mathbf{W_{i+2}^1}$, $\mathbf{W_{i+3}^{31}}$, and $\mathbf{W_{i+4}^{31}}$ cancel each of these active bits, respectively. Unlike the XOR operation whose output is always active, IF and MAJORITY create an active output with probability $1/2$ for a *random selection* of their three inputs. In addition, at Round $i + 2$ we require

$$\mathbf{f_{i+2}^1}(\mathbf{A_{i+1}^1}, A_i^3, A_{i-1}^3) = \overline{\mathbf{W_{i+2}^1}} \tag{6}$$

Table 4. Local collision conditions for rounds $i, \ldots, i+5$ with the XOR operation at rounds $i+2$, $i+3$, $i+4$.

Round	Conditions	Prob.
i	$\mathbf{A^1_{i+1}} = \mathbf{W^1_i}$	1/2
$i+1$	$\mathbf{W^6_{i+1}} = \overline{\mathbf{W^1_i}}$	1
$i+2$	$\mathbf{W^1_{i+2}} = \mathbf{W^6_{i+1}} \oplus A^3_i \oplus A^3_{i-1}$	1/2
$i+3$	—	1
$i+4$	—	1
$i+5$	—	1

Table 5. Conditions for rounds $i+2$, $i+3$, $i+4$ with the IF operation.

Round	Conditions	Prob.
i	$\mathbf{A^1_{i+1}} = \mathbf{W^1_i}$	1/2
$i+1$	$\mathbf{W^6_{i+1}} = \overline{\mathbf{W^1_i}}$	1
$i+2$	$(A^3_i = \overline{A^3_{i-1}})$ and $\mathbf{W^1_{i+2}} = \mathbf{W^6_{i+1}} \oplus A^3_{i-1}$	1/4
$i+3$	$A^{31}_{i+2} = 1$	1/2
$i+4$	$A^{31}_{i+3} = 0$	1/2
$i+5$	—	1

Table 6. Conditions for rounds $i+2$, $i+3$, $i+4$ with the MAJORITY operation.

Round	Conditions	Prob.
i	$\mathbf{A^1_{i+1}} = \mathbf{W^1_i}$	1/2
$i+1$	$\mathbf{W^6_{i+1}} = \overline{\mathbf{W^1_i}}$	1
$i+2$	$A^3_i = \overline{A^3_{i-1}}$, $\mathbf{W^1_{i+2}} = \mathbf{W^6_{i+1}}$	1/2
$i+3$	$A^{31}_{i+2} = \overline{A^1_i}$	1/2
$i+4$	$A^{31}_{i+3} = \overline{A^1_{i+2}}$	1/2
$i+5$	—	1

in order for the carry to remain unchanged, i.e., the output of the function is active, *and* it is opposite to the correction.[2] When the function is the IF operation and its output is active, Eq. (6) is satisfied with probability 1/2. On the other hand, when f_i is MAJORITY, and it is active, $\mathbf{MAJ^1_{i+2}}(\mathbf{A^1_{i+1}}, A^3_i, A^3_{i-1}) = \mathbf{A^1_{i+1}}$, and by setting $\mathbf{W^1_{i+2}} = \overline{\mathbf{W^1_i}}$, the probability is 1. Thus, by adding a condition on the message bits which we control, we gain a factor of 1/2 in the probability.

A summary of conditions of a local collision sequence and their probabilities with the XOR operation are given in Table 4. The conditions and probabilities for the IF and MAJORITY are given in Tables 5 and 6. In these tables, the first column shows the round

[2] This requirement is not applicable to rounds $i+3$ and $i+4$ since at that rounds bit 31 is active, and the carry is not considered.

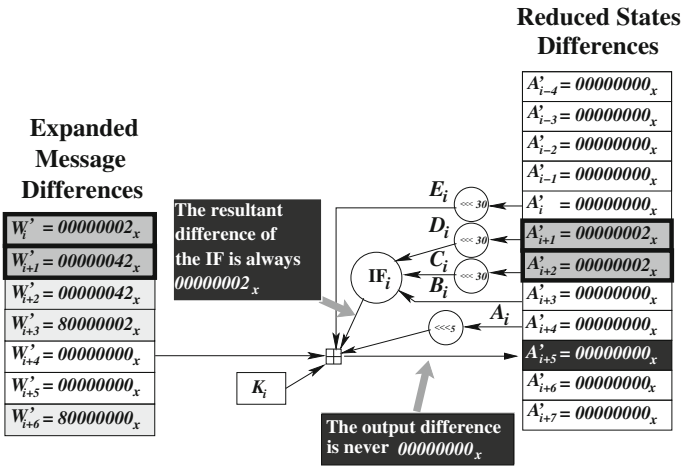

Fig. 4. Two consecutive disturbances in the IF rounds.

number, where the application of the disturbance is at round i. The second column shows the conditions on the expanded messages and reduced states bits. The third column is the probability that the conditions hold. We note that according to the data in these tables an attacker may know 2–3 rounds in advance if the required behavior of a local collision sequence is achieved, e.g., in Round i he can already test whether $A_i^3 = \overline{A_{i-1}^3}$ to know if he gains the desired behavior of Round $i + 2$.

3.1.1. *Two Consecutive Disturbances*

The computations of local collision sequences with two consecutive disturbances at rounds i and $i + 1$ are similar to the computations in Table 3, except for Round $i + 4$ where A_{i+5} is given by:

$$A_{i+5}^{31} = W_{i+4}^{31} \oplus A_{i+4}^{26} \oplus \mathrm{f}_{i+4}^{31}(A_{i+3}^{31}, \mathbf{A_{i+2}^1}, \mathbf{A_{i+1}^1}) \oplus A_i^1 \oplus K_{i+4}^{31} \oplus \mathrm{CARRY}_{i+5}^{31} \quad (7)$$

(see Fig. 4). At round $i + 4$, the two corrections for $\mathbf{A_{i+1}^1}$ and $\mathbf{A_{i+2}^1}$ cancel each other, thus W_{i+4}^{31} is inactive. On the other hand, when f_i is the IF operation, $\mathbf{IF}_{i+4}^{31}(A_{i+3}^{31}, \mathbf{A_{i+2}^1}, \mathbf{A_{i+1}^1})$ is always active with $\mathbf{A_{i+2}^1}$ and $\mathbf{A_{i+1}^1}$ active in the input. Thus, Eq. (7) is never satisfied when the IF operation is used. Hence, two consecutive disturbances are not allowed in the range $i = 0, \ldots, i = 16$. This situation is known as the *consecutive disturbances problem in the IF rounds*.[3]

A similar analysis when f_i is the MAJORITY operation shows that:

$$\mathrm{MAJ}_{i+4}^{31}(A_{i+3}^{31}, A_{i+2}^1, A_{i+1}^1) = \mathrm{MAJ}_{i+4}^{31}(A_{i+3}^{31}, \overline{A_{i+2}^1}, \overline{A_{i+1}^1}) \text{ if } A_{i+1}^1 = \overline{A_{i+2}^1}.$$

[3] Note that the resolution of this issue by Wang et al. [38] enables the usage of disturbance vectors with much higher probabilities, and as a result it reduced the complexity of the attack by a factor of about 2^{20}.

Thus, by keeping the condition

$$W_i^1 = \overline{W_{i+1}^1},$$

when two consecutive disturbances are in the range $i = 37, \ldots, i = 56$, the probability of the correction at round $i + 4$ is 1.

3.2. *A Characteristic and a Disturbance Vector*

The attack on SHA-0 is differential, and the differences we use have the form of local collision sequences. Each local collision sequence starts with a disturbance and leads to a local collision with some probability. It is clear that a pair of message blocks that has such differences collides if all the local collisions occur as expected. We now show how to construct a pair of message blocks that has such differences.

A pair of expanded message blocks is generated by the LFSR of Eq. (1), thus their XOR difference is also generated by this LFSR. We use this property to construct the XOR difference of the pair such that it has the form of local collision sequences, in two steps. Firstly, we construct the differences that represent the disturbances, and describe these differences by a *disturbance vector* $\mathcal{D}$. Secondly, we manipulate the disturbance vector $\mathcal{D}$ by the linear XOR and shift operations to construct the difference of the pair (in the form of local collision sequences). We describe this difference by a *disturbances and corrections vector* Δ. We note that since the operations we use for constructing these two vectors (the disturbance vector $\mathcal{D}$, and the disturbance and corrections vector Δ) are linear, both of them can be generated by the LFSR of Eq. (1).

Definition 1 A *disturbance vector* $\mathcal{D}$ is a vector of 85 words $\mathcal{D}_{-5}, \ldots, \mathcal{D}_{79}$, that indicates the locations of the disturbances. A bit in this vector is set to one if there is a disturbance in this location, and is set to zero otherwise.

In the following list we describe the constraints on the disturbance vector that are required for finding collisions of SHA-0.

1. Disturbances are applied only to bit 1 (see the discussion in Sect. 3.1).
2. The pair of messages starts with the standard initial value, i.e., the differences of the initial state are $s'_0 = (0, 0, 0, 0, 0)$, and the corresponding words of the disturbance vector are $\mathcal{D}_{-5} = 0, \ldots, \mathcal{D}_{-1} = 0$.
3. The differences of the final state are zero, i.e., $s'_{80} = (0, 0, 0, 0, 0)$, and the corresponding words of the disturbance vector are $\mathcal{D}_{75} = 0, \ldots, \mathcal{D}_{79} = 0$.
4. No consecutive disturbances are allowed in $\mathcal{D}_0, \ldots, \mathcal{D}_{16}$ (as mentioned earlier, this limitation was resolved by Wang et al. [38]).

Under these constraints we construct the disturbance vector using the LFSR of Eq. (1).

Given a disturbance vector, we define the difference of the analyzed expanded messages by *a disturbances and corrections vector* Δ. A disturbances and corrections vector Δ is generated from a disturbance vector $\mathcal{D}$ in the form of local collision sequences as follows: Let $\mathrm{SR}^l(\mathcal{D})$ be the vector of 85 words obtained by prepending l zero words to the first $85 - l$ words of $\mathcal{D}$ (i.e., a non-cyclic shift operation of the words). Then, for

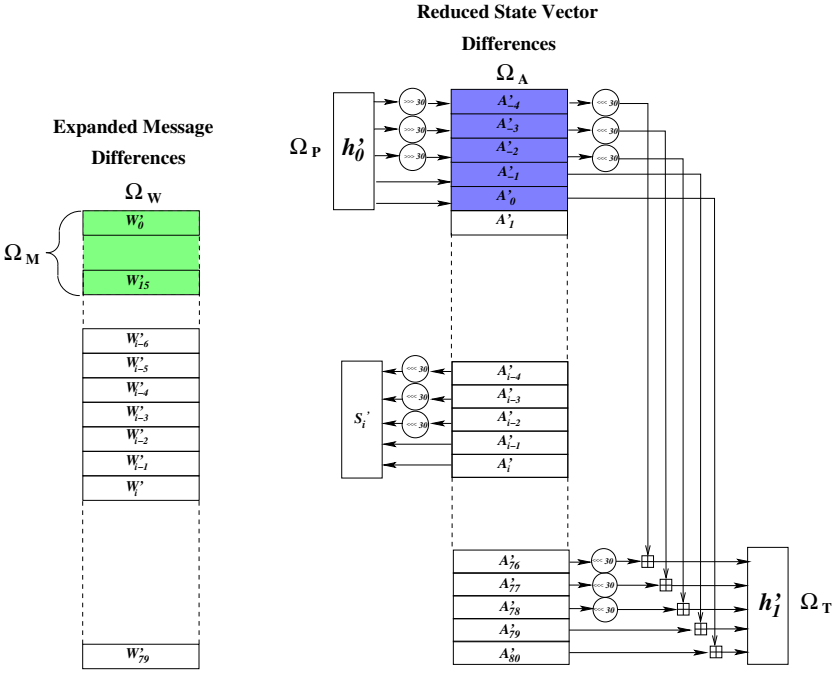

Fig. 5. A characteristic of a single-block attack.

all the disturbances in $\mathcal{D}$, corrections are made by $\mathrm{SR}^1(\mathcal{D})^{\lll 5}$, $\mathrm{SR}^2(\mathcal{D})$, $\mathrm{SR}^3(\mathcal{D})^{\lll 30}$, $\mathrm{SR}^4(\mathcal{D})^{\lll 30}$, and $\mathrm{SR}^5(\mathcal{D})^{\lll 30}$, where $\lll$ denotes a cyclic rotate of each word of the vector separately. The linear combination of local collision sequences that start in each disturbance of $\mathcal{D}$ is:

$$\mathcal{D} \oplus \left(\mathrm{SR}^1(\mathcal{D})^{\lll 5}\right) \oplus \mathrm{SR}^2(\mathcal{D}) \oplus \left(\left(\mathrm{SR}^3(\mathcal{D}) \oplus \mathrm{SR}^4(\mathcal{D}) \oplus \mathrm{SR}^5(\mathcal{D})\right)^{\lll 30}\right). \tag{8}$$

Δ is the 80 entries $0, \ldots, 79$, of the vector constructed by (8), and the difference W' of the two expanded message blocks is:

$$W' = (W'_0, \ldots, W'_{79}) = (\Delta_0, \ldots, \Delta_{79}) = \Delta.$$

3.2.1. *A Construction of Characteristics from Disturbance Vectors*

We are now ready to define a characteristic for the attack, and then we show how disturbance vectors are used to construct characteristics. The following definition is for a characteristic which is suitable to attack a pair of single-block messages (see Fig. 5). It is extended later to attack longer messages.

Definition 2 A characteristic of SHA-0 is a tuple $\Omega = (\Omega_P, \Omega_M, \Omega_W, \Omega_S, \Omega_T)$. Ω_P is a 160-bit string that represents the differences of the initial values, i.e., $\Omega_P = h_{k-1} \oplus h^*_{k-1}$. Ω_T is a 160-bit string that represents the differences of the hash results, i.e., $\Omega_T = h_k \oplus h^*_k$. Ω_P and Ω_T are also called *chaining differences*. In general we refer

to chaining differences as tuples of five words of 32 bits. $\Omega_M = M \oplus M^*$ is a 512-bit string of the messages difference, and $\Omega_W = (W'_0, \ldots, W'_{79})$ is the differences of the words of the expanded messages. Since the 512 bits of Ω_M are the first 16 words of W', we generally refer to them by the 16 words: $\Omega_M = (W'_0, \ldots, W'_{15})$. The *differences of the state vector* $\Omega_S = (s'_0, \ldots, s'_{80})$ start from a given difference of the initial values ($s'_0 = \Omega_P$), and continue with the state differences from the first computed state (s'_1) to the final state (s'_{80}). We note that Ω_S is derived from the differences of the *reduced state vector* $\Omega_A = (A'_{-4}, \ldots, A'_{80})$ by:

$$s'_i = (A'_i, A'_{i-1}, A'^{\lll 30}_{i-2}, A'^{\lll 30}_{i-3}, A'^{\lll 30}_{i-4}),\ i \in \{0, \ldots, 80\}.$$

We are now ready to describe how we use a disturbance vector to construct a characteristic. A characteristic that predicts a collision with a pair of single-block messages starts with the standard initial values and predicts equal hash results, thus $\Omega_P = (0, 0, 0, 0, 0)$ and $\Omega_T = (0, 0, 0, 0, 0)$. Using a disturbance vector $\mathcal{D}$ we generate the disturbances and corrections vector Δ and define the differences of the messages and expanded messages by $\Omega_M = (\Delta_0, \ldots, \Delta_{15})$ and $\Omega_W = (\Delta_0, \ldots, \Delta_{79})$. The predicted differences of the reduced state vector are then: $\Omega_A = (A'_{-4}, \ldots, A'_{80}) = (\mathcal{D}_{-5}, \ldots, \mathcal{D}_{79}) = \mathrm{SR}^1(\mathcal{D})$, and the predicted differences of the state vector $\Omega_S = (s'_0, \ldots, s'_{80})$ are: $s'_i = (\mathcal{D}_{i-1}, \mathcal{D}_{i-2}, \mathcal{D}^{\lll 30}_{i-3}, \mathcal{D}^{\lll 30}_{i-4}, \mathcal{D}^{\lll 30}_{i-5}), i \in \{0, \ldots, 80\}$. The probability of the characteristic depends on the number of disturbances, as discussed in Sect. 3.3.

Given a characteristic we measure how close a pair of messages is to a collision by the definition of conformance.

Definition 3 Given a characteristic Ω, a pair of messages M and M^* *conforms* to R rounds if

$$s'_0 = \Omega_P \text{ and } \forall_{i=1,\ldots,R}\ A'_i = \mathcal{D}_{i-1}.$$

The pair M and M^* is a *right pair* if it conforms to all rounds ($R = 80$ in SHA-0 and SHA-1).

We note that if a pair of messages conforms to the characteristic up to the last round of the compression function then a collision occurs. In our discussion on SHA-1 we relax the requirements for conformance and give a weaker definition.

3.2.2. *Compact Representations of a Disturbance Vector $\mathcal{D}$, Chaining Differences Ω_P, and Ω_T, and State Difference s'_i*

In SHA-0 disturbances are applied only to bit 1, thus a disturbance vector may be represented by the values of bit 1 in each word from -5 to 79. Furthermore, to identify a disturbance vector it suffices to specify the first 16 bits from which the LFSR may generate the complete vector. Thus, for SHA-0 we represent a disturbance vector, which is a vector of 85 words, by a single 16-bit word. In this representation each bit from the LSB to the MSB corresponds to a disturbance in $\mathcal{D}_{-5}, \ldots, \mathcal{D}_{10}$, respectively.

Similarly, we use a 5-bit word to represent the prediction of the chaining differences Ω_P, and Ω_T, and state difference s'_i. A difference in a word is represented by "1" and

Table 7. The disturbance vector of SHA-0 attack.

$\mathcal{D}$:	$-5, \ldots, 39$:	***00000***	*0010001000*	*0*0000101111	0110001110	0000010100
	$40, \ldots, 79$:		0100010010	0100111011	0011000011	11100**00000**

no difference by "0", e.g., $\Omega_T = \boldsymbol{10010}$ corresponds to differences in bit 1 of A'_{80} and A'_{77}, while A'_{79}, A'_{78}, and A'_{76} have no differences.

3.3. *Selecting a Disturbance Vector for the Attack*

A conformance of a pair of messages to 80 rounds for a given characteristic is a sufficient condition for a collision. If we assume that the probability of each local collision sequence is equal, then the fewer disturbances, the higher the probability to find a collision.

We now show that the requirement for the smallest number of disturbances can be relaxed and optimized for the attack such that disturbances are not counted from the first round. Note that up to Round 15 an attacker has full control over the results of the round function, thus we assume that the corrections of each condition up to Round 15 are made with negligible complexity. Moreover, following the analysis in Sect. 3.1, we see that an attacker has the ability to look ahead about three rounds. Thus, the conformance conditions up to Round 18 can be tested along with the computation of Round 15, and can be corrected with negligible complexity. We conclude that the disturbance vector with the least number of disturbances between Round 19 and the last round, has the highest probability to generate a collision.

We can now use the following procedure for the selection of the disturbance vector with the highest probability. A 16-word register is initialized such that the first five words are set to zero, and in the other 11 words bit 1 is set either to zero or one. We expand it to 85 words by the LFSR of Eq. (1). From the 2^{11} possible outcomes, 2^6 satisfy $\mathcal{D}_{75} = 0, \ldots, \mathcal{D}_{78} = 0$ and $\mathcal{D}_{79} = 0$, and have no consecutive disturbances in rounds $0, \ldots, 16$. From these 2^6 outcomes we select an outcome (an 85-word vector) which has the least Hamming weight from round 19 to 79.

Using the compact representation of a disturbance vector, the best disturbance vector for the attack is represented by the 16-bit string `0880`. The 85-bit string that describes the disturbance vector with the five leading zeros is shown in Table 7. In this table the entries of the disturbance vector from which the initial and final state differences are derived are marked with a boldface, and the first 16 bits that represent the disturbance vector are in italic.

3.4. *The Chaining Differences Transition Graph*

A graphical description of the selection of a characteristic for an attack is given in Fig. 6. In this figure we describe characteristics of an attack as edges in a directed graph, in which the vertices are the chaining differences. Each characteristic is derived from a disturbance vector $\mathcal{D}_i$, and has a probability p_i to achieve the differences given by its output edge. The vertex ***00000*** which is the initial *and* final chaining difference is drawn twice for clarity. The five zeros in it denote that there is no difference at that

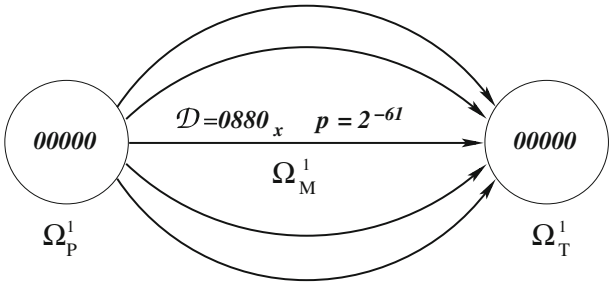

Fig. 6. The chaining differences transition graph of a single-block attack on SHA-0.

location. Among the different disturbance vectors that predict this chaining difference transition, the one that has the highest probability is selected for the attack. Following these notations, this simple graph shows the path from no difference in the initial values Ω^1_P = ***00000*** to no difference in the final hash result Ω^1_T = ***00000***, using a pair of single-block messages with Ω^1_M difference. We call this graph a *chaining differences transition graph*. The edge that we choose for the attack is called a *chaining difference path*. This edge is described by the 16-bit compact representation of the disturbance vector. In the case of a single-block attack the chaining difference transition path with the highest probability is represented by $\mathcal{D} = \texttt{0880}$.

3.5. *A Characteristic of a Single-Block Attack of SHA-0*

Given the disturbance vector $\mathcal{D} = \texttt{0880}$ we construct the characteristic of the attack in accordance with the procedure in Sect. 3.2.1. The characteristic we obtain is given in Table 8. At the top of this table we give the initial chaining difference Ω_P, and at the bottom the final chaining difference Ω_T. The first column (*R.*) is the round index, and in each row i we list the disturbance vector $\mathcal{D}_i$, the expanded message difference W'_i, the expected difference A'_i of the reduced state vector, and the probability to obtain A'_i. We note that the column Ω_A equals the disturbance vector shifted by one, i.e., $\Omega_A = \mathrm{SR}^1(\mathcal{D})$, the initial difference is given by $\Omega_P = (A'_0, A'_{-1}, A'^{\lll 30}_{-2}, A'^{\lll 30}_{-3}, A'^{\lll 30}_{-4}) =$ ***00000***, and the final difference is given by $\Omega_T = (A'_{80}, A'_{79}, A'^{\lll 30}_{78}, A'^{\lll 30}_{77}, A'^{\lll 30}_{76}) =$ ***00000***.

3.6. *Constructing a Pair for the Attack*

The conditions that a pair and its associated state vectors should satisfy are described in Tables 4, 5, and 6. These conditions are sorted into two types: *message-bit conditions* and *state-bit conditions*. Message-bit conditions depend only on the expanded message bits, while state-bit conditions depend also on the reduced state vector bits. A pair M_1, M_1^* is selected such that the message-bit conditions are satisfied for all of the 80 rounds. The message-bit conditions are summarized as follows:

1. For any disturbance in round $i \in \{0, \ldots, 74\}$

$$\mathbf{W^1_i} = \overline{\mathbf{W^6_{i+1}}}.$$

Table 8. A characteristic of a single-block attack on SHA-0.

$\Omega_P = (0, 0, 0, 0, 0) = \boldsymbol{00000}$																			
R.	$\mathcal{D}$	Ω_W	Ω_A	p	R.	$\mathcal{D}$	Ω_W	Ω_A	p	R.	$\mathcal{D}$	Ω_W	Ω_A	p	R.	$\mathcal{D}$	Ω_W	Ω_A	p
−5	0				17	2	80000042	2	1/4	39	0	80000002	0	1/2	61	0	00000002	0	1/2
−4	0		0		18	2	80000040	2	1/8	40	0	00000000	0	1/2	62	2	00000002	0	1/2
−3	0		0		19	2	00000040	2	1/8	41	2	80000002	0	1/2	63	2	00000042	2	1/2
−2	0		0		20	0	00000042	2	1/16	42	0	80000040	2	1/4	64	0	80000042	2	1/2
−1	0		0		21	2	80000000	0	1/4	43	0	00000002	0		65	0	80000002	0	1/2
0	0	00000000	0		22	2	80000042	2	1/2	44	0	80000000	0	1/2	66	0	00000000	0	1/2
1	0	00000000	0		23	0	00000042	2	1/4	45	2	80000002	0	1/2	67	0	00000000	0	
2	2	00000002	0		24	0	00000002	0	1/2	46	0	80000040	2	1/4	68	2	80000002	0	
3	0	00000040	2	1/2	25	0	00000000	0	1/2	47	0	00000002	0		69	2	00000042	2	1/2
4	0	00000002	0		26	2	00000002	0		48	2	80000002	0	1/2	70	2	00000040	2	1/2
5	0	80000000	0	1/4	27	2	80000042	2	1/2	49	0	80000040	2	1/4	71	2	80000040	2	1/2
6	2	80000002	0	1/2	28	2	00000040	2	1/2	50	0	80000002	0	1/2	72	2	00000040	2	1/4
7	0	80000040	2	1/4	29	0	80000042	2	1/2	51	2	80000002	0	1/2	73	0	80000042	2	1/4
8	0	00000002	0		30	0	00000002	0	1/4	52	0	80000040	2	1/4	74	0	80000002	0	1/4
9	0	80000000	0	1/4	31	0	80000000	0	1/2	53	0	80000002	0	1/2	75	0	80000000	0	1/2
10	0	80000000	0	1/2	32	0	00000000	0		54	2	80000002	0	1/2	76	0	00000000	0	
11	0	80000000	0	1/2	33	0	80000000	0		55	2	80000042	2	1/4	77	0	80000000	0	
12	0	00000000	0		34	0	00000000	0		56	2	80000040	2	1/4	78	0	00000000	0	
13	0	00000000	0		35	2	00000002	0		57	0	80000042	2	1/4	79	0	00000000	0	
14	2	00000002	0		36	0	00000040	2	1/2	58	2	00000000	0	1/4	80			0	
15	0	00000040	2	1/2	37	2	00000000	0		59	2	80000042	2	1/4					
16	2	00000000	0		38	0	80000040	2	1/2	60	0	00000042	2	1/2					
															$\Omega_T = (0, 0, 0, 0, 0) = \boldsymbol{00000}$				

2. For any disturbance in rounds $i \in \{38, \ldots, 57\}$

$$\mathbf{W_i^1} = \overline{\mathbf{W_{i+2}^1}}.$$

3. For any two consecutive disturbances in rounds i and $i+1$ where $i \in \{36, \ldots, 55\}$

$$\mathbf{W_i^1} = \overline{\mathbf{W_{i+1}^1}}.$$

4. For any two disturbances distant by two rounds, in rounds i and $i+2$ where $i \in \{36, \ldots, 55\}$

$$\mathbf{W_i^1} = \overline{\mathbf{W_{i+4}^1}}.$$

We note that all the conditions are on bit 1 and 6, and that there might be some additional conditions for specific disturbance vectors.

The conditions are written as equations of the variables $\mathbf{W_i^1}$'s, and $\mathbf{W_i^6}$'s, $i \in \{0, \ldots, 79\}$. The expansion equation (1) is then used to write the $\mathbf{W_i^1}$'s and $\mathbf{W_i^6}$'s $i \in \{16, \ldots, 79\}$ as functions of $\mathbf{W_i^1}$'s and $\mathbf{W_i^6}$'s $i \in \{0, \ldots, 15\}$. The obtained set of equations are then solved by a Gauss elimination (or any other technique) for all the $\mathbf{W_i^1}$'s and $\mathbf{W_i^6}$'s, and then the attacker selects a solution from the solution space. Each solution determines all the values of bit 1 and 6 of the expanded pair.

Once the message-bit conditions are solved and bit 1 and 6 are fixed, the attacker chooses the other message bits W_i^j, $i \in \{0, \ldots, 15\}$, $j \in \{0, \ldots, 31\} \backslash \{1, 6\}$ such that the state-bit conditions of Tables 4, 5, and 6 are satisfied. Even though these conditions depend in a non-linear way on the bits of the message, the state-bit conditions up to Round 15 can easily be satisfied by an appropriate selection of message bits. Once they are solved, the first 15 words are fixed, and many values of W_{15} are tested. When all (or a selected part) of the possible values of W_{15} are tested, the pair is replaced, and the above procedure is repeated. We note that conformance up to Round 18 depends on state bits of Round 15 and earlier rounds. Thus, most of the state-bit conditions are corrected up to Round 18 with a negligible complexity.

3.7. *Complexity Evaluation*

As we have already indicated in Sect. 3.6, an attacker may choose message bits such that a pair conforms to the characteristic up to Round 18 with a negligible complexity. Conformance to higher rounds is considered uncontrollable. Thus, in the evaluation of the probability that a chosen pair conforms to 80 rounds, we count the conditions of Tables 4, 5, and 6 from Round 18 up to Round 79, and assume that each condition contributes a factor `1/2` to the probability.

The complexity is measured by the number of SHA-0 calls. Hence, using an early termination, when a non-conformance is detected, a factor 4 is saved, i.e., for each chosen pair the execution of the compression function is terminated after 20 rounds on average.

Under these assumptions, the complexity of the attack of [10] (that uses the characteristic described in Table 8) is 2^{58} SHA-0 calls. We note that in [10] the announced

complexity is 2^{61} pairs of messages that an attacker needs to try. However, in an efficient implementation, an attacker selects a message and modify it such that it conforms to the first 15 rounds. He then modifies W_{15} many times ($\approx 2^{13}$ times), and reuses the values of $A_1, \ldots, A_{15}$. Hence, each computation starts at Round 15, and it is terminated when a non-conformance is detected (on average at Round 20). Thus, the complexity of testing a pair is equivalent to 1/8 SHA-0 call, and the overall complexity is about 2^{58} SHA-0 calls.

4. The Neutral Bits Technique

The neutral bits technique improves the complexity of the attack of Sect. 3. Using the neutral bits technique, an attacker gets the first 22 (or more) rounds for free (instead of 18 as in Sect. 3.6). Thus, the complexity of the attack that uses this technique depends only on the conditions of rounds 22–79.

Let δ_{b_i} be a string of 512 bits in which the bit with index b_i set to "1" and all others are set to "0". The basic idea of the neutral bits technique is to generate a pair M_k, $M_k \oplus \Omega_M$ that conforms to a characteristic Ω up to some threshold round R (e.g., $R = 22$), and that has a set $B = \{b_1, b_2, \ldots\}$ of bits (i.e., bit indices) with the following property: For each $b_i \in B$ a simultaneous complementation this bit b_i in both messages M_k and $M_k \oplus \Omega_M$, the obtained pair $M_k \oplus \delta_{b_i}$, $M_k \oplus \Omega_M \oplus \delta_{b_i}$ conforms to *at least* the same number of rounds. We call the bits that have this property *neutral bits*, and use these bits to generate many pairs that conform to at least R rounds with a negligible complexity. Thus, effectively we start the attack from Round R.

In the remainder of this section we give a formal definition of neutral bits, and a description of how to find and use them.

Definition 4 Let Ω be a characteristic, R a threshold round number, and M_k, $M_k \oplus \Omega_M$ a pair of message blocks that conforms to R rounds. A message bit W_i^j is a *(single) neutral bit* with respect to M_k, Ω, and R, if the pair obtained by a complementation of W_i^j in M_k as well as in $M_k \oplus \Omega_M$ also conforms to (at least) R rounds.

The above definition is concerned with a single bit difference. We now generalize the definition of neutral bits to any larger set of bits.

Definition 5 Let Ω be a characteristic, R a threshold round number, and M_k, $M_k \oplus \Omega_M$ a pair that conforms to R rounds. We say that a subset of bits $b_i \subseteq \{0, \ldots, 511\}$ is a *composite neutral bit*[4] with respect to M_k, Ω, and R, if the pair obtained by a simultaneous complementation of all the bits in b_i in M_k and $M_k \oplus \Omega_M$ also conforms to (at least) R rounds.

We note that a composite neutral bit may consist of subsets of message bits that are composite neutral bits by themselves. E.g., let $\{b_i, b_j, b_k\}$ be a composite neutral bit,

[4] In [3] we used the term *simultaneous neutral set*. The term is replaced to convey both the ideas that it is one neutral bit and that it is made of several bit positions.

then $\{b_i, b_j\}$ may also be a composite neutral bit and b_k may or may not be neutral. In the remainder of this paper we use the term neutral bits for composite neutral bit as well as for single neutral bits.

Once we have a set $B = \{b_1, \ldots, b_{|B|}\}$ of neutral bits with respect to M_k, Ω, and R, we are able to construct with each b_i a conforming pair, i.e., we can generate $|B|$ conforming pairs. We are now interested in constructing a much larger number of neutral bits with a set B that has a compact representation in memory, i.e., the number of pairs that can be generated from M_k, and B is not linear with the size of B. To describe how we enlarge the set, and how it is represented in memory, we first define a *neutral pair* and *2-neutral set* with respect to M_k, Ω and R.

Definition 6 A pair of neutral bits b_i and b_j is a *neutral pair* with respect to M_k, Ω, and R, if M_k, and each of the three message blocks obtained by complementing the bits b_i, b_j, or both b_i and b_j conform to (at least) R rounds.

Definition 7 A set $B = \{b_1, \ldots, b_{|B|}\}$ of neutral bits is a *2-neutral-set* with respect to M_k, Ω, and R, if every pair of neutral bits in B is a *neutral pair* with respect to M_k, Ω, and R, and the b_i's are disjoint, i.e., composite neutral bits in B are disjoint.

The following experimental observation was made with data gathered in collision attacks of SHA-0. It shows how we extend our original set of neutral bits, and how we represent them in memory.

Observation 1 Given a *2-neutral-set* $B = \{b_1, \ldots, b_{|B|}\}$ with respect to a message block M_k, Ω, and R, a large fraction of the $2^{|B|}$ different pairs which are obtained by complementing the bits of each possible combination of the neutral bits of B, conforms to (at least) R rounds.

A conclusion from Observation 1 is that conforming pairs may be represented in memory by a logarithmic factor of the number of pairs, e.g., M_k, Ω, and $B = \{b_1, \ldots, b_{|B|}\}$ may represent about $2^{|B|-3}$ pairs that conform to R rounds.

In subsequent sections we refer to versions of SHA-0 with extended and reduced number of rounds. The version of 82 rounds is particularly convenient, and we use it in the following example.

Example 1 A disturbance vector for 82-round extended SHA-0 is given in Table 9. For this disturbance vector the pair M_1, M_1^* given in Table 10 has many neutral bits. A 2-neutral set of this pair with respect to 22 rounds is given in Table 11 including single neutral bits ($|b_i| = 1$), pairs ($|b_i| = 2$), triplets ($|b_i| = 3$), quadruplets ($|b_i| = 4$), and quintuplets ($|b_i| = 5$),each pair of neutral bits in this table is a neutral pair, and the neutral bits are disjoint subsets of the 512 message bits. The size of the 2-neutral set

Table 9. A disturbance vector to attack SHA-0 extended to 82 rounds.

$\mathcal{D}$:	$-5, \ldots, 39$:	***00000***	*0001000010*	*1*001000111	1001011000	0011100000
	$40, \ldots, 81$:		0000001100	0000110110	0000011000	1011011**000** **00**

Table 10. The pair M_1, M_1^* of Example 1.

M_1	19EF75A8 D2F24D9A 8F179A7D 1A295690 2E84C143 D74B9DDC 18C10577 8107056E
	5B1A47ED 6212C3F2 3B2D04F8 F5581AB0 26D8CDBC AB3A3248 F347E871 46278F39
M_1^*	19EF75A8 D2F24D9A 8F179A7D 1A295692 2E84C103 D74B9DDE 98C10577 0107056E
	DB1A47EF 6212C3B2 3B2D04F8 75581AF0 A6D8CDBE AB3A324A 7347E831 C6278F3B

Table 11. The 2-neutral-set of Example 1.

Singles:	$W_{12}^4, W_{14}^9, W_{14}^{10}, W_{14}^{11}, W_{14}^{16}, W_{15}^4, W_{15}^5, W_{15}^9, W_{15}^{10}, W_{15}^{11}, W_{15}^{14}, W_{15}^{15}, W_{15}^{16}, W_{15}^{19}, W_{15}^{21}, W_{15}^{26}, W_{15}^{27}$
Pairs:	$(W_9^{13}, W_8^8), (W_{14}^{13}, W_{13}^8), (W_{15}^{13}, W_{14}^8), (W_{15}^{17}, W_{14}^{12}), (W_{15}^{20}, W_{14}^{15}), (W_{15}^{22}, W_{13}^{12})$
Triplets:	$(W_9^8, W_5^{15}, W_4^{10}), (W_{10}^{21}, W_6^{28}, W_5^{23}), (W_{11}^{24}, W_7^{31}, W_6^{26}), (W_{12}^2, W_8^9, W_7^4), (W_{12}^7, W_8^{14}, W_7^9),$
	$(W_{14}^{14}, W_{13}^{10}, W_{13}^9), (W_{14}^{18}, W_{13}^{13}, W_{12}^9), (W_{15}^8, W_{15}^3, W_{14}^{30}), (W_{15}^{12}, W_{10}^{14}, W_9^9)$
Quadru-	$(W_7^5, W_4^9, W_3^{12}, W_2^7), (W_{10}^{11}, W_6^{18}, W_3^{20}, W_2^{15}), (W_{11}^{12}, W_{10}^{18}, W_{10}^{17}, W_9^{12})$
plets:	$(W_{14}^7, W_{13}^{19}, W_{13}^{18}, W_{12}^{16}), (W_{15}^{25}, W_{13}^{21}, W_{13}^{15}, W_{12}^{16})$
Quintu-	$(W_{14}^{23}, W_{14}^{22}, W_{14}^{21}, W_{13}^{17}, W_{12}^{11}), (W_{15}^7, W_{14}^{17}, W_{10}^{24}, W_{10}^{23}, W_9^{18}),$
plets:	$(W_{15}^{24}, W_{15}^0, W_{14}^3, W_{13}^{22}, W_{13}^4)$

is 40, thus there are 2^{40} possible compositions of neutral bits, from which about 2^{37} (a fraction of $1/8$) are neutral. Moreover,

1. A composition of 10 single neutral bits leads to a conformance of 49 rounds.
2. A composition of four single neutral bits and the first quadruplet leads to a conformance of 54 rounds. Thus, in less than 2^{18} complexity we can find conformance to 54 rounds.
3. One of the 2^{23} compositions of singles and pairs from Table 11 leads to conformance of 58 rounds.

4.1. *A Collision Attack Using a 2-Neutral Set*

Given a 2-neutral set with respect to M_k, Ω, and R, we generate a new pair $M_k \oplus \delta$, $M_k \oplus \Omega_M \oplus \delta$, where δ is a composition of neutral bits from the set. For each different composition of the neutral bits, we obtain a different pair which with high probability conforms to (at least) R rounds. If the set is of size n, then 2^n different pairs can be generated by the set, of which a fraction of about $1/8$ conforms to R or more rounds. These pairs are used for the attack. The probability of an attack that uses these pairs is then:

$$p(s'_{80} = \Omega_{S_{80}} | s'_R = \Omega_{S_R}) = \prod_{t=R}^{79} p_t,$$

where p_t is the probability of Round t. Hence, the probabilistic analysis starts from round R. Using this technique in the attack on SHA-0, the attack complexity is reduced from 2^{60} to 2^{54} SHA-0 calls, and a more careful analysis shows that the complexity is reduced to 2^{48} SHA-0 calls. We note that many neutral bits are in W_{15}, thus, for most tested messages generated in the attack $A_1, \ldots, A_{15}$ are equal. Therefore, an efficient implementation computes these values once and reuses them many times afterward. The computation of each analyzed pair starts at Round 15 (instead of Round 0), and it is terminated when a non-conformance is detected. On average, a non-conformance is detected at Round 22. Hence, the complexity of testing each pair equals to a computation of 8 rounds, i.e., from Round 15 to Round 22. Further speed-up is made by modifying and testing Round 22, and only if this test passes a re-computation and testing of Rounds $15, \ldots, 22$ is made. In terms of SHA-0 calls, the complexity of such technique is about $1/8$ SHA-0 call for each pair.

A collision search algorithm that uses neutral bits consists of two phases. It starts with the generation of a 2-neutral set B, and continues with an exhaustive search for all the candidate pairs derived from the set. If the search ends and a collision is not found, then a new message is randomly selected, and the process is repeated.

The size of the set decreases with the round threshold R for which the set is generated. We use the notation $k(R)$ for the size of B with threshold R, i.e., $B = \{b_1, \ldots, b_{k(R)}\}$. The round number R for which we generate the set is selected to be the maximal round such that the complexity of finding a 2-neutral set of size $k(R)$ is less than or equal to the complexity of exhaustively testing the $2^{k(R)}$ messages for collision. In SHA-0, typical values for R are in the range 22–24.

The following observation shows that when the attack is performed using neutral bits, the probabilistic analysis of rounds greater than R, is better than expected:

Observation 2 Let R' and R ($R' > R$) be some rounds, and $k(R)$ the size of a 2-neutral set with respect to M_k, Ω, and R, such that $p(R \rightarrow R') = \prod_{t=R}^{R'-1} p_t \approx 2^{-k(R)}$. By generating the $2^{k(R)}$ pairs, we obtain few pairs that conform to $R' + l$, which we would expect to get with a larger set of neutral bits of a size $k(R) + \alpha$, where $2 \leq l \leq 4$ and $3 \leq \alpha \leq 8$.

4.2. *Finding a 2-Neutral Set*

In this section we describe an algorithm that finds a 2-neutral set. This algorithm chooses a random message M_k and modifies it such that the pair $M_k, M_k \oplus \Omega_M$ conforms to R rounds, and it has a 2-neutral set of size $k(R)$. The pair is optimized such that the size $k(R)$ of the set is as large as possible, and the set is neutral with respect to the highest possible round R. We note that since the time complexity of this algorithm is amortized over $2^{k(R)}$ pairs, it does not have to be negligible. We also do not claim that this algorithm is optimal.

4.2.1. *Finding Neutral Bits and Optimizing a Pair*

Given a randomly chosen pair $M_k, M_k \oplus \Omega_M$, we start to modify it one round at a time, to conform to as many rounds of the characteristic as possible. The modifications up to

Round 15 are performed by direct complementation of the message bits as mentioned in Sect. 3. Once the pair conforms to some intermediate round r, where $16 \leq r < R$, we find and count the neutral bits with respect to r. We then modify the pair such that its conformance to the first r rounds is not affected, and with the aim of increasing the number of neutral bits. When we cannot increase this number any further, we proceed to the next round. The process ends when we get to round R, which is typically between 22 to 24.

A search for a single neutral bit for a given pair is performed simply by complementing a bit in both messages of the pair and test whether conformance is not affected. Once it is found, it is added to the set B. This search covers the 512 bits of the message, excluding bit locations 1 and 6 which are set in advance to satisfy the message-bit conditions. The complexity of this search is estimated by the number of tests for neutrality we have to perform. In the case of a search for single neutral bits it is 480 tests ($512 - 2 * 16$). When a search for single neutral bits ends, a search for composite neutral bits of pairs is conducted. This search is performed similarly to the search for single neutral bits except that every pair of bits is complemented and tested for neutrality, and the search covers the 512 bits of the message to exclude bit 1 and 6, and the bits in B. On average, the size of B when this search starts is about 70, thus the number of pairs we test for neutrality is about $\binom{410}{2} = 83{,}845$. Similarly, we find larger composite neutral bits until we identify as many neutral bits as possible.

The modifications of the pair to extend the set are performed by trying to complement two or more bits of a local collision sequence W_i^j, W_{i+1}^{j+5}, W_{i+2}^j, W_{i+3}^{j+30}, W_{i+4}^{j+30}, W_{i+5}^{j+30}, in both messages of a pair. These modifications are repeated where the index j is advanced by one each time, until all j's are covered. Each time we modify the pair, we count the number of neutral bits. If the number is increased by the modification, then we replace the original pair with the new one and start the procedure of modifying and counting all over. We end up with a pair that conforms to R rounds and has the largest number of neutral bits that we can find.

4.2.2. *Finding Neutral Pairs and 2-Neutral Sets*

Given a pair with a set of neutral bits, we are interested in finding the largest 2-neutral set. We first identify neutral pairs by a simultaneous complementation of each pair of neutral bits in the set, and by testing whether the conformance to R rounds is not affected. We then represent each neutral bit as a vertex in a graph, and add an edge for each pair of vertices that corresponds to a neutral pair. The largest 2-neutral set corresponds to the largest clique in the 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 the general problem of 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. Therefore, we use simple heuristics that produce a very good approximation to the maximal clique: We initialize the 2-neutral set to be an empty set. We then generate a sorted list of vertices which represent single neutral bits, by the number of edges that are connected to each vertex. The first vertex in the sorted list is added to the 2-neutral set, and the vertices that are not connected to the vertex we have

Table 12. Complexities of attacks on reduced and extended versions of SHA-0 (further results are given in Table 1).

# Rounds	Complexity	# Rounds	Complexity	# Rounds	Complexity
50	–(*)	75	2^{52}	82	2^{43}
64	2^{29}	76	–(*)	83	2^{65}
65	2^{29}	77	2^{66}	84	2^{64}
68	2^{43}	78	2^{56}	85	2^{71}
72	2^{50}	79	2^{56}	86	2^{72}
73	2^{50}	80	2^{56}	87	–(*)
74	2^{50}	81	2^{43}	92	2^{74}

* There is no disturbance vector that predicts collisions after 50, 76, or 87 rounds (before the resolution of consecutive disturbances in the IF rounds)

just added are removed from the graph. The procedure is repeated with the remaining graph until the graph is empty. Once the graph is empty, the 2-neutral set contains a clique of single neutral bits. Our next step is to add larger composite neutral bits. We start by searching for neutral sets of size two (pairs) among the bits which are not in the 2-neutral set B we have just constructed. From the neutral sets we find, we remove those which are not neutral pairs with each neutral bit in B. With the remaining neutral sets we build a new graph and search the maximal clique as before. The procedure is then repeated with neutral sets of size three (triplet), size four (quadruplet), and as many other sets we can find.[5] The result of such an algorithm is shown in Example 1 and Tables 10 and 11.

5. The Complexity is not Monotonous with the Number of Rounds

The attacks of Sections 3 and 4 are also applicable to SHA-0 with fewer or more than 80 rounds, i.e., for the attack of SHA with R rounds, we use the method of Sect. 3.2 to find a disturbance vector with the least number of disturbances from Round 22 up to Round R and we use it to construct a characteristic. Consulting Tables 7 and 9 we observed that the disturbance vector of the full 80-round SHA-0 has more disturbances than that of the 82-round (22 vs. 19 respectfully). That is to say, we unexpectedly found that the complexity of attacking a version with more rounds is lower than that with fewer rounds.

Table 12 summarizes the complexities of attacking SHA with different number of rounds. From this table we see that there are few reduced and extended versions with no disturbance vector that predicts collisions: in these cases this attack cannot be used. We also see that the complexity of collision attacks on the different versions is not monotonous with the number of rounds, e.g., extended versions with 81 and 82 rounds are much easier to attack than 80-round SHA-0.

The phenomenon of non-monotonic complexities with the number of rounds is a result of the expansion of the disturbance vector by the LFSR. The number of disturbances that

[5] The search is terminated when the effort of finding a larger set exceeds the effort of testing all the candidates of the set.

the LFSR produces depends on its initialization, and it does not necessarily grow with R. Hence, the probability associated with each disturbance vector does not necessarily decrease with the number of rounds.

A prior case of attack on an extended version of a hash function that has a smaller complexity than the attack on the full version appeared in the differential cryptanalysis of N-Hash [5,27]. N-Hash has eight rounds, while the attack can find collisions for versions with 3, 6, 9, and 12 rounds. In particular, a collision of a 9-round extended version can be found faster than a collision of the full 8-round version.

This non-monotonicity occurs because collision attacks require that the characteristic prediction of the output difference will be zero. For comparison, in block ciphers, a reduction of the characteristic to a smaller number of rounds necessarily results in another characteristic with a higher probability. In hash functions, the reduced characteristic has higher probability as well, but it does not have the zero difference as required for a collision. Thus, unlike in the case of block ciphers, the probabilities of the best characteristics are not monotonically decreasing with the number of rounds.

6. The Multi-Block Tool

Most hash functions currently in use are based on the Merkle–Damgård construction which is proved to be collision resistant if the underlying compression function is collision resistant. In this section we investigate the strength of the construction in cases the compression function is not collision resistant.

In a single-block attack an attacker uses the standard initial value and tries to find two message blocks that hash to the same value, i.e., $\Omega_P = \Omega_T = 0$, and $\Omega_M \neq 0$. Though the domain of the input of the compression function consists of the message $\{0, 1\}^b$ and chaining value $\{0, 1\}^{m_c}$ domains, only the message domain is analyzed. As we show in Sect. 3, this analysis suffices to break SHA-0. However, there are variants of SHA-0 (see Table 12) in which this type of attack is not applicable. Hence, an extension of the analysis to the message and chaining value domains $\{0, 1\}^{b+m_c}$ is natural.

Given characteristics with $\Omega_P \neq 0, \Omega_M \neq 0$, and $\Omega_T = 0$, an attack consists of two steps. Firstly, characteristics that lead to the required $\Omega_P \neq 0$ are used. Secondly, the characteristic $\Omega_P \neq 0, \Omega_M \neq 0$, and $\Omega_T = 0$ is used to find a collision. The iterative structure of the Merkle–Damgård construction enables such concatenations of characteristics.

The multi-block technique allows for differences in both inputs of the compression function and also in the output of the compression function. This extension allows us to use characteristics with higher probabilities than previously used. Each characteristic by itself (except the last one) does not lead to a collision but to a near-collision, and the last characteristic leads to a collision. The technique shows how characteristics that predict near-collisions are found and concatenated to a longer characteristic that predicts a collision. In the next subsections we present the technique with examples that emphasize different aspects and advantages. We note that although our examples are all on the SHA family of hash functions, the multi-block technique is applicable to any differential attack on hash functions that follow the Merkle–Damgård (or any iterative) construction.

As a first and simple example we show a case where several blocks are used for an attack, which is useful when the initial state of the compression function is incompatible with the characteristic. The general and more interesting cases are described afterward.

6.1. *Solving Initial State Incompatibility by an Additional Block*

In some cases the initial state is incompatible with the characteristic. As an example consider:

$$A_1^0 = W_0^0 \oplus A_0^{27} \oplus \mathrm{IF}(A_{-1}^0, A_{-2}^2, A_{-3}^2) \oplus A_{-4}^2 \oplus K_0^0.$$

With h_0 the standard initial values, $A_0^{27} = 0$, $A_{-1}^0 = 1$, $A_{-2}^2 = 0$, $A_{-3}^2 = 0$, $A_{-4}^2 = 0$, and $K_0^0 = 1$, thus:

$$A_1^0 = W_0^0 \oplus 0 \oplus \mathrm{IF}(1, 0, 0) \oplus 0 \oplus 1 = \overline{W_0^0}.$$

In this case a disturbance in the first round in bit location 0 always leads to a difference in the carry from bit 0 to bit 1, which contradicts the conditions of the characteristic.

Another incompatibility may occur by the f_i function. Consider a disturbance in bit j of the first message word, $\mathbf{W_0^j}$, where $A_0^{j-30} = A_{-1}^{j-30}$. In this case, $\mathrm{IF}_2^j(\mathbf{A_1^j}, A_0^{j-30}, A_{-1}^{j-30})$ is not active, so the correction $\mathbf{W_2^j}$ always fails. In case the initial state equals the standard initial value, this incompatibility occurs in 24 out of the 32 bit locations (except for locations 1, 5, 9, 13, 17, 21, 25, and 29).

A straightforward solution for this incompatibility is to replace the initial state with a compatible one. In the case of a first block where the initial state equals the standard initial values, a first arbitrary message block (with zero difference) is added in both runs to obtain a compatible initial state. The actual attack starts with the second block using the compatible h_1 instead of the incompatible h_0. The result is a two-block attack with no difference in the first block but with a difference in the second block.

6.2. *Two-block Collisions*

In this section we describe a technique to find two-block collisions. In a two-block attack, the first pair starts with the same initial value h_0 (which is usually the standard initial value but may be different for reasons mentioned in the previous section). The pair M_1, M_1^* is found such that $h_1 = C(M_1, IV)$, $h_1^* = C(M_1^*, IV)$, and $h_1 \approx h_1^*$, where the operator $\approx$ denotes a small Hamming distance between these two values. The pair M_1, M_1^* and chaining variables $h_0 = IV$, h_1, and h_1^* form a *near-collision*. A second pair of messages M_2, M_2^* is now found such that $h_2 = C(M_2, h_1) = C(M_2^*, h_1^*) = h_2^*$. This second pair starts from the chaining variables with the small difference of the previous pair, and ends with a collision. The second pair M_2, M_2^* with the chaining variables h_1, h_1^*, and h_2 form a *pseudo-collision*. An illustration of near-collision, pseudo-collision, and a two-block collision is given in Fig. 7. Such an attack is useful since in many cases finding near-collisions and pseudo-collisions is much easier than finding a single-block collision. It is also useful when there is no disturbance vector that predicts a single-block collision.

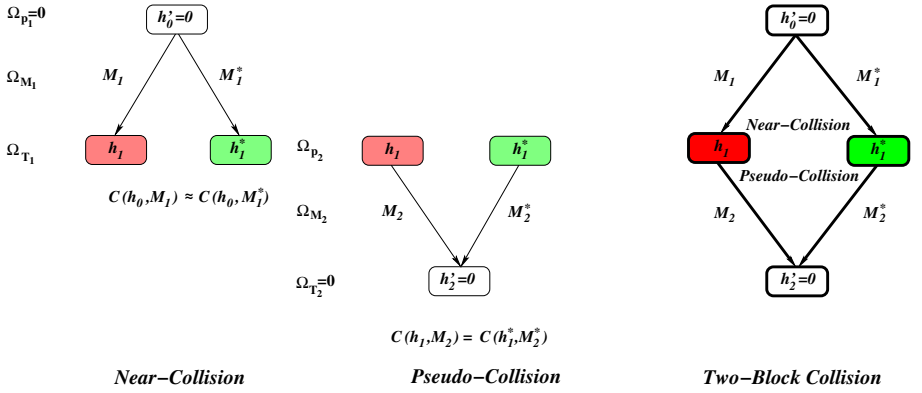

Fig. 7. Using intermediate near-collisions to find collisions with two blocks.

In order to apply such an attack an attacker searches for a pair of disturbance vectors from which he constructs the characteristics Ω^1, Ω^2. The disturbance vectors are selected such that the differences of the characteristics are: $\Omega^1_P = 0, \Omega^2_T = 0$, and $\Omega^1_T = \Omega^2_P$, and the least probability of both characteristics $(\min(p^1_\Omega, p^2_\Omega))$ is maximal. After the construction of the characteristics the attacker searches for right pairs. The search starts with the aim of finding a right pair for Ω^1 with the initial value h_0. Once it is found, h_1 and h^*_1 of the first right pair are used in the search of a right pair for Ω^2. The result forms a two-block collision.

6.3. *Characteristics For a Two-Block Attack*

Given the disturbance vectors $\mathcal{D}_1$ and $\mathcal{D}_2$, we proceed with the construction of the characteristic of each block. Constructing the characteristics of a two-block attack is somewhat different than for a single-block attack, since the requirements imposed by the concatenation of the first and second pairs should be considered. In a single-block attack $\Omega_T = \boldsymbol{00000}$ if and only if

$$(\Omega_{A_{R'}}, \Omega_{A_{R'-1}}, \Omega_{A_{R'-2}}, \Omega_{A_{R'-3}}, \Omega_{A_{R'-4}}) = \boldsymbol{00000}. \tag{9}$$

In a two-block attack we set conditions to obtain Ω_T. We denote the final chaining value of block k by h_k, and set conditions such that the difference $h_k \oplus h^*_k$, equals the difference defined by Ω^k_T where

$$\begin{aligned} h_{k,i} = A_{R'-i} + A_{-i} = A^{\lll 5}_{R'-1-i} + \mathrm{f}_{R'-1-i}(A_{R'-2-i}, A^{\lll 30}_{R'-3-i}, A^{\lll 30}_{R'-4-i}) + \\ A^{\lll 30}_{R'-5-i} + W_{R'-1-i} + K_{R'-1-i} + A_{-i}, \quad i = 0, \ldots, 4, \quad k = 1, 2. \end{aligned} \tag{10}$$

In addition we should consider the compatibility of the initial state with the characteristic. In the first block this requirement may be satisfied using the technique of Sect. 6.1, which turns the attack into a three-block attack with the same first block in both runs. Incompatibility of initial values with characteristics other than the first characteristic

cannot be resolved by the technique of adding an identical block to both runs. Hence, conditions on the final chaining values h_1, h_1^* (or h_2, h_2^* in case a first block is added) may be added in order to ensure that the chaining values are compatible with the characteristic of the next block. These additional conditions contribute to the probability of the previous characteristic, thus increasing the complexity of finding the near-collision in that block.

6.4. *Complexity Evaluation*

The complexity of the attack is measured by the number of SHA-0 calls. It is computed for each block by multiplying the complexity of generating a pair that conforms to at least R rounds, by the number of pairs we need to generate in order to find a right pair for each characteristic.

As shown in Sect. 4.1, the complexity of generating a pair that conforms to at least $R = 22$ rounds is about 0.25 SHA-0 call. The number of pairs we need to generate in the attack depends on the probabilities of obtaining a pair that satisfies the requirements we have just described. These probabilities are computed by counting the number of conditions that such a pair should satisfy. It is assumed that the conditions are independent, and that each condition is satisfied with probability $1/2$. These probabilities are listed as follows:

1. The probability

$$p_{R\to R'-5} = p(s'_{R'-5} = \Omega_{S_{R'-5}} | s'_R = \Omega_{S_R}, s'_0 = \Omega_P, M' = \Omega_M)$$

   that a pair that conforms to at least R rounds also conforms to $R'-5$ rounds (where R' is the number of rounds we attack). In the computation of this probability we count the conditions for conformance from Round R up to Round $R'-5$.
2. The probability

$$p_{R'-5\to\Omega_T} = p(h'_i = \Omega_T | s_{R'-5} = \Omega_{S'_{R'-5}}, s'_0 = \Omega_P, M' = \Omega_M)$$

   that a pair that conforms to $R'-5$ rounds has the required final chaining difference Ω_T, where i is the block number we attack.
3. Let $p_{connect}$ be the probability that a pair that conforms to the characteristic of the current block, also has chaining values which are compatible with the characteristic of the next block. $p_{connect}$ is computed by counting the conditions on the initial values of the next block, i.e., that depend on A_i^j's where $i \leq 0$.

Notice that in case the technique of Sect. 6.1 is used to generate an initial value compatible with Ω_P of the first block, the complexity of generating the first block equals $1/p_{connect}$.

6.5. *Collisions with More than Two Blocks*

The two-block attack is generalized to multi-block attack with longer paths of differences. In such an attack a first pair of message blocks M_1, M_1^* is found such that it leads to a near-collision $h_1 \approx h_1^*$. A second pair M_2, M_2^* is then found such that M_2 com-

The first pair creates a near−collision.

The second pair starts with a small difference in the initial value and ends with a near−collision.

Additional pairs are added as necessary to reduce the search complexity.

The last pair is a pseudo−collision.

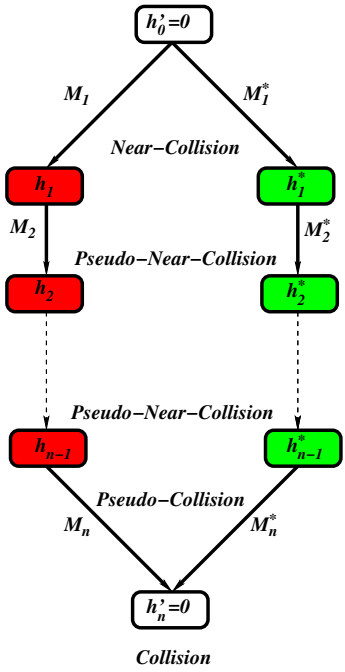

Fig. 8. The multi-block technique—using intermediate near-collisions to find collisions.

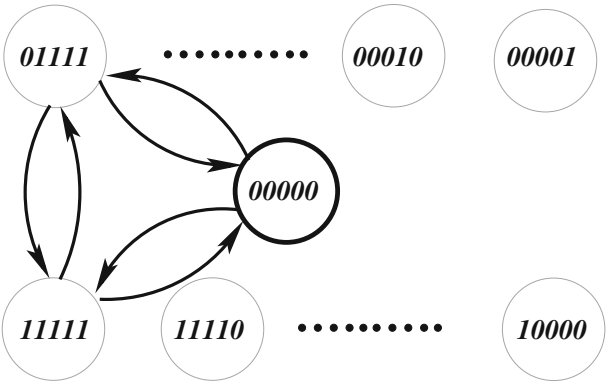

Fig. 9. The modified chaining transition graph.

pressed with h_1, and M_2^* compressed with h_1^* create a second near-collision $h_2 \approx h_2^*$. We call such near-collision to near-collision blocks, *near-pseudo-collision*. After one or more near-pseudo-collision blocks, a final pseudo-collision is applied to form a *multi-block-collision*. An illustration of a multi-block collision is given in Fig. 8. The transition graph contains 32 vertices, as shown in Fig. 9, and each pair of vertices is connected with about 2^6 edges.

We note that in some cases it might be desirable to have the same difference twice, i.e., $h'_i = h'_j, i \neq j$, although it looks like the rest of the stream could be directly computed from the first occurrence of that difference. Such cases may appear when the next block cannot start with the currently found pair of intermediate values, due to compatibility requirements with the next block. Of course in such cases it is possible to search again for another previous block, and get another near-collision with the same difference, which will hopefully solve the problem. But in some cases it may be faster to find a block with equal input and output differences (or a series of blocks that lead to the same difference after several blocks). Thus, the attacker may prefer not to discard the near-collision he found, but to use it as an easy starting point for a more usable one with the same difference.

6.6. *Revisiting Characteristics and Disturbance Vectors*

In the attacks we described so far, we assumed a one-to-one correspondence between disturbance vectors and characteristics (see Sect. 3.2). This assumption is a result of our approximation of the non-linear functions: IF, MAJORITY, and addition modulo 2^{32} by the XOR operation.

In particular, we assumed that a transition between a pair of vertices that represents the chaining differences Ω_P and Ω_T is made by an edge (disturbance vector) $\mathcal{D}$ that satisfies certain relations: $\mathcal{D}$ leaves a vertex which represents a difference Ω_P if

$$\Omega_P = (\mathcal{D}_{-1}, \mathcal{D}_{-2}, \mathcal{D}_{-3}^{\lll 30}, \mathcal{D}_{-4}^{\lll 30}, \mathcal{D}_{-5}^{\lll 30}). \tag{11}$$

Similarly, it enters a vertex that represents a difference Ω_T if

$$\begin{aligned}\Omega_T = ((\mathcal{D}_{-1} \oplus \mathcal{D}_{79}), (\mathcal{D}_{-2} \oplus \mathcal{D}_{78}),\\ (\mathcal{D}_{-3} \oplus \mathcal{D}_{77})^{\lll 30}, (\mathcal{D}_{-4} \oplus \mathcal{D}_{76})^{\lll 30}, (\mathcal{D}_{-5} \oplus \mathcal{D}_{75})^{\lll 30}).\end{aligned} \tag{12}$$

Using these relations we constructed the transition graph of Fig. 9 with about 2^6 edges (disturbance vectors) that connect each pair of vertices.

We now show how characteristics (edges) that do not follow the XOR approximation in the first few rounds are added to the transition graph. In these characteristics $\Omega_P \neq (\mathcal{D}_{-5}, \ldots, \mathcal{D}_{-1})$ but the remainder of the differences of the reduced state vector are still approximated by $\Omega_A = \mathrm{SR}^1(\mathcal{D})$. The updated transition graph includes edges $\mathcal{D}$ that leave vertex Ω_P, enter vertex Ω_T and satisfy:

$$\Omega_T = \Omega_P \oplus (\mathcal{D}_{79}, \mathcal{D}_{78}, \mathcal{D}_{77}^{\lll 30}, \mathcal{D}_{76}^{\lll 30}, \mathcal{D}_{75}^{\lll 30}) = \Omega_P \oplus \Omega_{s_{80}}, \tag{13}$$

where Ω_P is not necessarily the first five entries of $\mathcal{D}$. Note that if these characteristics are used, then a few additional conditions should be added to the first rounds. However, since the overall probability is measured from higher rounds (e.g., Round 22) they are not a factor in the complexity of the attack.

In the following example we demonstrate how a disturbance vector that at a first look seems incompatible is used to connect a pair of vertices in a transition graph.

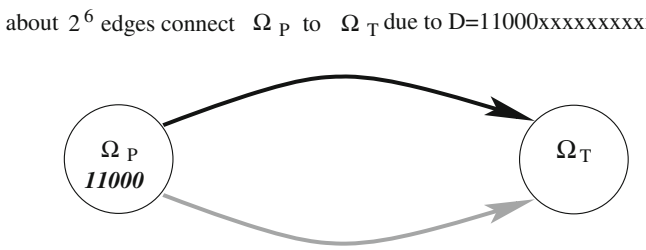

Fig. 10. Matching seemingly incompatible disturbance vectors to a chaining difference.

Example 2 Consider the transition graph in Fig. 10. In this graph, transitions are made from the vertex ***11000*** to Ω_T by compatible disturbance vectors (i.e., that satisfy Eq. (11)), but in addition disturbance vectors that start with ***01000*** (which obviously does not equal the chaining difference of the vertex) are used. The disturbance vector that is used to construct the differences Ω_W between the runs, starts with ***01000*** while the differences between the state vectors start with $\Omega_P = $ ***11000***. With these incompatible Ω_W and Ω_A we show that the difference $\mathbf{A^1_{-4}}$ that has no corrections in Ω_W, does not propagate, i.e.,

$$(A'_{-3}, A'_{-2}, A'_{-1}, A'_0, A'_1, A'_2) = (2, 0, 0, 0, 0, 0).$$

$\mathbf{A^1_{-4}}$ affects the computations only at Round 0; thus if it is compensated at Round 0, then it does not affect successive computations. By setting $A^{31}_{-1} = 1$ (instead of $A^{31}_{-1} = 0$ per Table 5), $\mathrm{IF}^{31}_0(A^{31}_{-1}, A^1_{-2}, \mathbf{A^1_{-3}}) = A^1_{-2}$ is inactive, and its correction $\mathbf{W^{31}_0}$ is used to compensate $\mathbf{A^1_{-4}}$ as required.

Similar techniques to the one shown in Example 2 are used to find more initial chaining differences (Ω_P's), and compatible disturbance vectors. A summary of these compatibilities is given in Table 13. In this table the first column is the compact representation of Ω_P. The next three columns show the corrections applied to bits 1, 6, and 31 of $W'_4, \ldots, W'_0$. In this representation, no correction is denoted by "0", and a correction by "1". The next three columns specify the required differences to correct the disturbances. A "0" denotes an inactive bit, "1" an active bit, and "*" denotes a bit that may be active or inactive. Compatibility is determined by considering each "*" in the required differences $W'_4, \ldots, W'_0$ once as a "0" and then as "1". The result is compared with the corrections of the local collision sequence, and compatibility is determined accordingly. The last column gives all the combinations of $\mathcal{D}^1_{-1}, \ldots, \mathcal{D}^1_{-5}$ which are compatible with Ω_P.

Table 13 is now used (instead of Eq. (11) to draw the edges that leave each vertex, and Eq. (13) is used to compute the vertex to which each edge is entered. In the obtained graph we search for the path with the least complexity as before.

Table 13. Chaining differences and their compatibility with disturbance vectors.

Ω_P	Corrections in $W'_4, \ldots, W'_0$			Required differences in $W'_4, \ldots, W'_0$			Compatible $\mathcal{D}_{-1}, \ldots, \mathcal{D}_{-5}$
$A_0^{1\prime}, \ldots, A_{-4}^{1\prime}$	Bit 1	Bit 6	Bit 31	Bit 1	Bit 6	Bit 31	
00000	00000	00000	00000	00000	00000	00000	***00000***
00001	00000	00000	00001	00000	00000	00001	***00001***
00010	00000	00000	00011	00000	00000	0001*	***00010*** ***00011***
00011	00000	00000	00010	00000	00000	0001*	***00010*** ***00011***
00100	00000	00000	00111	00000	00000	001**	***00100*** ***00101*** ***00110*** ***00111***
00101	00000	00000	00110	00000	00000	001**	***00100*** ***00101*** ***00110*** ***00111***
00110	00000	00000	00100	00000	00000	001*1	***00100*** ***00111***
00111	00000	00000	00101	00000	00000	001*0	***00101*** ***00110***
01000	00001	00000	01110	0000*	00000	01**0	***01000*** ***01011*** ***01101*** ***01110***
01001	00001	00000	01111	0000*	00000	01**1	***01001*** ***01010*** ***01100*** ***01111***
01010	00001	00000	01101	0000*	00000	01***	***01000*** ***01001*** ***01010*** ***01011*** ***01100*** ***01101*** ***01110*** ***01111***
01011	00001	00000	01100	0000*	00000	01***	***01000*** ***01001*** ***01010*** ***01011*** ***01100*** ***01101*** ***01110*** ***01111***
01100	00001	00000	01001	0000*	00000	01*1*	***01000*** ***01001*** ***01110*** ***01111***
01101	00001	00000	01000	0000*	00000	01*1*	***01000*** ***01001*** ***01110*** ***01111***
01110	00001	00000	01010	0000*	00000	01*01	***01010*** ***01100***
01111	00001	00000	01011	0000*	00000	01*00	***01011*** ***01101***
10000	00010	00001	11100	000*0	00001	1**00	***10000*** ***10110***
10001	00010	00001	11101	000*0	00001	1**01	***10001*** ***10111***
10010	00010	00001	11111	000*0	00001	1**1*	***10010*** ***10011*** ***10100*** ***10101***
10011	00010	00001	11110	000*0	00001	1**1*	***10010*** ***10011*** ***10100*** ***10101***
10100	00010	00001	11011	000*0	00001	1****	***10000*** ***10001*** ***10010*** ***10011*** ***10100*** ***10101*** ***10110*** ***10111***
10101	00010	00001	11010	000*0	00001	1****	***10000*** ***10001*** ***10010*** ***10011*** ***10100*** ***10101*** ***10110*** ***10111***
10110	00010	00001	11000	000*0	00001	1***1	***10001*** ***10010*** ***10100*** ***10111***
10111	00010	00001	11001	000*0	00001	1***0	***10000*** ***10011*** ***10101*** ***10110***
11000	00011	00001	10010	000**	00001	1*1*0	***10000*** ***10011*** ***11101*** ***11110***
11001	00011	00001	10011	000**	00001	1*1*1	***10001*** ***10010*** ***11100*** ***11111***
11010	00011	00001	10001	000**	00001	1*1**	***10000*** ***10001*** ***10010*** ***10011*** ***11100*** ***11101*** ***11110*** ***11111***
11011	00011	00001	10000	000**	00001	1*1**	***10000*** ***10001*** ***10010*** ***10011*** ***11100*** ***11101*** ***11110*** ***11111***
11100	00011	00001	10101	000**	00001	1*01*	***10100*** ***10101*** ***11000*** ***11001***
11101	00011	00001	10100	000**	00001	1*01*	***10100*** ***10101*** ***11000*** ***11001***
11110	00011	00001	10110	000**	00001	1*001	***10111*** ***11010***
11111	00011	00001	10111	000**	00001	1*000	***10110*** ***11011***

7. A Four-Block Collision of SHA-0

The application of the technique of Sect. 6.6 to SHA-0 results in the 4-block path outlined in Fig. 11. A complete description of the four disturbance vectors of this path is given in Table 14.

In this path, the first pair of blocks starts with the standard initial value (i.e., $s_0 = s_0^* = IV$ and $\Omega_{P_1} = \boldsymbol{00000}$) and ends with a final chaining difference $\Omega_{T_1} = \Omega_{P_2} = \boldsymbol{00011}$.

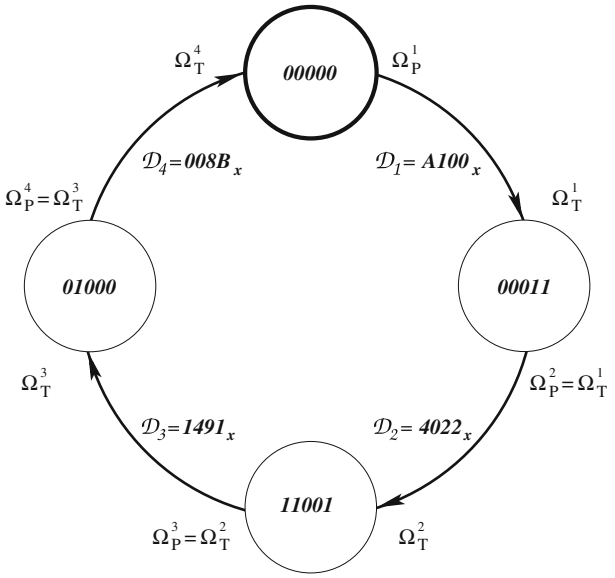

Fig. 11. The four-block chaining differences path of SHA-0.

Table 14. The disturbance vectors of the four-block collision.

$\mathcal{D}_1$	00000	0001000010	1001000111	1001011000	0011100000
		0000001100	0000110110	0000011000	1011011000
$\mathcal{D}_2$	01000	1000000001	0000101001	0001111001	0110000011
		1000000000	0011000000	1101100000	0110001011
$\mathcal{D}_3$	10001	0010010100	0100101111	1100001000	0100001100
		0010110010	0000000001	1101001110	1000010001
$\mathcal{D}_4$	11010	0010000000	0100001010	0100011110	0101100000
		1110000000	0000110000	0011011000	0001100010

The characteristic we use for this transition is constructed using the disturbance vector $\mathcal{D}_1 = \texttt{A100}$. Consulting Table 14, $\mathcal{D}_1$ predicts a final state difference $s'_{80} = \boldsymbol{00011}$, hence using Eq. (13), $\mathcal{D}_1$ is suitable for the first transition. The transition of the second pair of blocks is from a chaining difference $\Omega_{P_2} = \boldsymbol{00011}$, to a final difference $\Omega_{T_2} = \boldsymbol{11001}$ using $\mathcal{D}_2 = \texttt{4022}$. Consulting Table 13, the initial chaining difference $\Omega_{P_2} = \boldsymbol{00011}$ is compatible with $\boldsymbol{00010}$, hence $\mathcal{D}_2 = \texttt{4022}$ is suitable to leave vertex Ω_{P_2}. The predicted final state of $\mathcal{D}_2$ is $s'_{80} = \boldsymbol{11010}$, thus by Eq. (13) the predicted chaining difference of the second block is $\Omega_{T_2} = \Omega_{P_3} = \boldsymbol{11010}$, and $\mathcal{D}_2$ is suitable for the second transition. The transition of the third block is from $\Omega_{P_3} = \boldsymbol{11001}$ to $\Omega_{T_3} = \boldsymbol{01000}$ using $\mathcal{D}_3 = \texttt{1491}$. $\Omega_{P_3} = \boldsymbol{11001}$ is compatible with $\boldsymbol{10001}$, thus $\mathcal{D}_3 = \texttt{1491}$ is suitable to leave vertex Ω_{P_3}. $\mathcal{D}_3$ predicts a final difference $s'_{80} = \boldsymbol{10001}$, hence the predicted chaining difference of the third block is $\Omega_{T_3} = \Omega_{P_4} = \boldsymbol{01000}$, and $\mathcal{D}_3$ is suitable for the third transition. The last transition in this path is made from $\Omega_{P_4} = \boldsymbol{01000}$ to $\Omega_{T_4} = \boldsymbol{00000}$ using $\mathcal{D}_4 = \texttt{008B}$. $\Omega_{P_4} = \boldsymbol{01000}$ is compatible with $\boldsymbol{01011}$, thus $\mathcal{D}_4 = \texttt{008b}$ is suitable

to leave Ω_{P_4}. $\mathcal{D}_4$ predicts a final difference $s'_{80} = \boldsymbol{01000}$, hence $\Omega_{T_4} = \boldsymbol{00000}$, and a collision is predicted after the fourth block.

Using these disturbance vectors we constructed the characteristics of each block, which are given in Tables 15, 16, 17, and 18. In these tables, the locations of inconsistency between a disturbance vector and a reduced state vector are marked with a gray background. The conditions at these locations are set as explained in Sect. 6.6. In the last five rounds we set conditions such that the desired Ω_T is obtained.

We implemented this attack on a highly parallel computer using this path of four characteristics, and a four-block collision was found. The resulting collision is given in Table 19. The chaining variables and differences of this four-block collision of SHA-0 are given in Table 20.

7.1. *Complexity Evaluation*

The complexity evaluation of the four-block attack on SHA-0 is similar to the evaluation of the two-block attack. The contributions of each of the factors $p_{22\to 75}$, $p_{75\to\Omega_T}$, and $p_{connect}$ to the overall probability and to the complexity of the attack are listed in Table 21. In this table we present the probabilities and complexity of each block in two rows. In the first row we present the probabilities to obtain the required differences of the pair of message blocks. In the second row we write the probability of connecting the current block with the next block, and the total probability and complexity of finding a right pair. The overall complexity of the attack under these assumptions is approximately 2^{46} SHA-0 calls.

When measured in terms of the number of pairs to test, about 2^{51} pairs are tested until a collision is found. This number is equivalent to about 2^{49} SHA-0 calls. The actual number (in comparison with the data in Table 21) is due to an analysis that considers conformance to Ω_A in all of the 80 rounds, and then checks whether the correct Ω_T is obtained. In addition, an implementation error of the neutral bits technique contributed a factor to the excessive pairs that were tested.

8. Attacks on Reduced Versions of SHA-1

The only difference between SHA-1 and SHA-0 is an additional rotation operation in the expansion process that mixes the bits in the expanded message in a more efficient way than SHA-0 does. However, the expansion remains linear, and therefore the construction of local collision sequences with the technique described in the previous sections is still applicable.

The bit rotation increases the Hamming weight of disturbance vectors. However, it is not sufficient to make this weight as heavy as in random codes of the same size. Indeed, surprisingly low weight vectors still exist in SHA-1.

Even though the additional rotation mixes the message bits in a more efficient way, it does not affect the efficiency of the neutral bits technique. Therefore, the neutral bits technique can be used to attack SHA-1 with no modifications. We note that the existence of neutral bits in SHA-1 is not a surprise, as it is still possible to create small differences in the state vector with the local collision sequences technique.

Table 15. A characteristic of a four-block attack on SHA-0 (first block).

$\Omega_P^1 = (0, 0, 0, 0, 0) = \mathbf{00000}$

R.	$\mathcal{D}$	Ω_W	Ω_A	p	R.	$\mathcal{D}$	Ω_W	Ω_A	p	R.	$\mathcal{D}$	Ω_W	Ω_A	p	R.	$\mathcal{D}$	Ω_W	Ω_A	p
−5	0				17	2	80000002	0	1/2	39	0	80000000	0		61	0	00000000	0	1/2
−4	0		0		18	2	80000042	2	1/4	40	0	00000000	0		62	0	00000000	0	
−3	0		0		19	2	00000040	2	1/2	41	0	00000000	0		63	0	80000000	0	
−2	0		0		20	2	80000040	2	1/4	42	0	00000000	0		64	0	00000000	0	
−1	0		0		21	0	00000042	2	1/4	43	0	00000000	0		65	2	00000002	0	
0	0	00000000	0		22	0	80000002	0	1/4	44	0	00000000	0		66	2	00000042	2	1/2
1	0	00000000	0		23	2	80000002	0	1/2	45	0	00000000	0		67	0	00000042	2	1/2
2	0	00000000	0		24	0	00000040	2	1/2	46	2	00000002	0		68	0	80000002	0	1/2
3	2	00000002	0		25	2	80000000	0		47	2	00000042	2	1/2	69	0	00000000	0	1/2
4	0	00000040	2	1/2	26	2	80000042	2	1/2	48	0	00000042	2	1/2	70	2	00000002	0	
5	0	00000002	0		27	0	80000042	2	1/4	49	0	80000002	0	1/2	71	0	80000040	2	1/2
6	0	80000000	0	1/4	28	0	00000002	0	1/2	50	0	00000000	0	1/4	72	2	00000000	0	
7	0	80000000	0	1/2	29	0	00000000	0	1/2	51	0	00000000	0		73	2	80000042	2	1/2
8	2	80000002	0	1/2	30	0	00000000	0		52	0	80000000	0	1/2	74	0	80000042	2	1/4
9	0	00000040	2	1/2	31	0	80000000	0		53	0	00000000	0		75	2	00000000	0	1/2
10	2	00000000	0	1/4	32	2	00000002	0		54	2	00000002	0		76	2	00000042	2	1/2
11	0	80000040	2	1/4	33	2	00000042	2	1/2	55	2	00000042	2	1/2	77	0	00000042	2	1/4
12	0	80000002	0	1/8	34	2	00000040	2	1/2	56	0	00000042	2	1/2	78	0	00000002	0	1/2
13	2	00000002	0		35	0	80000042	2	1/2	57	2	80000000	0	1/2	79	0	00000000	0	1/2
14	0	80000040	2	1/4	36	0	00000002	0	1/4	58	2	00000042	2	1/8	80			0	
15	0	80000002	0	1/2	37	0	80000000	0	1/2	59	0	00000042	2	1/2					
16	0	80000000	0	1/4	38	0	00000000	0		60	0	00000002	0	1/4					

$\Omega_T^1 = (0, 0, 0, 2, 2) = \boldsymbol{00011}$

Table 16. A characteristic of a four-block attack on SHA-0 (second block).

$\Omega_P^2 = \Omega_T^1 = (0, 0, 0, 2, 2) = \boldsymbol{00011}$

R.	$\mathcal{D}$	Ω_W	Ω_A	p	R.	$\mathcal{D}$	Ω_W	Ω_A	p	R.	$\mathcal{D}$	Ω_W	Ω_A	p	R.	$\mathcal{D}$	Ω_W	Ω_A	p
−5	**0**				17	0	80000040	2	1/4	39	2	00000042	2	1/2	61	2	00000042	2	1/2
−4	2		**2**		18	0	80000002	0	1/4	40	2	00000040	2	1/2	62	0	00000042	2	1/2
−3	0		2		19	2	00000002	0	1/4	41	0	80000042	2	1/4	63	2	80000000	0	1/2
−2	0		0		20	0	80000040	2	1/4	42	0	00000002	0	1/4	64	2	00000042	2	1/2
−1	0		0		21	0	80000002	0		43	0	80000000	0	1/2	65	0	00000042	2	1/4
0	2	80000002	0		22	0	80000000	0	1/2	44	0	00000000	0		66	0	00000002	0	1/2
1	0	80000040	2	1/4	23	2	80000002	0		45	0	80000000	0	1/2	67	0	00000000	0	1/2
2	0	00000002	0		24	2	80000042	2	1/2	46	0	00000000	0		68	0	00000000	0	
3	0	80000000	0	1/4	25	2	00000040	2	1/2	47	0	00000000	0		69	0	80000000	0	
4	0	80000000	0	1/2	26	2	80000040	2	1/2	48	0	00000000	0		70	0	00000000	0	
5	0	80000000	0	1/2	27	0	00000042	2	1/4	49	0	00000000	0		71	2	00000002	0	
6	0	00000000	0		28	0	80000002	0	1/4	50	0	00000000	0		72	2	00000042	2	1/2
7	0	00000000	0		29	2	80000002	0	1/2	51	0	00000000	0		73	0	00000042	2	1/2
8	0	00000000	0		30	0	00000040	2	1/2	52	2	00000002	0		74	0	80000002	0	1/2
9	2	00000002	0		31	2	80000000	0		53	2	00000042	2	1/2	75	0	00000000	0	1/2
10	0	00000040	2	1/2	32	2	80000042	2	1/2	54	0	00000042	2	1/2	76	2	00000002	0	
11	0	00000002	0		33	0	80000042	2	1/4	55	0	80000002	0	1/2	77	0	80000040	2	1/2
12	0	80000000	0	1/4	34	0	00000002	0	1/2	56	0	00000000	0	1/4	78	2	00000000	0	
13	0	80000000	0	1/2	35	0	00000000	0	1/2	57	0	00000000	0		79	2	80000042	2	1/2
14	2	80000002	0	1/2	36	0	00000000	0		58	0	80000000	0	1/2	80			2	1/4
15	0	00000040	2	1/2	37	0	80000000	0		59	0	00000000	0						
16	2	00000000	0		38	2	00000002	0		60	2	00000002	0						

$\Omega_T^2 = (2, 2, 0, 0, 2) = \boldsymbol{11001}$

Table 17. A characteristic of a four-block attack on SHA-0 (third block).

$\Omega_P^3 = \Omega_T^2 = (2, 2, 0, 0, 2) = \boldsymbol{11001}$

R.	$\mathcal{D}$	Ω_W	Ω_A	p	R.	$\mathcal{D}$	Ω_W	Ω_A	p	R.	$\mathcal{D}$	Ω_W	Ω_A	p	R.	$\mathcal{D}$	Ω_W	Ω_A	p
−5	2				17	2	80000042	2	1/4	39	0	80000002	0	1/2	61	2	00000040	2	1/2
−4	0		2		18	2	80000040	2	1/8	40	0	00000000	0	1/2	62	0	80000042	2	1/2
−3	**0**		0		19	2	00000040	2	1/8	41	0	00000000	0		63	2	00000000	0	1/4
−2	0		0		20	2	00000040	2	1/16	42	2	80000002	0		64	0	80000040	2	1/2
−1	2		**2**		21	2	80000040	2	1/4	43	0	00000040	2	1/2	65	0	00000002	0	1/2
0	0	80000040	2		22	0	80000042	2	1/4	44	2	00000000	0		66	2	00000002	0	1/2
1	0	00000002	0	1/2	23	0	80000002	0	1/4	45	2	80000042	2	1/4	67	2	80000042	2	1/2
2	2	80000002	0	1/4	24	0	80000000	0	1/2	46	0	80000042	2	1/4	68	2	80000040	2	1/2
3	0	80000040	2	1/2	25	0	00000000	0		47	0	00000002	0	1/4	69	0	80000042	2	1/2
4	0	80000002	0	1/2	26	2	80000002	0		48	2	00000002	0	1/4	70	2	00000000	0	1/4
5	2	80000002	0	1/4	27	0	00000040	2	1/2	49	0	00000040	2	1/2	71	0	80000040	2	1/2
6	0	80000040	2	1/4	28	0	00000002	0		50	0	80000002	0		72	0	00000002	0	1/2
7	2	80000000	0	1/2	29	0	80000000	0	1/2	51	0	80000000	0	1/2	73	0	00000000	0	1/2
8	0	80000040	2	1/4	30	0	80000000	0		52	0	80000000	0	1/2	74	0	80000000	0	
9	0	80000002	0	1/4	31	2	80000002	0		53	0	80000000	0	1/2	75	2	80000002	0	
10	0	00000000	0	1/8	32	0	00000040	2	1/2	54	0	00000000	0		76	0	00000040	2	1/2
11	2	80000002	0	1/2	33	0	00000002	0		55	0	00000000	0		77	0	00000002	0	
12	0	80000040	2	1/4	34	0	80000000	0	1/2	56	0	00000000	0		78	0	80000000	0	1/2
13	0	00000002	0		35	0	80000000	0		57	0	00000000	0		79	2	80000002	0	1/2
14	2	80000002	0	1/4	36	2	80000002	0		58	0	00000000	0		80			2	
15	0	80000040	2	1/4	37	2	00000042	2	1/2	59	2	00000002	0						
16	2	80000000	0	1/2	38	0	00000042	2	1/2	60	2	00000042	2	1/2					

$\Omega_T^3 = (0, 2, 0, 0, 0) = \boldsymbol{01000}$

Table 18. A characteristic of a four-block attack on SHA-0 (fourth block).

$\Omega_P^4 = \Omega_T^3 = (0, 2, 0, 0, 0) = \boldsymbol{01000}$

R.	$\mathcal{D}$	Ω_W	Ω_A	p	R.	$\mathcal{D}$	Ω_W	Ω_A	p	R.	$\mathcal{D}$	Ω_W	Ω_A	p	R.	$\mathcal{D}$	Ω_W	Ω_A	p
−5	**2**				17	0	00000040	2	1/2	39	0	80000000	0		61	0	00000000	0	
−4	**2**		**0**		18	2	00000000	0		40	2	00000002	0		62	2	00000002	0	
−3	0		**0**		19	0	80000040	2	1/4	41	2	00000042	2	1/2	63	2	00000042	2	1/2
−2	2		0		20	0	80000002	0	1/4	42	2	00000040	2	1/2	64	0	00000042	2	1/2
−1	0		2		21	2	00000002	0	1/2	43	0	80000042	2	1/4	65	2	80000000	0	1/2
0	0	00000002	0		22	0	80000040	2	1/2	44	0	00000002	0	1/4	66	2	00000042	2	1/2
1	0	00000000	0	1/4	23	0	80000002	0		45	0	80000000	0	1/2	67	0	00000042	2	1/4
2	2	80000002	0	1/2	24	0	80000000	0	1/2	46	0	00000000	0		68	0	00000002	0	1/2
3	0	80000040	2	1/4	25	2	80000002	0		47	0	80000000	0	1/2	69	0	00000000	0	1/2
4	0	00000002	0		26	2	80000042	2	1/2	48	0	00000000	0		70	0	00000000	0	
5	0	80000000	0	1/4	27	2	00000040	2	1/2	49	0	00000000	0		71	0	80000000	0	
6	0	80000000	0	1/2	28	2	80000040	2	1/2	50	0	00000000	0		72	0	00000000	0	
7	0	80000000	0	1/2	29	0	00000042	2	1/4	51	0	00000000	0		73	2	00000002	0	
8	0	00000000	0		30	0	80000002	0	1/4	52	0	00000000	0		74	2	00000042	2	1/2
9	0	00000000	0		31	2	80000002	0	1/2	53	0	00000000	0		75	0	00000042	2	1/2
10	0	00000000	0		32	0	00000040	2	1/2	54	2	00000002	0		76	0	80000002	0	1/2
11	2	00000002	0		33	2	80000000	0		55	2	00000042	2	1/2	77	0	00000000	0	1/2
12	0	00000040	2	1/2	34	2	80000042	2	1/2	56	0	00000042	2	1/2	78	2	00000002	0	
cre 13	0	00000002	0		35	0	80000042	2	1/4	57	0	80000002	0	1/2	79	0	80000040	2	
14	0	80000000	0	1/4	36	0	00000002	0	1/2	58	0	00000000	0	1/4	80			0	
15	0	80000000	0	1/2	37	0	00000000	0	1/2	59	0	00000000	0						
16	2	80000002	0	1/2	38	0	00000000	0		60	0	80000000	0	1/2					

$\Omega_T^4 = (0, 0, 0, 0, 0) = \boldsymbol{00000}$

Table 19. A four-block collision of SHA-0.

M	
M_1	A766A602 B65CFFE7 73BCF258 26B322B3 D01B1A97 2684EF53 3E3B4B7F 53FE3762
	24C08E47 E959B2BC 3B519880 B9286568 247D110F 70F5C5E2 B4590CA3 F55F52FE
M_2	EFFD4C8F E68DE835 329E603C C51E7F02 545410D1 671D108D F5A4000D CF20A439
	4949D72C D14FBB03 45CF3A29 5DCDA89F 998F8755 2C9A58B1 BDC38483 5E477185
M_3	F96E68BE BB0025D2 D2B69EDF 21724198 F688B41D EB9B4913 FBE696B5 457AB399
	21E1D759 1F89DE84 57E8613C 6C9E3B24 2879D4D8 783B2D9C A9935EA5 26A729C0
M_4	6EDFC501 37E69330 BE976012 CC5DFE1C 14C4C68B D1DB3ECB 24438A59 A09B5DB4
	35563E0D 8BDF572F 77B53065 CEF31F32 DC9DBAA0 4146261E 9994BD5C D0758E3D
M^*	
M_1^*	A766A602 B65CFFE7 73BCF258 26B322B1 D01B1AD7 2684EF51 BE3B4B7F D3FE3762
	A4C08E45 E959B2FC 3B519880 39286528 A47D110D 70F5C5E0 34590CE3 755F52FC
M_2^*	6FFD4C8D 668DE875 329E603E 451E7F02 D45410D1 E71D108D F5A4000D CF20A439
	4949D72C D14FBB01 45CF3A69 5DCDA89D 198F8755 AC9A58B1 3DC38481 5E4771C5
M_3^*	796E68FE BB0025D0 52B69EDD A17241D8 7688B41F 6B9B4911 7BE696F5 C57AB399
	A1E1D719 9F89DE86 57E8613C EC9E3B26 A879D498 783B2D9E 29935EA7 A6A72980
M_4^*	6EDFC503 37E69330 3E976010 4C5DFE5C 14C4C689 51DB3ECB A4438A59 209B5DB4
	35563E0D 8BDF572F 77B53065 CEF31F30 DC9DBAE0 4146261C 1994BD5C 50758E3D

Table 20. The intermediate chaining variables and differences of the four-block collision of SHA-0.

Common initial value					
$h_0 = h_0^*$	67452301	EFCDAB89	98BADCFE	10325476	C3D2E1F0
h_0'	00000000	00000000	00000000	00000000	00000000
h_1	83C1CE2D	C5BF5480	C2AF2358	104B337B	9E78A1E7
h_1^*	83C1CE2D	C5BF5480	C2AF2358	904B337B	1E78A1E7
h_1'	00000000	00000000	00000000	80000000	80000000
h_2	27AE025A	9D36F7B6	29FA88E7	87B70063	984119F3
h_2^*	27AE0258	9D36F7B4	29FA88E7	87B70063	184119F3
h_2'	00000002	00000002	00000000	00000000	80000000
h_3	4DD120B4	D6EC801F	468628A7	0CC26464	371F36B2
h_3^*	4DD120B4	D6EC801D	468628A7	0CC26464	371F36B2
h_3'	00000000	00000002	00000000	00000000	00000000
Common chaining value					
$h_4 = h_4^*$	81FB4643	08FDF1F4	A3C4F3A3	6188FED3	FD2378E6
h_4'	00000000	00000000	00000000	00000000	00000000
Common hash result					
	C9F16077	7D4086FE	8095FBA5	8B7E20C2	28A4006B

The multi-block technique is applicable to any differential attack on iterative hash functions, including SHA-1. Actually this technique is more important in the case of SHA-1 since it is difficult to find a disturbance vector that predicts a single-block collision.

8.1. *Selecting a Disturbance Vector for the Attack*

The attacks on SHA-0 use only bit 1 as the location of disturbances. This selection is possible since the LFSR that expands the message does not mix bits in different

Table 21. Probabilities and complexities of the four-block attack on SHA-0.

Block	$p_{22\to 75}$	$p_{75\to \Omega_T}$	$p_{connect}$	p_{total}	Complexity
1	2^{-38}	2^{-5}			
			2^{-2}	2^{-45}	2^{43}
2	2^{-40}	2^{-4}			
			2^{-3}	2^{-47}	2^{45}
3	2^{-40}	2^{-3}			
			2^{-2}	2^{-45}	2^{43}
4	2^{-39}	2^{-2}			
			1	2^{-41}	2^{39}

bit indices. Furthermore, disturbances in bit 1 have the highest probability since the probabilistic behavior of the carry when corrections are made to bit 31 is eliminated. In SHA-1 the rotation by one bit in the expansion process moves the disturbances to different bit indices. Therefore, bits in several locations are used for disturbances. Notice that since disturbances may be located in different bit indices, the short representations we used in SHA-0 for disturbance vectors, state differences, and initial and final chaining differences are not applicable for SHA-1.

Since disturbances can be made at any bit location, the number of candidate disturbance vectors is raised from 2^{16} in SHA-0 to 2^{512} in SHA-1. From these disturbance vectors those that have the minimal Hamming weight should be selected. In order to reduce the search domain, we heuristically chose disturbances in adjacent bits, e.g., bits 0 and 1. With this selection, disturbances sometimes cancel each other in the expansion process due to the rotation by one bit. By searching in a domain of vectors with adjacent disturbances we found some that lead to attacks faster than the birthday attack on various reduced versions of SHA-1.

Once a disturbance vector with a low Hamming weight is found we may cyclically rotate all the words of the vector simultaneously by one bit and obtain a different disturbance vector with the same low Hamming weight. Similarly, different disturbance vectors with the same low Hamming weight are obtained by a rotation of two bits, three bits, and so forth, up to 31 bits. Each of those 32 disturbance vectors with the same Hamming weight may have a different probability, mainly due to the different number of computations with differences at bit location 31 where probabilistic behavior of the carry is eliminated. Therefore, once a low Hamming weight disturbance vector is found, we compute the probabilities of each of the 32 different disturbance vectors that are obtained by rotations, and select the one with the highest probability for the attack.

8.2. *Constructing a Two-Edge Path with the Same Disturbance Vector*

As we have already seen in Example 10 and in the attack on full SHA-0 (Sect. 7), a disturbance vector may be compatible with many initial chaining differences. In particular, a disturbance vector which is compatible with a zero chaining difference and with the chaining difference that it predicts might be used for our construction: Firstly, since it is compatible with a zero chaining difference then it can be used to construct the edge

that leaves the zero initial value toward a near-collision. Secondly, it is compatible with the differences it predicts, thus it can also be used to construct an edge that leaves this near-collision. Finally, by Eq. (13) (which is applicable to SHA-0 as well as to SHA-1) it predicts a collision, thus it meets the requirements for the construction of our path.

The first advantage of such a construction is its simplicity: Instead of constructing a very complex transition graph and then search for the optimal path, we are concentrated only on a very specific type of paths in a graph. The second advantage is that using the efficient algorithm that finds disturbance vectors with low Hamming weight (see Sect. 8.1), the paths that are generated by the construction have high probability. Moreover, if the disturbance vector that is used to construct the path has the highest probability then this path is optimal. We note that by using carries as an additional resource for disturbances and corrections the number of compatible chaining differences substantially increases. Thus, if a low Hamming weight disturbance vector is found then it is most likely suitable for the construction.

8.3. *A Collision of SHA-1 Reduced to 40 Rounds*

This section demonstrates the idea of Sect. 8.2 by a typical type of a two-block collision. The two blocks use the same basic characteristic with some changes in the first few rounds that allow the concatenation of the two blocks. This type of attack is generally more efficient than finding two different characteristics when using the general construction of the multi-block technique. It enables high probabilities in the two blocks even when only one characteristic with high probability is known (compared with the other case in which a lower probability characteristic is also used).

The construction of a two-edge path with the same disturbance vector is demonstrated on SHA-1 reduced to 40 rounds. The disturbance vector we are using is the same vector used in the 34-round attack [7], rotated by 28 bits to the left and expanded to 40 rounds. The characteristic of the first and second blocks are given in Table 22.

Using these characteristics we easily found a two-block collision of 40-round SHA-1. The messages and chaining variables of the 40-round collision are given in Table 23. The probability and complexity of the attack is given in Table 24.

8.4. *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 are correctable. We therefore list here two sets of results: the first set of results for SHA-1 reduced to fewer rounds, where consecutive disturbances are correctable by the techniques we have already described. The second set of results, denoted later by NO-IF, have consecutive disturbances which are not correctable by these techniques. These disturbance vectors might be used if the reduced version of SHA-1 starts at a different location (such as from Round 20 with the XOR function at the first 20 rounds). They also may be used if these consecutive disturbances would be corrected with more creative usage of carries as a source of disturbances and corrections.

Table 25 lists the Hamming weights of the best disturbance vectors results we found to attack various reduced versions of SHA-1. For each reduced version, and each set of

Table 22. A two-block characteristic of 40-round reduced SHA-1.

$\Omega_P^1 = (0, 0, 0, 0, 0)$

R.	$\mathcal{D}$	Ω_W	Ω_A	p
−5	00000000			
−4	00000000		00000000	
−3	00000000		00000000	
−2	00000000		00000000	
−1	00000000		00000000	
0	20000000	20000000	00000000	
1	00000000	00000004	20000000	1/2
2	20000000	00000000	00000000	
3	00000000	08000004	20000000	1/2
4	20000000	08000000	00000000	1/2
5	00000000	00000004	20000000	1/4
6	30000000	18000000	00000000	1/4
7	00000000	00000006	30000000	1/16
8	00000000	38000000	00000000	1/4
9	20000000	24000000	00000000	1/32
10	00000000	0C000004	20000000	1/8
11	00000000	2C000000	00000000	1/4
12	00000000	08000000	00000000	1/4
13	00000000	08000000	00000000	1/2
14	20000000	28000000	00000000	1/2
15	00000000	00000004	20000000	1/2
16	00000000	20000000	00000000	
17	00000000	08000000	00000000	1/2
18	00000000	08000000	00000000	1/2
19	00000000	08000000	00000000	1/2
20	20000000	20000000	00000000	
21	00000000	00000004	20000000	1/2
22	20000000	00000000	00000000	
23	00000000	08000004	20000000	1/2
24	00000000	28000000	00000000	1/4
25	00000000	00000000	00000000	1/4
26	00000000	08000000	00000000	1/2
27	00000000	08000000	00000000	1/2
28	00000000	00000000	00000000	
29	00000000	00000000	00000000	
30	00000000	00000000	00000000	
31	00000000	00000000	00000000	
32	00000000	00000000	00000000	
33	00000000	00000000	00000000	
34	40000000	40000000	00000000	
35	00000000	00000008	40000000	1/2
36	00000000	40000000	00000000	
37	80000000	90000000	00000000	1/2
38	40000000	50000010	80000000	1/2
39	00000000	90000008	40000000	1/8
40			00000000	

$\Omega_T^1 = (0, 40000000, 20000000, 0, 0)$

$\Omega_P^2 = \Omega_T^1 = (0, 40000000, 20000000, 0, 0)$

R.	$\mathcal{D}$	Ω_W	Ω_A	p
−5	00000000			
−4	00000000		00000000	
−3	00000000		00000000	
−2	00000000		80000000	
−1	00000000		40000000	
0	20000000	20000000	00000000	
1	00000000	00000004	20000000	1/8
2	20000000	00000000	00000000	1/4
3	00000000	08000004	20000000	1/8
4	20000000	08000000	00000000	1/16
5	00000000	00000004	20000000	1/4
6	30000000	18000000	00000000	1/4
7	00000000	00000006	30000000	1/16
8	00000000	38000000	00000000	1/4
9	20000000	24000000	00000000	1/32
10	00000000	0C000004	20000000	1/8
11	00000000	2C000000	00000000	1/4
12	00000000	08000000	00000000	1/4
13	00000000	08000000	00000000	1/2
18	00000000	08000000	00000000	1/2
19	00000000	08000000	00000000	1/2
20	20000000	20000000	00000000	
21	00000000	00000004	20000000	1/2
22	20000000	00000000	00000000	
23	00000000	08000004	20000000	1/2
24	00000000	28000000	00000000	1/4
25	00000000	00000000	00000000	1/4
26	00000000	08000000	00000000	1/2
27	00000000	08000000	00000000	1/2
28	00000000	00000000	00000000	
29	00000000	00000000	00000000	
30	00000000	00000000	00000000	
31	00000000	00000000	00000000	
32	00000000	00000000	00000000	
33	00000000	00000000	00000000	
34	40000000	40000000	00000000	
35	00000000	00000008	40000000	1/2
36	00000000	40000000	00000000	

Table 22. continued.

$\Omega_P^2 = \Omega_T^1 = (0, 40000000, 20000000, 0, 0)$

R.	$\mathcal{D}$	Ω_W	Ω_A	p	R.	$\mathcal{D}$	Ω_W	Ω_A	p
14	20000000	28000000	00000000	1/2	37	80000000	90000000	00000000	1/2
15	00000000	00000004	20000000	1/2	38	40000000	50000010	80000000	1/2
16	00000000	20000000	00000000		39	00000000	90000008	40000000	1/4
17	00000000	08000000	00000000	1/2	40			00000000	

$\Omega_T^2 = (0, 0, 0, 0, 0)$

Table 23. Two-block collision of 40-round SHA-1.

Common initial value	
h_0	67452301 EFCDAB89 98BADCFE 10325476 C3D2E1F0
M_1	404B674C B70CB385 D2DDAC0D 3A0E9BD3 CA7F1780 7FEFDA17 05E43AF2 444344C2
	641A2CB6 86C2CFE6 EBCDEF67 6577E095 1A9CAD10 CFE48484 78639157 B13B759A
M_1^*	604B674C B70CB381 D2DDAC0D 320E9BD7 C27F1780 7FEFDA13 1DE43AF2 444344C4
	5C1A2CB6 A2C2CFE6 E7CDEF63 4977E095 129CAD10 C7E48484 50639157 B13B759E
Chaining values and differences	
h_1	2B0283FF 1E8DD54E DEB06917 E978C73E 19AE1EEB
h_1^*	2B0283FF 5E8DD54E FEB06917 E978C73E 19AE1EEB
h_1'	00000000 40000000 20000000 00000000 00000000
M_2	E63C47F7 0AB5F259 47DE1E6B 09E06877 6229CC42 604CF1AB 9B14B8F3 7261186C
	1A5370F9 822E13EB FB7157EF 6B0919C5 1F3D744B FA4DE198 FBB10C06 FDA3C3E9
M_2^*	C63C47F7 0AB5F25D 47DE1E6B 01E06873 6A29CC42 604CF1AF 8314B8F3 7261186A
	225370F9 A62E13EB F77157EB 470919C5 173D744B F24DE198 D3B10C06 FDA3C3ED
Common chaining value	
h_2	F8C3A3AF 9386BE31 433ABEA3 E5467C05 0BD7EF08

Table 24. Probabilities and complexities of the 40-round SHA-1 attack.

Block	$p_{22\to35}$	$p_{35\to\Omega_T}$	$p_{connect}$	p_{total}	Complexity
1	2^{-8}	2^{-5}			
			2^{-4}	2^{-17}	2^{15}
2	2^{-8}	2^{-4}			
			1	2^{-12}	2^{10}

results (SHA-1 or NO-IF) the table lists the Hamming weight of the disturbance vector from round 20 for three cases: the first, marked by *HW*, is the Hamming weight of the best disturbance vector predicting a single-block collision. 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. The numbers in boldface are the number of disturbances in the disturbance vectors we actually used to attack the reduced version. We note that the disturbance vectors that are listed under the *2B* column are used in a two-block attack with the technique described in Sect. 8.1, i.e., the same disturbance vector is used to construct the characteristic of the first and second block.

Table 25. The hamming weights of the best disturbance vectors that we found (counted from round 20).

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

Table 26. The disturbance vector of 53-round SHA-1.

Rnd.	$\mathcal{D}$	Rnd.	$\mathcal{D}$	Rnd.	$\mathcal{D}$	Rnd.	$\mathcal{D}$
−5	00000000	−1	00000000	3	10000000	7	D0000000
−4	00000000	0	00000000	4	00000000	8	00000000
−3	00000000	1	00000000	5	80000000	9	40000000
−2	00000000	2	00000000	6	20000000	10	A0000000

The complexities of the attacks can be approximated by the number of disturbances in rounds where the IF, XOR, and MAJ functions are used. For each disturbance where the IF, XOR, and MAJ functions are in use we approximate probabilities of 2^{-5}, 2^{-2}, and 2^{-4}, respectively. The overall complexity is then $2^{5\cdot HW_{IF}+2\cdot HW_{XOR}+4\cdot HW_{MAJ}}$, where HW is the Hamming weight of the disturbance vector from round 20 and on (i.e., the *HW* value in Table 25). A less accurate approximation of $2^{3\cdot HW}$ might be used.

Using this approximation, we can see that disturbance vectors with Hamming weight of up to 26 predict collisions 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).

According to Table 25, SHA-1 reduced to 53 rounds can be attacked with a complexity less than a generic attack. It is a single block attack and the disturbance vector from which the characteristic is derived is given in Table 26. This disturbance vector has consecutive disturbances in two locations: the first is bit 29 of $\mathcal{D}_{10}$ and $\mathcal{D}_{11}$, and the second is bit 31 of $\mathcal{D}_{15}$ and $\mathcal{D}_{16}$. From Round 20 to Round 52, this disturbance vector has 16 disturbances, thus the complexity of a characteristic which is derived from it is approximated by 2^{60}.

Table 27. The disturbance vector of 58-round SHA-1.

Rnd.	$\mathcal{D}$	Rnd.	$\mathcal{D}$	Rnd.	$\mathcal{D}$	Rnd.	$\mathcal{D}$
−5	00000000	−1	00000000	3	20000000	7	A0000000
−4	00000000	0	10000000	4	D0000000	8	20000000
−3	00000000	1	00000000	5	00000000	9	80000000
−2	00000000	2	80000000	6	40000000	10	40000000

The reduced version of SHA-1 with the largest number of rounds we could attack with a complexity less than a generic attack is SHA-1 reduced to 58 rounds. This attack is a two-block attack, and its characteristic is derived from the disturbance vector which is given in Table 27. We note that the disturbance vectors of 53-round and 58-round SHA-1 are derived from the same stream of the LFSR (the latter starts at the consequent three rounds of the former). This disturbance vector has consecutive disturbances in three locations: bit 29 of $\mathcal{D}_7$ and $\mathcal{D}_8$, bit 31 of $\mathcal{D}_{12}$ and $\mathcal{D}_{13}$ and bit 31 of $\mathcal{D}_{13}$ and $\mathcal{D}_{14}$. From Round 20 it has 17 disturbances and the approximated complexity of a two-block attack that uses characteristics which are derived from this disturbance vector, is 2^{78}.

9. Advances in Cryptanalysis of Hash Functions

At the rump session of CRYPTO 2004 two novel techniques for the cryptanalysis of hash functions were presented by Wang et al. The first technique applies differential cryptanalysis by a simultaneous usage of subtraction *and* XOR differences. This technique called *precise characteristic* is used to construct a characteristic that possibly has a low probability in the first rounds, and relatively high probability in later rounds. Given such a characteristic, the second technique shows how to use it to construct an efficient attack. This technique called *message modification technique* constructs pairs of messages that conform to the characteristic up to at least Round R with a low complexity (even though the probability of the characteristic in these rounds may be very low). Hence, the complexity of the attack is affected only by the relatively high probability in rounds beyond R.

These techniques are applicable to any hash function of the MD4 and SHA families. Wang demonstrated their usefulness on MD4 (in [35]), MD5 (in [37]), RIPEMD-128 (in [35]), SHA-0 (in [38]), SHA-1 (in [39] and [40]), and HAVAL (in [41]).

In the context of SHA-0 and SHA-1 the construction of a characteristic starts with a selection of a disturbance vector. Some of the candidate disturbance vectors have low Hamming weights in rounds beyond Round 20, thus a high probability characteristic may be derived. However, they also have disturbances at $\mathcal{D}_{-5}, \ldots, \mathcal{D}_{-1}$ (that represent differences in the initial values), or consecutive disturbances at the first 17 rounds. In [7] we showed techniques that locally use the non-linear behavior of the addition mod 2^{32} and IF operations to make some of these disturbance vectors usable. Wang et al. show how the analysis by subtraction and XOR differences facilitates the usage of the non-linear addition mod 2^{32} and IF operations such that in practice most disturbance vectors may be selected. In particular, they show how to use disturbance vectors which were considered unusable by previous techniques.

Given a disturbance vector with disturbances that represent differences in the initial values or consecutive disturbances, it is shown how to derive a characteristic. In the first rounds (usually Rounds 0,…,9) the differences of the messages cannot generate differences in the reduced state vector that have the form of local collision sequences. Therefore, the non-linear behavior of the addition mod 2^{32} and IF operations are used to control the differences of the state vector. In these rounds the subtraction and XOR differences of the characteristic represent these controlled differences. The usage of the non-linear behavior of the functions imposes many conditions on the values of certain bits, and substantially reduce the probability of the characteristic. Therefore, the characteristic is designed to minimize the number of rounds where the differences of the reduced state vector are controlled by the non-linear behavior of the functions.

Typically, starting around Round 10 (and up to the last round) the differences of the reduced state follow the disturbance vector, and the differences of the characteristic are defined accordingly. The analysis of subtraction and XOR differences in these rounds facilitate the selection of differences that give an overall high probability, and in particular higher than the probabilities of characteristics that use only XOR differences. The complete characteristic is expressed in [39] and [40] as conditions on values of message bits, state bits, and their differences.

Given a characteristic, a collision search starts by satisfying the conditions on the message bits. The remaining bits of the 16 words (that were not involved in the message-bit conditions) are randomly selected, and the conditions on the state bits are gradually satisfied from Round 0 to Round 15 by *the message modification technique*. The message modification technique complements a message bit for each unsatisfied condition. *Advanced message modification technique* is applied in Round 16 and up to Round 26. In this technique a *control path* is defined to correct each unsatisfied condition. A control path consists of one or more message and state bits, whose complementation corrects some yet unsatisfied condition, and does not affect any already satisfied conditions. The claimed complexity of these controlled corrections of all the conditions up to Round 25, is two SHA-1 calls [40]. From round 26 to the last round the remaining conditions are satisfied at random (and their success is not controlled by the attacker).

The complexity of an attack that uses these techniques is determined by two factors: The complexity of generating a pair of messages that follows the characteristic up to Round 26, and the number of conditions beyond Round 26. The attack is a two-block attack. The characteristics of the first and second blocks are constructed from the same disturbance vector. The claimed total complexity of [40] is 2^{63}.

In [13] De Cannière and Rechberger present an algorithm to find high probability characteristics of SHA-1. Their idea is based on an algorithm that estimates the complexity of a given characteristic, which is then used as a starting point to find more complex characteristics that minimize the complexity. They used their ideas and found a collision of SHA-1 reduced to 64 rounds with a complexity 2^{32}. In [14] De Cannière, Mendel and Rechberger described a better characteristic, and used it to find a collision of SHA-1 reduced to 70 rounds.

In 2007 the neutral bits technique was enhanced by Joux and Peyrin in [23]. In their paper they show how the amplified boomerang attack [34] of block ciphers is adapted to cryptanalyze hash functions using ideas from the neutral bits technique. The enhance-

ment in the context of SHA-1 is done by selecting a pair of messages that conforms to R rounds, and modify the pair by complementing certain groups of bits which are called *auxiliary differentials*. These auxiliary differentials are used to correct unsatisfied conditions at rounds beyond Round R. The claimed improvement to the attack complexity on SHA-1 is by a factor of 2^{-5}.

The techniques of De Cannière et al. [13,14] were improved in 2010 by Grechnikov [21]. His improvements include speed optimization and an improvement in the search for characteristics. With these improvements he was able to find 2-block collisions of SHA-1 reduced to 72 and 73 rounds. In 2011 Grechnikov and Adinetz [22] optimized the search technique of [21] to run on a GPU cluster, and they found a collision of SHA-1 reduced to 75 rounds. We note that both [21] and [22] report that their results were received earlier than expected. We also note that we had a similar experience in our attacks.

In [11] Chen introduces a characteristic and a collision search called *second-order differential* that leads to a collision of SHA-1 reduced to 72 rounds with a complexity of 2^{50} SHA-1 calls. It is a 2-block attack in which the first block equals in both runs, and the differences are in the second message block. This attack can be extended to a 3-block attack on the full SHA-1 in which the same disturbance vector (of the 72-round attack) is used to construct the characteristics of the second and third blocks. The selection of this disturbance vector was made such that it takes into account the dependencies of the local collision sequences and it is one of a few optimal disturbance vectors. The complexity of this attack is estimated to be 2^{58} calls of SHA-1.

In [33] Stevens analyzes the dependency of local collision sequences (which were assumed independent in most previous attacks). He introduces techniques that enable to determine the theoretical maximum success probability for a given set of local collisions, and the smallest set of message conditions that attains this probability. Using these techniques a near-collision attack of SHA-1 was found with a claimed complexity equivalent to $2^{57.5}$ SHA calls. This near-collision may be used to find a collision of SHA-1 with a complexity 2^{61}.

10. Summary

This paper presents various techniques for the cryptanalysis of hash functions. The usefulness of these techniques is demonstrated on SHA-0 and SHA-1, but they are applicable to other hash functions as well.

The neutral bits technique shows that a poor avalanche effect of a round function leads to the elimination of the probabilistic behavior of many rounds.

The multi-block technique is applicable to any iterative hash function. In the context of Merkle–Damgård construction it shows that the analysis of both inputs of the compression function (the message block and chaining variable) may result in a much more efficient attack. We also show that a two-block attack in which the same difference is used in the first and second block, is usually the most efficient.

In the case of SHA-0 we show that the collision resistance of a compression function is not monotonous with the number of rounds. We conclude that adding rounds to a compression function might result in a weaker function.

The discussion on consecutive disturbances reveals some weaknesses of the non-linear IF, MAJORITY, and addition modulo 2^{32} functions. Since these functions are used in many other algorithms, these techniques might be found useful to attack other algorithms as well.

Some of the techniques for the cryptanalysis of hash functions we discussed in this paper are also useful for the cryptanalysis of stream ciphers. We expect that with some adaptations they will be useful for the cryptanalysis of block ciphers as well.

Acknowledgements

This research was supported in part by the Israel MOD research and Technology unit.

References

[1] J.-P. Aumasson, S. Fischer, S. Khazaei, W. Meier, C. Rechberger, New features of Latin dances, in *FSE 2008*. LNCS, vol. 5086 (Springer, Berlin, 2008), pp. 470–488

[2] E. Biham, New results on SHA-0 and SHA-1, Stafford Tavares invited lecture in SAC 2004. http://www.cs.technion.ac.il/~biham/Reports/Slides/invited-talk-sac-2004.ps.gz

[3] E. Biham, R. Chen, Near-collisions of SHA-0, in *Advances in Cryptology, Proceedings of CRYPTO 2004*. LNCS, vol. 3152 (Springer, Berlin, 2004), pp. 290–305

[4] E. Biham, R. Chen, New results on SHA-0 and SHA-1, in *CRYPTO 2004 Rump Session*

[5] E. Biham, A. Shamir, Differential cryptanalysis of Snefru, Khafre, REDOC-II, LOKI and Lucifer, in *Advances in Cryptology, Proceedings of CRYPTO 1991*. LNCS, vol. 576 (Springer, Berlin, 1992), pp. 156–171

[6] E. Biham, A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard* (Springer, Berlin, 1993)

[7] E. Biham, R. Chen, A. Joux, P. Carribault, C. Lemuet, W. Jalby, Collisions of SHA-0 and reduced SHA-1, *Advances in Cryptology, Proceedings of EUROCRYPT 2005*. LNCS, vol. 3494 (Springer, Berlin, 2005), pp. 36–57

[8] B. den Boer, A. Bosselaers, An attack on the last two rounds of MD4, in *Advances in Cryptology, Proceedings of CRYPTO 1991*. LNCS, vol. 576 (Springer, Berlin, 1992), pp. 194–203

[9] B. den Boer, A. Bosselaers, Collision of the compression function of MD5, in *Advances in Cryptology, Proceedings of EUROCRYPT 1993*. LNCS, vol. 765 (Springer, Berlin, 1994), pp. 293–304

[10] F. Chabaud, A. Joux, Differential collisions in SHA-0, in *Advances in Cryptology, Proceedings of CRYPTO '98*. LNCS, vol. 1462 (Springer, Berlin, 1999), pp. 56–71

[11] R. Chen, New Techniques for Cryptanalysis of Cryptographic Hash Functions, Ph.D. thesis, Technion, 2011. http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-get.cgi/2011/PHD/PHD-2011-08.pdf and https://www.iacr.org/phds/index.php?p=detail&entry=651

[12] Ivan B. Damgård, A design principle for Hash functions, *Advances in Cryptology, Proceedings of CRYPTO 1989*. LNCS, vol. 435 (Springer, Berlin, 1990), pp. 416–427

[13] C. De Cannière, C. Rechberger, Finding SHA-1 characteristics: general results and applications, in *Advances in Cryptology, Proceedings of ASIACRYPT 2006*. LNCS, vol. 4284 (Springer, Berlin, 2006), pp. 1–20

[14] C. De Cannière, F. Mendel, C. Rechberger, Collisions for 70-Step SHA-1: on the full cost of collision search, in *Advances in Cryptology, Proceedings of SAC 2007*. LNCS, vol. 4876 (Springer, Berlin, 2007), pp. 56–73

[15] H. Dobbertin, Cryptanalysis of MD4. *J. Cryptol.***11**, 253–271 (1998)

[16] H. Dobbertin, Cryptanalysis of MD5 compress, in *EUROCRYPT 1996 Rump Session*

[17] RIPE, Integrity primitives for secure information systems, in *Final Report of RACE Integrity Primitives Evaluation (RIPE Race 1040)*. LNCS, vol. 1040 (Springer, Berlin, 1995)

[18] H. Dobbertin, A. Bosselaers, B. Preneel, RIPEMD-160: a strengthened version of RIPEMD, in *Proceedings of Fast Software Encryption*. LNCS, vol. 1039 (Springer, Berlin, 1996), pp. 71–82

[19] S. Fischer, S. Khazaei, W. Meier, Chosen IV statistical analysis for key recovery attacks on stream ciphers, in *AFRICACRYPT 2008*. LNCS, vol. 5023 (Springer, Berlin, 2008), pp. 236–245
[20] A. Joux, Collisions in SHA-0, in *CRYPTO 2004 Rump Session*
[21] E.A. Grechnikov, Collisions for 72-step and 73-step SHA-1: improvements in the method of characteristics. *Cryptology*, ePrint Archive 2010/413
[22] E.A. Grechnikov, A.V. Adinetz, Collision for 75-step SHA-1: intensive parallelization with GPU. *Cryptology*, ePrint Archive 2011/641
[23] A. Joux, T. Peyrin, Hash functions and the (amplified) Boomerang attack, in *Advances in Cryptology, Proceedings of CRYPTO 2007*. LNCS, vol. 4622 (Springer, Heidelberg, 2007), pp. 244–263
[24] S. Khazaei, W. Meier, New directions in cryptanalysis of self-synchronizing stream ciphers, in *INDOCRYPT 2008*. LNCS, vol. 5365. (Springer, Berlin, 2008), pp 15–26
[25] R. Merkle, One-way Hash function and DES, in *Advances in Cryptology, Proceedings of CRYPTO 1989*. LNCS, vol. 435 (Springer, Berlin, 1990), pp. 428–446
[26] R. Merkle, A fast software one-way Hash function. *J. Cryptol.***3**(1), 43–58 (1990)
[27] S. Miyaguchi, K. Ohta, M. Iwata, 128-bit hash function (N-Hash), in *Proceedings of SECURICOM'90*, March 1990, pp. 123–137
[28] National Institute of Standards and Technologies, Secure Hash standard, in *Federal Information Processing Standards, FIPS-180* (U.S. Department of Commerce, Washington, 1993)
[29] National Institute of Standards and Technologies, Secure Hash standard, in *Federal Information Processing Standards, FIPS-180-1* (U.S. Department of Commerce, Washington, 1995)
[30] V. Rijmen, E. Oswald, Update on SHA-1, in *RSA Crypto Track 2005*. LNCS, vol. 3376 (Springer, Heidelberg, 2005), pp. 58–71
[31] R. Rivest, The MD4 message-digest algorithm, in *Advances in Cryptology, Proceedings of CRYPTO 1990*. LNCS, vol. 537 (Springer, Berlin, 1990), pp. 303–311
[32] R. Rivest, The MD5 message-digest algorithm, in *Network Working Group Request for Comments: 1321*, April 1992
[33] M. Stevens, New collision attacks on SHA-1 based on optimal joint local-collision analysis, in *Proceedings of EUROCRYPT 2013*. LNCS, vol. 7881 (Springer, Berlin, 2013), pp. 245–261
[34] D. Wagner, The Boomerang attack, in *Advances in Cryptology, Proceedings of FSE 1999*. LNCS, vol. 1636 (Springer, Berlin, 1999), pp. 156–170
[35] X. Wang, X. Lai, H. Chen, X. Yu, Cryptanalysis of the Hash functions MD4 and RIPEMD, in *Advances in Cryptology, Proceedings of EUROCRYPT 2005*. LNCS, vol. 3494 (Springer, Berlin, 2005), pp. 1–18
[36] X. Wang, D. Feng, X. Lai, H. Yu, Collisions for Hash functions MD4, MD5, in *HAVAL-128 and RIPEMD*. http://eprint.iacr.org/2004/199
[37] X. Wang, H. Yu, How to break MD5 and other Hash functions, in *Advances in Cryptology, Proceedings of EUROCRYPT 2005*. LNCS, vol. 3494 (Springer, Berlin, 2005), pp. 19–35
[38] X. Wang, H. Yu, Y.L. Yin, Efficient collision search attacks on SHA-0, in *Advances in Cryptology, Proceedings of CRYPTO 2005*. LNCS, vol. 3621 (Springer, Berlin, 2005), pp. 1–16
[39] X. Wang, H. Yu, Y.L. Yin, Finding collisions in the full SHA-1, in *Advances in Cryptology, Proceedings of CRYPTO 2005*. LNCS, vol. 3621 (Springer, Berlin, 2005), pp. 17–36
[40] X. Wang, A.C. Yao, F. Yao, *Cryptanalysis on SHA-1*, Presented by Adi Shamir at CRYPTO 2005 rump session. http://csrc.nist.gov/groups/ST/hash/documents/Wang_SHA1-New-Result.pdf
[41] H. Yu, X. Wang, A. Yun, S. Park, Cryptanalysis of the full HAVAL with 4 and 5 passes, in *Advances in Cryptology, Proceedings of FSE 2006*. LNCS, vol. 4047 (Springer, Berlin, 2006), pp. 89–110
[42] Y. Zheng, J. Pieprzyk, J. Sebbery, HAVAL—a one-way algorithm with variable length of output, in *Asiacrypt 1992*. LNCS, vol. 718 (Springer, Berlin, 1993), pp. 83–104