Classification and Generation of Disturbance Vectors for Collision Attacks against SHA-1

Stéphane Manuel[1]

CRI - Paris Rocquencourt
stephane.manuel@inria.fr

Abstract. The main contribution of this paper is to provide a classification of disturbance vectors used in differential collision attacks against SHA-1. We show that all published disturbance vectors can be classified into two types of vectors, *type-I* and *type-II*. We present a deterministic algorithm which produce efficient disturbance vectors with respect to any given cost function. We define two simple cost functions to evaluate the efficiency of a candidate disturbance vector. Using our algorithm and those cost function we retrieved all previously known vectors and found that the most efficient disturbance vector is the one first reported as *Codeword*2 by Jutla and Patthak in [7].

Key words: Hash Functions, SHA-1, Collision Attack, Disturbance Vector.

1 Introduction

SHA-1 has been a widely used hash function since it was published by NIST as a Federal Processing Standard in 1995 [13]. SHA-1 is an evolution of a previous standard named SHA-0 [12]. SHA-1 and SHA-0 only differ in their message expansion.

Many researches have discussed collision attacks against SHA-0 and SHA-1 [5, 1, 2, 18–20, 3, 11, 17, 4, 6, 8]. Chabaud and Joux [5] pointed out the weakness of the state update transformation common to SHA-0 and SHA-1. They described a linear differential path composed of interleaved 6-step local collisions. The core of this differential path is represented by a disturbance vector (so-called L-characteristic) which indicates where the 6-step local collisions are initiated. Once a disturbance vector is chosen, one can evaluate the complexity of a collision attack against SHA-0 or SHA-1 directly from this vector. The critical factor for choosing a disturbance vector is considered to be the Hamming weight of its last 60 coordinates. A lot of work has been spent in order to find good vectors [19, 9, 7, 16, 15, 21]. The algorithms proposed are essentially probabilistic algorithms based on coding theory tools.

This article presents a generalized algorithm able to produce efficient disturbance vectors, with respect to any given cost function. Based on the experiments done using this algorithm, we present a classification for these vectors. First, we will describe our algorithm and introduce what we will call type-I and type-II classes. We then show that all the previously proposed and/or used disturbance vectors lie into these classes. We define two cost functions in order to compare the efficiency of known vectors. The optimal

The author is supported in part by the french Agence Nationale de la Recherche under the project designation EDHASH, DPg/ANR-CI FA/VB 2007-010.

vector with respect to those cost functions is the one first reported as *Codeword*2 by Jutla and Patthak in [7].

The paper is organized as follows. In Section 2, we give a brief description of SHA-1. Then, in Section 3, we describe our new algorithm, define type-I and type-II classes and show that all known vectors belong to these classes. In Section 4, we define the cost functions used in order to evaluate the efficiency of disturbance vectors and give a comparison of known vectors. We then briefly discuss how the efficiency evaluation of a disturbance vector reflects the complexity of the collision attack. Finally, we draw conclusions in Section 5.

2 Short Description of SHA-1

SHA-1 [13] is a 160-bit dedicated hash function based on the design principle of MD4. It applies the Merkle-Damgård paradigm to a dedicated compression function. The input message is padded and split into k 512-bit message blocks. At each iteration of the compression function h, a 160-bit chaining variable H_t is updated using one message block M_{t+1}, *i.e* $H_{t+1} = h(H_t, M_{t+1})$. The initial value H_0 (also called IV) is predefined and H_k is the output of the hash function.

The SHA-1 compression function is build upon the Davis-Meyer construction. It uses a function E as a block cipher with H_t for the message input and M_{t+1} for the key input, a feed-forward is then needed in order to break the invertibility of the process:

$$H_{t+1} = E(H_t, M_{t+1}) \boxplus H_t,$$

where $\boxplus$ denotes the addition modulo 2^{32} 32-bit words by 32-bit words. This function is composed of 80 steps (4 rounds of 20 steps), each processing a 32-bit message word W_i to update 5 32-bit internal registers (A, B, C, D, E). The feed-forward consists in adding modulo 2^{32} the initial state with the final state of each register. Since more message bits than available are utilized, a message expansion is therefore defined.

Message Expansion. The message block M_t is split into 16 32-bit words $W_0, \ldots, W_{15}$. These 16 words are then expanded linearly, as follows:

$$W_i = (W_{i-16} \oplus W_{i-14} \oplus W_{i-8} \oplus W_{i-3}) \lll 1 \text{ for } 16 \leq i \leq 79.$$

State Update Transformation. First, the chaining variable H_t is divided into 5 32-bit words to fill the 5 registers $(A_0, B_0, C_0, D_0, E_0)$. Then the following transformation is applied 80 times:

$$\text{STEP}_{\text{i+1}} := \begin{cases} A_{i+1} = (A_i \lll 5) \boxplus f_i(B_i, C_i, D_i) \boxplus E_i \boxplus K_i \boxplus W_i, \\ B_{i+1} = A_i, \\ C_{i+1} = B_i \ggg 2, \\ D_{i+1} = C_i, \\ E_{i+1} = D_i. \end{cases}$$

where K_i are predetermined constants and f_i are Boolean functions defined in Table 1.

round	step i	$f_i(B,C,D)$	K_i
1	$1 \leq i \leq 20$	$f_{IF} = (B \wedge C) \oplus (\overline{B} \wedge D)$	`0x5a827999`
2	$21 \leq i \leq 40$	$f_{XOR} = B \oplus C \oplus D$	`0x6ed6eba1`
3	$41 \leq i \leq 60$	$f_{MAJ} = (B \wedge C) \oplus (B \wedge D) \oplus (C \wedge D)$	`0x8fabbcdc`
4	$61 \leq i \leq 80$	$f_{XOR} = B \oplus C \oplus D$	`0xca62c1d6`

Table 1. Boolean functions and constants in `SHA-1`.

Feed-Forward. The sums: $(A_0 \boxplus A_{80})$, $(B_0 \boxplus B_{80})$, $(C_0 \boxplus C_{80})$, $(D_0 \boxplus D_{80})$, $(E_0 \boxplus E_{80})$ are concatenated to form the chaining variable H_{t+1}.

Note that all updated registers but A_{i+1} are just rotated copies, so we only need to consider the register A at each step. Thus, we have:

$$A_{i+1} = (A_i \lll 5) \boxplus f_i(A_{i-1}, A_{i-2} \ggg 2, A_{i-3} \ggg 2) \boxplus (A_{i-4} \ggg 2) \boxplus K_i \boxplus W_i.$$

3 Disturbance Vectors Searching Algorithm

3.1 Description of the Algorithm

The search algorithm we used is mainly based on the simple observation that the message expansion of `SHA-1` can be defined in two directions: forward expansion and backward expansion. Namely, one can fix $W_0, \ldots, W_{15}$ and then expand them forward to obtain $W_{16}, \ldots, W_{79}$:

- Forward expansion: $W_i = (W_{i-16} \oplus W_{i-14} \oplus W_{i-8} \oplus W_{i-3}) \lll 1$ for $16 \leq i \leq 79$.

This is the standard way defined in `SHA-1` specifications. We can also expand backward to obtain $W_{-64}, \ldots, W_{-1}$ defined by:

- Backward expansion: $W_i = (W_{i+16} \ggg 1) \oplus W_{i+13} \oplus W_{i+8} \oplus W_{i+2}$ for $-64 \leq i \leq -1$.

Any sequence of 80 consecutive 32-bit words $W_i, \ldots, W_{i+79}$ with $-64 \leq i \leq 0$ is a valid expanded message.

We will call *information window* the 16 32-bit words $W_0, \ldots, W_{15}$. For a given information window, we define an *extended expanded message* (EEM) consisting of 144 32-bit words:

$$W_{-64}, \ldots, W_{-1}, W_0, \ldots, W_{15}, W_{16}, \ldots, W_{79},$$

where $W_{-64}, \ldots, W_{-1}$ (respectively $W_{16}, \ldots, W_{79}$) are generated using the backward (respectively forward) expansion. Each EEM is composed of 65 valid expanded messages, each of these is a plausible candidate as disturbance vector.

This is the core property we use to build disturbance vectors. Our algorithm is described in Algorithm 1.

This algorithm generalizes the algorithm described by Wang *et al.* in [19]. We explicitly use forward and backward expansions. We relax the constraints imposed by Wang *et al.* both on columns and bit positions where to insert disturbances.

Previously proposed algorithms mainly focused on searching for vectors with lowest Hamming weight in the last 60 of the 80 expanded 32-bit words. Jutla and Patthak [7]

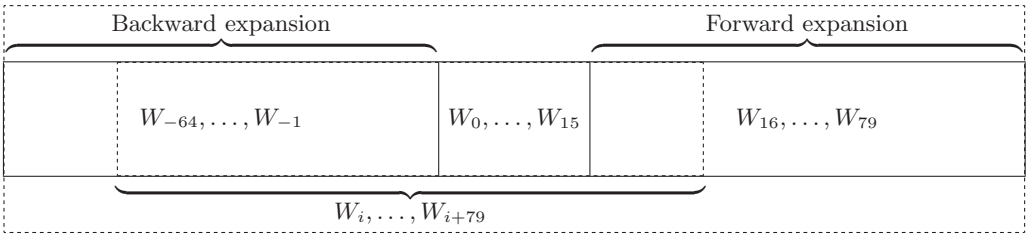

Fig. 1. Extended expanded message.

Algorithm 1 Disturbance vectors searching algorithm

Require: $w > 0$, cost function f
  for all information windows of total binary Hamming weight w or less **do**
    generate corresponding EEM
    for all 65 disturbance vectors of the EEM **do**
      evaluate efficiency with cost function f
      store vector with best evaluation
    end for
  end for
  return best vector found

have demonstrated that the minimum Hamming weight of such a vector is 25. However, the holding probability of a local collision can vary according to the bit position where it starts. A local collision starting at step 20 holds with probability 2^{-2} if the first disturbance is located on bit 1, with probability 2^{-3} on bit 31 and probability close to 2^{-4} on other bits. The holding probability also depends on the round function (IF, MAJ or XOR) where the corrections occur. Hence, a vector with higher Hamming weight may have a better efficiency than a vector with weight exactly 25. Therefore our approach consists in evaluating the efficiency of disturbance vectors without directly considering their Hamming weights. This is also the approach chosen in [4], but no algorithm is given.

Yajima *et al.* proposed in [21] an algorithm evaluating the number of chaining variable conditions (CVCs) of a disturbance vector. Their method uses new techniques in order to accurately count the number of CVCs from a disturbance vector. However following Wang *et al.* approach, they chose to define their search space as a rectangle range : disturbances may only appear on the first two columns of their information window. They do not limit the Hamming weight of their information window, but limit the positions where disturbances can occur. We use a different trade-off. We limit the Hamming weight of our information window but allow disturbances to be placed anywhere in this window.

In order to use our algorithm, we need a cost function to evaluate the efficiency of candidate disturbance vectors. We ran our algorithm using two different cost functions described in section 4.

3.2 Experimental Results

We first searched for disturbances vectors leading to two-block collision attacks. We run the algorithm with $w = 4$ and experimentally observed some facts:

– fact 1: we were able to find all previously known vectors,

– fact 2: all efficient vectors "looked" similar in some way,
– fact 3: the most efficient vectors were the same under cyclic shift of their W_i,
– fact 4: all efficient vectors mainly have their disturbances on low weight or heavy weight bits of their W_i.

Considering fact 3 and fact 4 led us to add some heuristics to our algorithm. With these heuristics, we were able to extend our search for efficient vectors to information windows of binary Hamming weight 6 or less. This new search confirmed the previous constated facts. Furthermore, we remarked that no better vector appeared during the extended search. The best vectors were obtained with information windows of weight 1 and weight 3. Table 2 describes the two information windows which lead to the best disturbance vectors.

We also searched for disturbance vectors which could lead to one-block collisions, imposing the constraint that no disturbance starts after step 75. We experimentally verified that the most efficient vector for one-block collisions is the one reported in [19].

We exhaustively tested all vectors generated by information windows of Hamming weight lower or equal to 4. However, this is only a subspace of the whole space of all possible disturbance vectors.

i	Weight 1 Information Window	Weight 3 Information Window
0	----------	----------
1	----------	o---------
2	----------	----------
3	----------	o---------
4	----------	----------
5	----------	----------
6	----------	----------
7	----------	----------
8	----------	----------
9	----------	----------
10	----------	----------
11	----------	----------
12	----------	----------
13	----------	----------
14	----------	----------
15	----------o	----------o

Table 2. Information windows leading to the most efficient disturbance vectors.

It is worth noticing that some of the observations we made were already present in the literature. Wang *et al.* [19] stated that different choices of bit position produce disturbance vectors that are rotations of each other with same Hamming weight. Rijmen and Oswald [16] noticed that the codewords they found have a large amount of W_i in common. Jutla and Patthak [7] indicated that their first codeword was earlier reported by Wang *et al.*. Pramstaller *et al.* [15] also pointed out that their vector was the same disturbance vector as the one used by Wang *et al.* with a shifted version of the indices. However to our knowledge, no publication prior to this work described a model which takes into account those observations. The interpretation of the experimental facts leads us to conclude that efficient vectors can be classified in only two classes. These classes are defined in the next section.

3.3 Classification of Disturbance Vectors

We generate disturbance vectors using information windows and extended expanded messages. It is easy to see that cyclic shifts of the W_i in a given information window will generate vectors that are rotations of each other. Also disturbance vectors within a given EEM only differ from the index i ($-64 \leq i \leq 0$) where valid expanded messages start. Based on these properties, we can define an equivalence relationship. We say that two disturbance vectors are equivalent if:

- the vectors are globally invariant under cyclic shift of their 32-bit words $W_0, \ldots, W_{79}$ or,
- the vectors are generated by the same extended expanded message.

We experimentally verified that all efficient disturbance vectors lie in only two different classes. We name these classes type-I and type-II. The type-I class contains the vector first reported by Wang *et al.* in [19]. Type-II class contains the vector first reported as $Codeword2$ by Jutla and Patthak in [7]. Table 5 and Table 6, at the end of this document, present in a synthetic way previously known vectors and show that they all are of type-I or type-II (notation $\ggg i$ indicates that to retrieve the vector published in the reference, one should cyclically shift by i bits to the left the corresponding 80 32-bit words). We now define a new notation. We will note I(i, j) (respectively II(i, j)) the disturbance vector of type-I (respectively type-II) generated as follows :

- cyclically shift by j bits to the left the 16 32-bit words of the type-I (respectively type-II) information window,
- expand backward i times,
- expand forward $64 - i$ times.

Table 3 details the corresponding notations for known disturbance vectors. The advantage of this notation is to provide a practical description for disturbance vectors. This classification also permits to explain the observations previously remarked in several papers.

Whereas vectors in the same class are equivalent according to our definition, the complexity of their associated collision attacks may vary. We discuss in the next section how we evaluated the efficiency of disturbance vectors.

4 Efficiency Evaluation

4.1 Cost Function

In order to compare the efficiency of different disturbance vectors we use a cost function. Such a cost function has to reflect the complexity of a collision attack based on a given vector. The literature describes two different approaches for complexity evaluation:

- conditions counting [8, 17–20] and,
- probabilities computation [1–6, 10].

For a given disturbance vector, the complexity of a collision attack can be evaluated by counting the number of conditions to fulfill or by computing the product of local collisions holding probabilities for each disturbance bit. We remark that complexity evaluations described in [3, 4], even if based on probabilities computations, take into account more

Disturbance vectors	Notation
Wang *et al.* CRYPTO 2005 [19]	
58 steps	$\mathrm{I}(43, 2)$
80 steps	$\mathrm{I}(49, 2)$
Rijmen & Oswald CT-RSA 2005 [16]	
$Codeword1$	$\mathrm{I}(45, 1)$
$Codeword2$	$\mathrm{I}(41, 1)$
$Codeword3$	$\mathrm{I}(39, 1)$
Jutla & Patthak ePrint 2005 [7]	
$Codeword1$	$\mathrm{I}(52, 0)$
$Codeword2$	$\mathrm{II}(52, 0)$
$Codeword3$	$\mathrm{I}(51, 0)$
Pramstaller *et al.* IMA 2005 [15]	$\mathrm{I}(50, 2)$
De Cannière & Rechberger ASIACRYPT 2006 [3]	$\mathrm{I}(35, 2)$
De Cannière *et al.* SAC 2007 [4]	$\mathrm{II}(46, 2)$
Yajima *et al.* ASIACCS 2008 [21]	$\mathrm{II}(56, 2)$

Table 3. New notation for known disturbance vectors.

factors influencing the complexity computation. Although in order to roughly evaluate candidate disturbance vectors, we arbitrarily chose to base our cost functions on a simple probability computation.

Both functions start computing probabilities from step 21 to step 80. Following [4], cost functions only discard the carry conditions in the last two steps. Cost functions also consider the technique described in [19] for two consecutive disturbances within the same step. This technique has been extended and named *strict differential bit compression* by Yajima *et al.* [21]. Cost function 1 uses *the probabilities for local collisions in SHA-1* described in the article by Mendel *et al.* at FSE 2006 [10]. Cost function 2 uses the probabilities given in Tables B.1 and B.2 of Annex B detailled in the Ph. D thesis of Thomas Peyrin [14].

We do not claim that those cost functions exactly evaluate the effective complexity of a collision attack against `SHA-1`. Their purpose is to evaluate efficiency, and a *simple* cost function is a convenient way to make a raw (and reasonably fast) comparison between candidate disturbance vectors. Furthermore, we point out that our algorithm may use any other cost function.

Using those cost functions, we gather in Table 4 all disturbance vectors we are aware of and give an efficiency comparison. We remark that the evaluations we obtained are very close to the claimed complexity of successfull collision attacks [19, 3, 4] on reduced versions of `SHA-1`. We also notice that the most efficient disturbance vector which appears in the literature is $Codeword2$ given by Jutla and Patthak [7]. This is confirmed by our experiments. With respect to the two cost functions we used, the disturbance vector of Jutla and Patthak is the best vector found by our algorithm.

Previously described algorithms based their rectangle range in order to maximize the number of perturbations occurring on bit position 1. Our approach has permitted to show that over positions may also be interresting. Indeed, this is the case of the disturbance vector of Jutla and Patthak.

We can define different cost functions, for example by starting complexity evaluation from steps higher than 21. In [20], the authors used advanced message modifications to start complexity evaluation from step 25. In [6], the authors proposed Boomerangs in the neutral bits framework in order to speed up the search. We found several new vectors which have better efficiency according to cost functions with complexity evaluation starting from step 25. However, we can not confirm the practicability of such speed-up techniques independently of the used disturbance vector.

Disturbance vectors	Given evaluations	Evaluation 1 $-\log_2$	Evaluation 2 $-\log_2$
Wang *et al.* CRYPTO 2005 [19]			
58 steps	2^{33} hash operations	35	35
80 steps	2^{69} hash operations	73	73
Rijmen & Oswald CT-RSA 2005 [16]			
80 steps			
*Codeword*1	-	90	90
*Codeword*2	-	97	97
*Codeword*3	-	102	102
Jutla & Patthak ePrint 2005 [7]			
80 steps			
*Codeword*1	-	70	76
*Codeword*2	-	65	69
*Codeword*3	-	71	76
Pramstaller *et al.* IMA 2005 [15]			
80 steps	-	73	73
De Cannière & Rechberger ASIACRYPT 2006 [3]			
64 steps	2^{35} hash operations	34	34
80 steps	-	94	94
De Cannière *et al.* SAC 2007 [4]			
70 steps	2^{44} hash operations	43	43
80 steps	-	88	88
Yajima *et al.* ASIACCS 2008 [21]			
80 steps	70 (72) CVCs	75	75

Table 4. Efficiency comparison of known disturbance vectors. Evaluation 1 (respectively 2) is computed using cost function based on Mendel *et al.* (respectively based on Peyrin).

4.2 From efficiency evaluation to complexity evaluation

The cost function computes an evaluation of the efficiency of a given disturbance vector. We should now consider how far is this efficiency evaluation from the complexity of an

effective attack against SHA-1 based on a given vector. First, our cost functions assume that fullfilling conditions from steps 16 to 20 can be done independently of the disturbance vector. This is an arguable assumption. However, this seems to be a common assumption in the literature. Second, in order to build an effective two-block collision attack against SHA-1 from a given disturbance vector, one shall need at least (1) a non-linear characteristic generator and (2) speed-up techniques. Efficient disturbance vectors may have different behaviors with respect to these tools. Thus, we do not claim that one could directly build an effective collision attack against SHA-1 from the most efficient disturbance vector found by our algorithm.

5 Conclusion

In this paper, we introduce a new algorithm to produce disturbance vectors to be used in collision attacks against SHA-1. Given a cost function, this algorithm will produce the most efficient disturbance vector with respect to this cost function. Based on the experiments done using this algorithm, we were able to retrieve all previously proposed and/or used disturbance vectors. By identifying classes of efficient vectors, we showed that all known vectors lie into type-I or type-II classes. We run our algorithm and found that the most efficient disturbance vector with respect to the chosen cost functions is the one first reported as *Codeword*2 by Jutla and Patthak in [7].

References

1. E. Biham and R. Chen. Near-Collisions of SHA-0. In M.K. Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 290–305. Springer-Verlag, 2004.
2. E. Biham, R. Chen, A. Joux, P. Carribault, C. Lemuet and W. Jalby. Collisions of SHA-0 and Reduced SHA-1. In R. Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 36–57. Springer-Verlag, 2005.
3. C. De Cannière and C. Rechberger. Finding SHA-1 Characteristics: General Results and Applications. In X. Lai and K. Chen, editors, *Advances in Cryptology – ASIACRYPT 2006*, volume 4284 of *Lecture Notes in Computer Science*, pages 1–20. Springer-Verlag, 2006.
4. C. De Cannière, F. Mendel and C. Rechberger. Collisions for 70-step SHA-1: On the Full Cost of Collision Search. in C. Adams and A. Miri and M. Wiener, editors, *Selected Areas in Cryptography – SAC 2007*, volume 4876 of *Lecture Notes in Computer Science*, pages 56–73. Springer-Verlag, 2007.
5. F. Chabaud and A. Joux. Differential Collisions in SHA-0. In H. Krawczyk, editor, *Advances in Cryptology – CRYPTO 2008*, volume 1462 of *Lecture Notes in Computer Science*, pages 56–71. Springer-Verlag, 1998.
6. A. Joux and T. Peyrin. Hash Functions and the (Amplified) Boomerang Attack. In A. Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 244–263. Springer-Verlag, 2004.
7. C.S. Jutla and A.C. Patthak. A Matching Lower Bound on the Minimum Weight of SHA-1 Expansion Code. *Cryptology ePrint Archive*, Report 2005/266, 2005. Available from: http://eprint.iacr.org.
8. S. Manuel and T. Peyrin. Collisions on SHA-0 in one hour. In K. Nyberg, editor, *Fast Software Encryption – FSE 2008*, volume 5086 of *Lecture Notes in Computer Science*, pages 16–35. Springer-Verlag, 2008.

9. K. Matusiewicz and J. Pieprzyk. Finding Good Differential Patterns for Attacks on SHA-1. In Proceedings of *International Workshop on Coding and Cryptography – WCC 2005*, volume 3969 of *Lecture Notes in Computer Science*, pages 164-177. Springer-Verlag, 2005.
10. F. Mendel, N. Pramstaller, C. Rechberger and V. Rijmen. The Impact Of Carries on the Complexity of Collision Attacks on SHA-1. In M.J.B. Robshaw, editor, *Fast Software Encryption – FSE 2006*, volume 4047 of *Lecture Notes in Computer Science*, pages 278–292. Springer-Verlag, 2006.
11. Y. Naito, Y. Sasaki, T. Shimoyama, J. Yajima, N. Kunihiro and K. Otha. Improved Collision Search for SHA-0. In X. Lai and K. Chen, editors, *Advances in Cryptology – ASIACRYPT 2006*, volume 4284 of *Lecture Notes in Computer Science*, pages 21–36. Springer-Verlag, 2006.
12. National Institute of Standards and Technology. FIPS 180: Secure Hash Standard, May 1993. Available from: `http://csrc.nist.gov`.
13. National Institute of Standards and Technology. FIPS 180-1: Secure Hash Standard, April 1995. Available from: `http://csrc.nist.gov`.
14. T. Peyrin. Analyse de fonctions de hachage cryptographiques. Ph.D Thesis in Cryptology. November 2008
15. N. Pramstaller, C. Rechberger and V. Rijmen. Exploiting Coding Theory for Collision Attacks on SHA-1. in N.P. Smart, editor, *Cryptography and Coding 2005*, volume 3796 of *Lecture Notes in Computer Science*, pages 78–95. Springer-Verlag, 2005.
16. V. Rijmen and E. Oswald. Update on SHA-1. in A.J. Menezes, editor, *The Cryptographers'Track at the RSA conference – CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 58–71. Springer-Verlag, 2005.
17. M. Sugita, M. Kawazoe, L. Perret and H. Imai. Algebraic Cryptanalysis of 58-Round SHA-1. In A. Biryukov, editor, *Fast Software Encryption – FSE 2007*, volume 4593 of *Lecture Notes in Computer Science*, pages 349–365. Springer-Verlag, 2007.
18. X. Wang, H. Yu and Y.L. Yin. Efficient Collision Search Attacks on SHA-0. In V. Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, 2005.
19. X. Wang, Y.L. Yin, and H. Yu. Finding Collisions in the Full SHA-1. In V. Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36. Springer-Verlag, 2005.
20. X. Wang, Y.L. Yin, and H. Yu. New Collision Search for SHA-1. In Proceedings of *NIST Cryptographic Hash Workshop*, 2005. Available from: `http://csrc.nist.gov`.
21. J. Yajima, T. Iwasaki, Y. Naito, Y. Sasaki, T. Shimoyama, N. Kunihiro and K. Ohta. A Strict Evaluation Method on the Number of Conditions for the SHA-1 Collision Search. In proceedings *ASIACCS 2008*. March 18-20, Tokyo, Japan, 2008.

$Type-I$ *Disturbance Vector*	Wang *et al.* [19, 20, 17, 6] $\ggg 2$	Rijmen & Oswald [16] $\ggg 1$			Jutla & Patthak [7] *Code word*1	 *Code word*3	Pramstaller *et al.* [15] $\ggg 2$	De Cannière & Rechberger [3] $\ggg 2$
----------					0			
-oo--------					1	0		
----------					2	1	0	
-o-o--------	0				3	2	1	
o---------	1				4	3	2	
o---------	2				5	4	3	
o-o---------	3				6	5	4	
-o---------	4	0			7	6	5	
----------	5	1			8	7	6	
-oo--------	6	2			9	8	7	
o---------	7	3			10	9	8	
o---------	8	4	0		11	10	9	
o---------	9	5	1		12	11	10	
----------	10	6	2	0	13	12	11	
----------	11	7	3	1	14	13	12	
-o---------	12	8	4	2	15	14	13	
----------	13	9	5	3	16	15	14	
o-o---------	14	10	6	4	17	16	15	0
o---------	15	11	7	5	18	17	16	1
o-o---------	16	12	8	6	19	18	17	2
----------	17	13	9	7	20	19	18	3
o---------	18	14	10	8	21	20	19	4
----------	19	15	11	9	22	21	20	5
oo--------	20	16	12	10	23	22	21	6
----------	21	17	13	11	24	23	22	7
o---------	22	18	14	12	25	24	23	8
o---------	23	19	15	13	26	25	24	9
-o---------	24	20	16	14	27	26	25	10
----------	25	21	17	15	28	27	26	11
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
----------	61	57	53	51	64	63	62	47
----------	62	58	54	52	65	64	63	48
----------	63	59	55	53	66	65	64	49
----------o	64	60	56	54	67	66	65	50
----------	65	61	57	55	68	67	66	51
----------	66	62	58	56	69	68	67	52
----------o-	67	63	59	57	70	69	68	53
----------	68	64	60	58	71	70	69	54
----------	69	65	61	59	72	71	70	55
----------o--	70	66	62	60	73	72	71	56
----------	71	67	63	61	74	73	72	57
----------o-	72	68	64	62	75	74	73	58
----------o---	73	69	65	63	76	75	74	59
----------	74	70	66	64	77	76	75	60
----------	75	71	67	65	78	77	76	61
----------o----	76	72	68	66	79	78	77	62
----------	77	73	69	67		79	78	63
----------o-o-	78	74	70	68			79	64
----------o-----	79	75	71	69				65
----------oo-		76	72	70				66
----------		77	73	71				67
----------o-----		78	74	72				68
----------o---		79	75	73				69
----------o-o---			76	74				70
----------o------			77	75				71
----------o---			78	76				72
----------			79	77				73
----------o----o---				78				74
----------				79				75
----------o-o-----								76
----------o--------								77
----------oo--o--								78
----------								79

Table 5. Known disturbance vectors of type-I.

$Type-II$ *Disturbance Vector*	Yajima *et al.* [21] $\ggg 2$	De Cannière *et al.* [4] $\ggg 2$	Jutla & Patthak [7] *Codeword2*
oo-o----------	0		
o-----------	1		
ooo----------	2		
o-o----------	3		
---o----------	4		
o-----------	5		0
ooo----------	6		1
--o----------	7		2
oo-o----------	8		3
o-----------	9		4
ooo----------	10		5
o-o----------	11	0	6
---o----------	12	1	7
o-----------	13	2	8
ooo----------	14	3	9
--o----------	15	4	10
oo----------	16	5	11
o-----------	17	6	12
-oo----------	18	7	13
o-----------	19	8	14
oo----------	20	9	15
o-----------	21	10	16
o-o----------	22	11	17
o-----------	23	12	18
ooo----------	24	13	19
----------	25	14	20
$\vdots$	$\vdots$	$\vdots$	$\vdots$
----------	61	51	57
----------	62	52	58
----------	63	53	59
----------	64	54	60
----------	65	55	61
----------	66	56	62
----------	67	57	63
----------	68	58	64
----------	69	59	65
----------	70	60	66
----------o	71	61	67
----------	72	62	68
----------	73	63	69
----------o-	74	64	70
----------o	75	65	71
----------	76	66	72
----------o--	77	67	73
----------o-	78	68	74
----------o-	79	69	75
----------o---		70	76
----------o--		71	77
----------		72	78
----------o--o-		73	79
----------o---		74	
----------o-o-		75	
----------o-----		76	
----------o-oo-		77	
----------		78	
----------o--o-o		79	

Table 6. Known disturbance vectors of type-II.