PAPER Special Section on Cryptography and Information Security

A Strict Evaluation Method on the Number of Conditions for SHA-1 Collision Search

Jun YAJIMA[†a)]**, *Member*, Terutoshi IWASAKI**[††*]**, Yusuke NAITO**[†††**]**, Yu SASAKI**[†††***]**, Takeshi SHIMOYAMA**[†]**, Thomas PEYRIN**[††††]**, *Nonmembers*, Noboru KUNIHIRO**[†††]**, *and* Kazuo OHTA**[†††]**, *Members***

SUMMARY This paper proposes a new algorithm for evaluating the number of chaining variable conditions(CVCs) in the selecting step of a disturbance vector (DV) for the analysis of SHA-1 collision attack. The algorithm is constructed by combining the following four strategies, *"Strict Differential Bit Compression"*, *"Differential Path Confirmation for Rounds 2 to 4"*, *"DV expansion"* and *"Precise Counting Rules in Every Step"* that can evaluate the number of CVCs more strictly compared with the previous approach.

key words: *Hash Function, Collision Attack, SHA-1, Disturbance Vector*

1. Introduction

SHA-1 has been a widely used scheme since it was issued by NIST as a Federal Information Processing Standard in 1995 [1]. Recently, many researches have discussed collision search attacks on SHA-1 [2]–[6].

In 2005, Wang et.al. succeeded in attacking the full 80 step SHA-1 with 2^{69} complexity [4]. They have adopted the multi-block collision technique introduced by [2], [7], and adjusted the differential path known by then in the first round (step 1 to step 16) to another possible differential path. In the attack, they used local collisions. Local collision (LC in short) is collision within 6-steps differential path which is independently introduced by Chabaud and Joux [8] and Wang et al.[9]. In order to find appropriate combination of LCs for SHA-1, the disturbance vector (DV) (introduced in [8], [9]) is used. After that, the attack complexity was reduced to $2^{61} \sim 2^{62}$ by improving message modification techniques in [5], [6].

Roughly speaking, collision search [4] consists of the following two phases; preparing phase and searching collision phase.

Preparing Phase

Stage(1) Select a Disturbance Vector(DV) and obtain the message differentials $\Delta M = M' - M$.

Stage(2) Locate differential paths which are the differences between two sequences of chaining variables yielded by the calculation of $H(M)$ and $H(M')$. And derive the sufficient conditions of chaining variables for the result of $H(M) = H(M')$.

Stage(3) Determine message modifications (MM) using in Stage(4) so that M satisfies all the chaining variable conditions (CVCs) and message conditions efficiently. And evaluate the complexity of collision search.

Searching Collision Phase

Stage(4) Search M using MM which satisfies all the chaining variable conditions (CVCs) and message conditions

The complexity of the above collision search attack is determined by the number of CVCs in the differential path which are not satisfied by Message modification(MM).

By the recent researches, the stages (1)-(4) in the above attack procedure were improved. On (2), an automated path search algorithm of differential paths was discussed in [10]–[12]. On (3), extending the applicable steps of MM from step 21 [4] to step 25 in [6] reduced the complexity of the collision search of SHA-1 from 2^{69} to 2^{61}. And also [3], [13], [14] have an effect on the reduction of complexity of the collision search of SHA-1. On (1), only Wang's method described in [4] has been proposed. After many researches, De Cannière et al. found a colliding message pair against reduced SHA-1 with 70 steps[15]. And recently, they have started the collision search project on their web page[16].

In the preparing phase on collision attack, the following properties are important

i Less #{total CVCs in step 17-80}
ii Many #{satisfiable CVCs by MM in step 17-80}
iii Can find a differential path
iv Much message freedom for the collision search process

In this paper, we focus on the stage(1) in the preparing phase. At the stage (1), #{total CVCs in step 17-80} (i) should be evaluated from an analyzing DV. This number affects directly the total complexity of collision search. We found that the previous evaluation method on (i) have used rough evaluation. In order to find much effective DVs, we propose a strict evaluation method on (i) by improving the

Manuscript received March 21, 2008.

[†]The author is with FUJITSU LABORATORIES LTD., Kawasaki-shi, 211-8588 Japan.

[††]The author is with Chuo University, Bunkyo-ku, 112-8551 Japan.

[†††]The University of Electro-Communications, Chofu-shi, 182-8585 Japan

[††††]Versailles Saint-Quentin-en-Yvelines University, France, and Advanced Industrial Science and Technology(AIST), Chiyoda-ku 101-0021 Japan.

[*]Presently, the author is with Nomura Research Institute, Ltd.

[**]Presently, the author is with Information Technology R & D Center, Mitsubishi Electric Corporation.

[***]Presently, the author is with NTT Information Sharing Platform Laboratories, NTT Corporation.

a) E-mail: jyajima@labs.fujitsu.com

previous method. we will point out some possibilities for improving Wang et al.'s counting rules, described in [4], and we will propose a new evaluating algorithm, which consists of *"Strict Differential Bit Compression"*, *"Differential Path Confirmation for Rounds 2 to 4"*, *"DV expansion"* and *"Precise Counting Rules in Every Step"*. Then we implement the algorithm and estimate #CVCs for an analyzing DV corresponding to the step number of SHA-1.

2. Description of SHA-1

SHA-1[1] input is an arbitrary length message M, and output is 160-bit data $H(M)$. The message is padded to realize a multiple of 512 bits. Padded message M is divided into several messages M_i each 512 bits long ($M = (M_1 \| M_2 \| ... \| M_n)$). These divided messages are input to the compression function. The structure of the compression function is as follows. In this paper, we call the calculation in a single run of the compression function "1 block", and we omit the description of "mod 2^{32}". The compression function has 80 steps. Steps 1-20, 21-40, 41-60, and 61-80 are called the first, second, third and fourth rounds, respectively.

Step 1 Divide the input message M_j into 32 bit messages $m_0, m_1, ..., m_{15}$.

Step 2 Calculate m_{16} to m_{79} by $m_i = (m_{i-3} \oplus m_{i-8} \oplus m_{i-14} \oplus m_{i-16}) \lll 1$.

Step 3 Calculate chaining variables a_i, b_i, c_i, d_i, e_i in step i by the following procedures for $i = 1, 2, ..., 80$
$a_i = (a_{i-1} \lll 5) + f(b_{i-1}, c_{i-1}, d_{i-1}) + e_{i-1} + m_{i-1} + k_{i-1}, b_i = a_{i-1}, c_i = b_{i-1} \lll 30, d_i = c_{i-1}, e_i = d_{i-1}$

Step 4 $(a_0 + a_{80}, b_0 + b_{80}, c_0 + c_{80}, d_0 + d_{80}, e_0 + e_{80})$ is the output of the compression function.

a_0, b_0, c_0, d_0, e_0 and k_i are defined in [1]. After the first block, a_0, b_0, c_0, d_0, e_0 are the output values of the compression function in the previous block. Function $f(b, c, d)$ is a Boolean function defined in each round, $(b \wedge c) \vee (\neg b \wedge d)$ for the first round, $b \oplus c \oplus d$ for the second and the fourth rounds and $(b \wedge c) \oplus (c \wedge d) \oplus (d \wedge b)$ for the third round.

3. Collision Search for SHA-1

3.1 Overview of a Collision Attack of SHA-1

Collision search procedure takes the following approach by two phases[4].

Preparing Phase

Stage(1) Select a Disturbance Vector(DV) and obtain the message differentials $\Delta M = M' - M$.

Stage(2) Locate differential paths which are the differences between two sequences of chaining variables yielded by the calculation of $H(M)$ and $H(M')$. And derive the sufficient conditions of chaining variables for the result of $H(M) = H(M')$.

Stage(3) Determine message modifications (MM) using in Stage(4) so that M satisfies all the chaining variable conditions (CVCs) and message conditions efficiently. And evaluate the complexity of collision search.

Searching Collision Phase

Stage(4) Search M using MM which satisfies all the chaining variable conditions (CVCs) and message conditions

In the proparing phase, we have to set some bitwise conditions on messages m_i and chaining variables a_i, b_i, c_i, d_i, e_i. These conditions on messages and on chaining variables are called *message condition* (MC in short) and *chaining variable condition* (CVC in short), respectively. The number of CVC (#CVC in short) impacts the complexity of the collision search of SHA-1.

For CVCs up to step 16, the technique of *basic message modification* (BMM in short) can be applied, so we can ignore these conditions. With some additional effort, we can modify the messages so that conditions after step 16 also hold. This technique is called *advanced message modification* (AMM in short). Roughly speaking, if the number of CVCs that will not be applicable for neither BMM nor AMM is n, then the complexity of collision search can be estimated by 2^n by using a naive collision search procedure.

In present, Wang et al. have finished (3) in the above procedure. And they have pointed out that the complexity needed for find a collision is 2^{69} in [4]. In [6], they reduced the complexity from 2^{69} to $2^{61} \sim 2^{62}$ by changing a DV. However, no one still find a colliding message pair because the complexity is too much for present computers.

Though their works are known as the best results for finding a collision, the stages (1)-(4) have not been described in detail in [4]. Many researchers are trying to clarify the detail and trying to improve each step. On (2), an automated path search algorithm of differential paths was discussed in [10]–[12]. On (3), message modifications were discussed in [3], [13], [14]. On (4), reduced 70-step SHA-1 was analyzed in [15]. And they found a collision of reduced SHA-1. On (1), only Wang's method has been described in [4] for 80-step full SHA-1.

3.2 Complexity for Finding a Collision of SHA-1

The complexity needed for find a collision is determined by the following equations.

$$Complexity \approx 2^{\#\{essentialCVCs\}}$$
$$\#\{essentialCVCs\} = \#\{\text{total CVCs in step 17-80}\} - \#\{\text{satisfiable CVCs by MM in step 17-80}\}$$

In order to find a colliding message pair, the following properties are also important.

i Less #{total CVCs in step 17-80}
Mainly depend on the stage (1) (in the procedure described in the previous subsection.)

ii Many #{satisfiable CVCs by MM in step 17-80}
Mainly depend on the stages (1), (2) and (3).

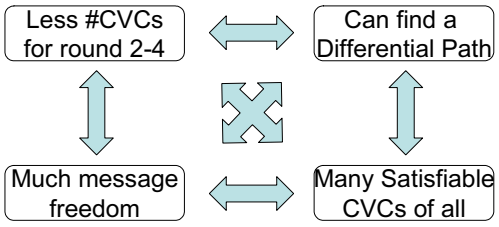


Fig. 1 Trade-off between the properties for collision search

iii Can find a differential path
Mainly depend on the stages (1) and (2).
iv Much message freedom for the collision search process
Mainly depend on the stages (1), (2) and (3).

These properties are in the trade-off relations[†]. However, at stage (1) in the attack procedure, it is very difficult to select the most effective disturbance vector by considering all the above properties. The reasons are described below.

- Each property depends on the results of the stages (1),(2) and (3). So some properties can be evaluated after the stage (3) only.
- The stage (2) consumes computer power a lot. So we cannot execute (2) many times.
- The execution order of the stages (1)-(4) is optimized. Changing the order may cause bad effect to collision attack.

By these reasons, Wang et al. only evaluate #{total CVCs in step 17-80} in the above equations at stage (1). And after stage (3), they consider all properties. However, they have used a rough evaluation method that evaluates #{total CVCs in step 17-80} from a DV analyzed at stage (1). Therefore, much effecttive DVs may not be found yet. In this paper, we focus on the evaluation algorithm of #{total CVCs in step 17-80} from a DV, and propose more strict evaluation method.

4. Previous Works on Researching Disturbance Vector

4.1 Local Collision and Disturbance Vector

It is known that collisions within 6-step that can start at any step i can be constructed on SHA-1, and real collision attacks succeeded based on this collisions. This kind of collisions is called *local collision* (LC in short). An example of local collision is shown in Table 1.

Sequences of local collisions joined together are also differential paths for SHA-1. They can be specified by the vector of 32 bit elements $(x_0, ..., x_{79})$ called *disturbance vector* (DV in short) it corresponds to the message differential on the starting steps of the local collisions.

[†]Some trade-off relations are considerable in the collision search. One of them are described in [15].

Table 1 An Example of Local Collision

Step i	x_{i-1}	m_{i-1}	a_i	b_i	c_i	d_i	e_i
i	00000001	0	0				
i+1		5		0			
i+2		0			30		
i+3		30				30	
i+4		30					30
i+5		30					

The element in the columns on x_i is a hexadecimal number. Each element e (ex. 5) in the columns of $m_i, a_i \cdots, e_i$ means number $\pm 2^e$ (ex. 00000020).

4.2 Disturbance Vector Search by Wang's Approach

This subsection shows the approach used to derive DVs as proposed by Wang et al.[4].

First, set the search space described below. We call this search space as "rectangle range".

- The search space of Wang's approach (rectangle range) $\{x_i = (0, \ldots, 0, x_{i,1}, x_{i,0}) | i = t, \ldots, t+15\}$ where $x_{i,j}$ is a bit and $t = 0, \ldots, 64$ ($65 \cdot 2^{32}$ possibilities in total), which is a part of DV in 16-step.

After that, evaluate #CVC corresponding to a DV by using the following "Wang's Calculation Algorithm of #CVC".

Wang's Calculation Algorithm of #CVC

1. Each element in rectangle range (a set of a vector corresponding 16 steps) is expanded into the vector of 80 steps using SHA-1 message expansion. Then DV $\{x_0, \ldots, x_t, \ldots, x_{t+15}, \ldots, x_{79}\}$ is generated.
2. Reduce the Hamming weight (HW) of each DV by using the special counting rule 1.
3. Estimate a number of conditions in rounds 2-4 by using the counting rules in Table 2
4. Select a DV with the lowest #CVCs.

Table 2 Rules of Wang et al. for counting #CVCs in round 2-4

step	disturb in bit 2	disturb in other bits	comments
19	0	1	For a_{21}
20	0	2	For a_{21},a_{22}
21	1	3	Condition a_{20} is "truncated"
22-36	2	4	
37	3	4	
38-40	4	4	
40-60	4	4	
61-76	2	4	
77	2	3	Conditions are "truncated"
78	2	2	starting at step 77.
79	(1)	(1)	Conditions for step 79,80
80	(1)	(1)	can be ignored in analysis.

[Special counting rules by Wang et al.[4]]

1. If two disturbances start in both bit 2 and bit 1 in the same step, then they only result in 4 conditions.
2. For Round 3, two consecutive disturbances in the same bit position only account for 6 conditions (rather than 8). This is due to the property of the MAJ function.

We note that the original Wang's Algorithm has a step for restricting of DVs which have low HW[††] in rounds 2 to 4 between the step 2 and 3 in the above algorithm. Since, the complexity of counting the #CVCs for all DVs is not so large, we need not restriction by considering the HW. Then we are comparing the #CVCs among all possible DVs in our approach described from the following section.

5. Proposed method

5.1 Problems of the previous method on the counting #CVC from one DV

On the approach of the DV search by Wang et al. described in Section 4, we found five problems in "Wang's Calculation Algorithm of #CVC". We discuss the details of the problems and introduce the countermeasures.

Problem1.Rough Application of Special Counting Rule1
Wang et al. reduce the HW of DV by the special counting rule 1. This rule can be stated as "If two local collisions start in both bit 1 and bit 0 in the same step, then they result in only 4 conditions." They consider only the case of bit 1 and 0, however, this technique can also be applied to other consecutive bits.

Problem2.Inaccurate Counting Rules
In the counting rules, Wang et al. evaluate that the #CVCs for LCs that cross from 3rd round to 4th round (step 57 to 60 in bit position 1) is 4. However, these evaluations are not correct. They count more #CVCs than needed.

Problem3.Not all cases of addition in MSB considered
In SHA-1, the #CVCs is reduced if addition is done in MSB. This case occurs if LCs start in round 2 or 4 from bit position $j, (j = 1, 31)$. However, their counting rules consider only the case of $j = 1$.

Problem4.Rough Application of Special Counting Rule2
In the method of Wang et al., Special Counting Rules 2 takes account of the property of the function f in 3rd round, and reduces the #CVCs in a special case. However, there are many other such properties for f, and then, their counting rules don't cover all the properties of f.

5.2 The Proposed Method

As we discussed in the previous section, the DV search by Wang et al. contains five problems. We use four new techniques "Strict Differential Bit Compression", "Differential Path Confirmation for Rounds 2 to 4", "DV expansion" and "Precise Counting Rules in Every Step" to solve the problems. Our algorithm is described as Algorithm1-3. In our algorithm, we accurately count the #CVCs from a DV by using the four techniques. The detail of our algorithm is described below.

[††]They claim that 27 is a reasonable threshold since three CVCs are needed, on average for each local collision so the total success probability becomes 2^{-81} or worse, which is worse than the bound of the attack complexity derived from birthday paradox (2^{-80}).

Table 3 Example of Strict Differential Bit Compression

#CVC by Wang et al.'s counting technique							
Step$i$	x_{i-1}	a_i	b_i	c_i	d_i	e_i	#CVC
18	e0000000	29,30,31					3
19	0		29,30,31				3
20	0			27,28,29			3
21	0				27,28,29		3
22	0					27,28,29	0
23	0						0
						total	12

#CVC by our technique							
Step$i$	x_{i-1}	a_i	b_i	c_i	d_i	e_i	#CVC
18	e0000000	29					1
19	0		29				1
20	0			27			1
21	0				27		1
22	0					27	0
23	0						0
						total	4

(1) Strict Differential Bit Compression

When consecutive "1"s exist in DV in the same step, we treat this DV as having only "1" exists at the lowest position in the consecutive bits. For example, we treat `e0000000` as `20000000`. As we described in Problem 1 in Section 5.1, Wang et al.'s "Special Counting Rule 1" addresses only the least significant bits. In order to identify the least #CVCs strictly, we apply the compression technique to the other bits yielding the compressed DV. As a result, we can count only essential differential bits given by DV. This solves Problem 1. However, DV compression is impossible if consecutive bits exist between the bit position 1 and 2, or 26 and 27 because there are 30-bit and 5-bit rotation shifts in the SHA-1 compression function.

Table 3 shows differential path expanded by DV. In this example, there are three differential bits in steps 18 to 22. Wang et al.'s counting technique counts twelve CVCs for those differential bits. On the other hand, our method needs only four CVCs by adding the appropriate sign condition of m_i, and by letting the condition to eliminate the carry of differentials.

(2) Differential Path Confirmation for Rounds 2 to 4

In our counting method, DVs are compressed by using the Differential Bit Compression described in (1). In some of these DVs, the differential path in rounds 2 to 4 cannot be constructed by using the compressed DV and LCs. In order to exclude such DVs, we have to confirm the contradiction of differential values in each step by using a simple procedure described in Algorithm 3. This technique also solves Problem 1.

(3) DV expansion

We derive a differential path (DP) from a compressed DV in rounds 2 to 4 in order to count #CVC in each step more precisely than Wang et al.'s special counting rule 2. We can construct a DP by using DV and LCs. For example, we expand DP from $x_{23} = 0x40000000$ and $x_{24} = 0x20000000$

Table 4 Example of Effect by DV expansion

Step i	x_{i-1}	a_i	b_i	c_i	d_i	e_i	#CVC		Comment
							Wang's	Ours	
24	40000000	30					1	1	for eliminating carry
25	20000000	28	30				2	2	for f function (see Table 5)
26	0		28	28			2	0	and eliminating carry
27	0			26	28		2	2	for f function
28	0				26	28	1	1	for f function
29	0					26	0	0	
						Total	8	6	

as shown in Table 4.

Table 4 shows an example of the effect of this technique. This technique, combined with technique (4), counts #CVC as 6, while Wang et al.'s technique counts #CVC as 8. We explain the counting method of #CVC based on DPs in the next paragraph. This technique solves Problem 4 of Wang et al.'s technique described in Section 5.1.

(4) Precise Counting Rules in Every Step

In our technique, we count #CVC from the DP derived by technique (3) in each step, while Wang et al. count #CVC by each DV bit. We constructed new counting rules to count #CVC in each step. The rules are shown in Tables 5. In our rules, the #CVCs are counted by the exact differentials in f, not DV. In order to cover all cases of addition in MSB, Each case of differentials at MSB and not at MSB are considered in round 2 and 4. Problem 2 and Problem 3 with Wang et al.'s technique shown in Section 5.1 are solved by counting in each step by referring the table for each f function. #CVC in each step is the sum of the result of i and ii.

i Count the number of "1" bits in chaining variable a. (This result gives us the number of conditions after eliminating the carry of differentials of chaining variable a.)
ii Obtain #CVC for f in each step corresponding to the differential of chaining variables b, c, d, which are elements of DP, by referring to Tables 5.

Table 5 Counting Table for XOR and MAJ functions

Differential (XOR($\Delta b_{i,j}, \Delta c_{i,j}, \Delta d_{i,j}$))	Condition bit	#CVC	
		$j = 31$	$j \neq 31$
XOR(1, 0, 0)	$a_{i-2,j+2}$	0	1
XOR(0, 1, 0)	$a_{i-1,j}$	0	1
XOR(0, 0, 1)	$a_{i-1,j}$	0	1
XOR(1, 1, 0)	none	0	
XOR(0, 1, 1)	none	0	
XOR(1, 0, 1)	none	0	
XOR(1, 1, 1)	none	0	

Differential (MAJ($\Delta b_{i,j}, \Delta c_{i,j}, \Delta d_{i,j}$))	Condition bit	#CVC
MAJ(1, 0, 0)	$a_{i-2,j+2}$	1
MAJ(0, 1, 0)	$a_{i-1,j}$	1
MAJ(0, 0, 1)	$a_{i-1,j}$	1
MAJ(1, 1, 0)	none	0
MAJ(0, 1, 1)	none	0
MAJ(1, 0, 1)	none	0
MAJ(1, 1, 1)	none	0

We omit (mod 32) on the subscription j of a.
Sign is not included in the differential but this is not a problem because the sign of the differential of inputs to the f function doesn't influence #CVC.

Strict Calculation Algorithm of #CVC

Algorithm 1: #CVC calculation algorithm

i Expand DV to 80 steps in consideration of message expansion.
ii Evaluate #CVC of the DV by using the following "#CVC calculation core algorithm" (Algorithm 2).
iii Check the contradiction of the differential path for each steps in rounds 2-4 by using the following "Differential Path Confirmation Algorithm" (Algorithm 3).

Algorithm 2: #CVC calculation core algorithm We assume that chaining variables a_i ($i \in \{1, \ldots, k\}$) are satisfiable.

i Compress input DV by using "Strict Differential Bit Compression" shown in Section 5.2-(1).
ii Derive the differential of output in each step by using "DV expansion" shown in Section 5.2-(3).
iii Calculate #CVC from step $k+1$ to step 80 by using "Precise Counting Rules" shown in Section 5.2-(4). (#CVC is counted by each a_i ($i \in \{k-1, \ldots, 80\}$))
iv Remove conditions for prevent carries at a_{79}, a_{80} because it is possible to disregard them in collision search.
v Return the total #CVC of a_i for all $i \in \{k+1, \ldots, 80\}$.

Algorithm 3: Differential Path Confirmation Algorithm (Section 5.2-(2))

i Derive the message (without sign) differentials in each step by using LCs from (uncompressed) DV.
ii Derive the (without sign) differentials of output in each step by using compressed-DV.
iii Check $(\delta a_{i-5,j} \lll 30) \oplus (\delta a_{i-1,j} \lll 5) \oplus (\delta a_{i,j}) \oplus (\delta m_{i-1,j}) \oplus (\delta f_{i-1,j}) = 0$ in each bit position j.
When the above equation is TRUE, finish this check as SUCCESS.
iv Check the contradiction more strictly by considering the carry effect of each parameter.

6. Experimental Results

In this section, we show a computer experiment using our new algorithm proposed in Section5.2.

Table 6 One example of effective DV for various message modifiable steps

i	x_{i-1}	#	i	x_{i-1}	#	i	x_{i-1}	#	i	x_{i-1}	#
1	40000003		21	00000003	4	41	00000002	2	61	00000002	1
2	00000002		22	00000002	3	42	00000000	2	62	00000000	0
3	80000003		23	80000002	4	43	00000002	2	63	00000000	0
4	80000002		24	00000002	3	44	00000000	2	64	00000000	0
5	40000000		25	80000003	3	45	00000003	2	65	00000000	0
6	00000002		26	00000000	2	46	00000000	1	66	00000000	0
7	80000003		27	80000000	4	47	00000000	2	67	00000000	0
8	80000000		28	00000002	3	48	00000002	1	68	00000000	0
9	40000003		29	80000001	3	49	00000000	1	69	00000000	0
10	00000002		30	00000000	2	50	00000000	1	70	00000000	0
11	80000003		31	00000000	3	51	00000000	0	71	00000000	0
12	80000002		32	00000002	2	52	00000000	1	72	00000000	1
13	40000000		33	00000002	1	53	00000002	1	73	00000004	1
14	00000002		34	00000000	0	54	00000000	1	74	00000000	1
15	80000003		35	00000000	0	55	00000000	1	75	00000000	2
16	80000000		36	00000000	0	56	00000000	0	76	00000008	2
17	00000003		37	00000000	0	57	00000000	0	77	00000004	2
18	00000002		38	00000000	1	58	00000000	1	78	00000000	(2)
19	80000001		39	00000002	1	59	00000002	1	79	00000010	0
20	00000002		40	00000000	2	60	00000000	1	80	00000008	0

The elements in the columns in # mean the number of CVCs associated with each variable a_i in step i.
The number in the symbol '()' means the #CVC for second block.

Table 7 The number of remaining CVCs after message modification

MM until step	#CVC	MM until step	#CVC
21	70(72)	26	55(57)
22	67(68)	27	51(53)
23	63(65)	28	48(50)
24	60(62)	29	45(47)
25	57(59)	30	43(45)

6.1 Implementation of Our Algorithm

We implemented our algorithm on a PC. The running time to obtain #CVCs of all DVs is about 8 hours by using 4 threads in SUSE Linux 10.0 on AMD Opteron(tm) Processor Dual Core Model 275 (2.2GHz, Dual CPU).

The search procedure is described below.

DV search procedure for this experiment

(i) Execute (ii)-(iv) for all DVs lie within Rectangle Range.
(ii) Decide a DV of certain 16 steps in SHA-1 80 steps.
(iii) Execute "#CVC calculation algorithm" proposed in section 5.2 by using the DV decided in ii, and get #CVC of the DV.
(iv) Construct the actual differential path and the set of CVCs with no contradiction for each step in rounds 2-4.
(v) Sort all DVs in decreasing order and take the minimal DV.
(vi) Construct a differential path from the DV in round 1.

In this experiment, we tried to find the DV that has only less #CVCs.

Table 8 non-linear differential path and conditions for step 1-16

step	CVC
1	01100111010001010010001100000001
2	-01--...1....1.....0.110.1++1000
3	-.0++00..0...0....0..+.++00+-1--
4	100.111111.....+.--+0..10+11--0-
5	1111001001 0+0++..1.0+0.011-100+1
6	-01++000++10100000.000-..-+0-+11
7	111--0-0000+00+00...0.--.-1--110
8	+1+++++++++++++++.1..+.+00+0000
9	....110001001.11.001....1.1+0100
10	.+1010011000010001111.+.-+00..0-+
11	10.0...................1.0..+10-
12	+0.0...................+10-000-
13	.+.....................1.11--
14	0+10.....................1.-0..
15	-0.......................110-
16	-..1.....................01++

step	MM
1	+00--.................1...+0-0101
2	x..++.............0.11--0-011
3	0.....................11-0..100
4	..0+..................00+++00.1
5	00++0011..............+1-00+-
6	00++10..................1-100
7	0101..01..............-1110+-
8	x-.+..010.............xx+00-0
9	x+-+000.....0.............+...0
10	xxxx...................+x.-0-0
11	x.....................x0...0.
12	x..+..................++x....
13	x.-+...................x.-..--
14	..-x.....................-...
15	.....................+....--
16	xx.+..................---..+.

Notations are as follows;
"." : no condition, "0" : $0 \to 0$, "1" : $0 \to 0$,
"-" : $1 \to 0$, "+" : $0 \to 1$, "x" : $0 \to 1$ or $1 \to 0$.

6.2 Results and Consideration

We found several DVs that have less #CVCs and one of them is shown in Table 6. After that, we derive a differential path, CVC and MC in round 2 to 4, and 1. The results are shown in Table 9 and Table **??**, respectively. We note that it is confirmed all the MCs which can be expanded to the MC in Round 1 by message expansion has not contradiction each other.

By using our method, we can evaluate the number of CVCs precisely. Table 7 shows the number of essential CVCs for each cases that we can modify message up to step 21-30. For example, if we assume that we can modify messages up to step 25, the number of its essential CVCs is 57 (or 59) for 1st (or 2nd) block. However, we don't confirm the practicability of the message modifications for new DV. We can not say immediately that we can find a collision for SHA-1 with this number of CVCs. As we mentioned in the subsection 3.2, there are some trade-off relations (ex.message freedom.) For the practical collision search, we need to consider the remaining factors which we didn't discuss in this paper.

7. Conclusion and Future Works

In this paper, we proposed a new strict evaluation algorithm of #CVC from one DV. This algorithm can evaluate #CVC more strictly than the previous one. Our algorithm is used for finding much effective DVs.

Table 9 An Example of Difference Path, CVC and MC in Round2-4 about DV1

step	DP	CVC	MC
16	±31	$a_{16,31} \neq m_{20,29}$	
17	±0	$a_{17,0} = m_{20,30}$	
18	±1	$a_{18,2} = a_{17,2}$	
19	±31 ∓ 0	$a_{19,31} \neq m_{23,29}$, $a_{19,0} \neq m_{20,1}$, $a_{19,31} \neq m_{22,29}$, $a_{19,0} = m_{22,30}$, $a_{19,3} = a_{18,3}$, $a_{19,0} \neq a_{17,0}$ $a_{19,31} \neq m_{21,29}$, $a_{19,30} = a_{18,0}$	
20	±1	$a_{20,1} \neq m_{20,6}$, $a_{20,1} \neq m_{22,6}$, $a_{20,2} = a_{19,2}$, $a_{20,29} = a_{18,31}$, $a_{20,30} \neq a_{18,0}$	
21	±0	$a_{21,30} \neq a_{20,0}$, $a_{21,29} = a_{20,31}$, $a_{21,0} = m_{20,1}$, $a_{21,3} = a_{20,3}$	$m_{20,31} \neq m_{20,30}$
22	±1	$a_{22,1} = a_{20,1}$, $a_{22,30} \neq a_{20,0}$, $a_{22,3} = a_{21,3}$	$m_{21,6} \neq m_{20,1}$, $m_{21,6} \neq m_{21,5}$
23	±31 ± 1	$a_{23,1} = m_{22,0}$, $a_{23,31} = m_{22,31}$, $a_{23,30} = a_{22,0}$, $a_{23,3} = a_{22,3}$	$m_{22,0} = m_{20,1}$
24	±1	$a_{24,1} = a_{22,1}$, $a_{24,29} = a_{22,31}$, $a_{24,2} = a_{23,2}$	$m_{23,6} \neq m_{22,0}$, $m_{23,4} \neq m_{22,31}$
25	±31 ± 0	$a_{25,0} \neq m_{24,0}$, $a_{25,31} = m_{24,30}$, $a_{25,29} = a_{24,31}$	$m_{24,6} = m_{22,26}$, $m_{24,0} \neq m_{22,0}$, $m_{24,30} = m_{20,1}$ $m_{24,30} = m_{22,31}$
26		$a_{26,29} \neq a_{24,31}$, $a_{26,30} = a_{24,0}$	$m_{25,4} \neq m_{24,30}$, $m_{25,6} \neq m_{25,5}$, $m_{25,6} \neq m_{24,0}$, $m_{25,1} = m_{22,6}$, $m_{25,29} \neq m_{22,31}$, $m_{25,30} = m_{20,1}$
27	±31	$a_{27,31} = m_{26,31}$, $a_{27,30} = a_{26,0}$, $a_{27,3} = a_{26,3}$, $a_{27,29} = a_{26,31}$	$m_{26,1} = m_{24,0}$, $m_{26,1} \neq m_{26,0}$, $m_{26,29} \neq m_{22,31}$ $m_{26,31} = m_{24,30}$
28	±1	$a_{28,1} = m_{27,1}$, $a_{28,29} \neq a_{26,31}$, $a_{28,2} = a_{27,2}$	$m_{27,4} \neq m_{26,31}$, $m_{27,30} \neq m_{24,0}$
29	±31 ± 0	$a_{29,31} = m_{28,30}$, $a_{29,0} = m_{28,0}$, $a_{29,29} = a_{28,31}$	$m_{28,6} \neq m_{27,1}$, $m_{28,29} \neq m_{24,30}$, $m_{28,30} = m_{26,31}$ $m_{28,30} \neq m_{24,0}$
30		$a_{30,29} \neq a_{28,31}$, $a_{30,30} = a_{28,0}$	$m_{29,1} \neq m_{27,1}$, $m_{29,30} \neq m_{24,0}$, $m_{29,5} \neq m_{28,0}$, $m_{29,4} \neq m_{28,30}$
31		$a_{31,30} = a_{30,0}$, $a_{31,29} = a_{30,31}$, $a_{31,3} = a_{30,3}$	$m_{30,29} \neq m_{26,31}$, $m_{30,0} \neq m_{28,0}$
32	±1	$a_{32,1} = m_{31,1}$, $a_{32,3} = a_{31,3}$	$m_{31,30} \neq m_{28,0}$
33	±1	$a_{33,1} = m_{32,1}$	$m_{32,6} \neq m_{31,1}$, $m_{32,30} \neq m_{28,0}$, $m_{32,29} \neq m_{28,30}$
34			$m_{33,1} \neq m_{31,1}$, $m_{33,6} \neq m_{32,1}$, $m_{33,30} \neq m_{28,0}$ $m_{33,29} \neq m_{28,30}$
35			$m_{34,1} \neq m_{32,1}$
36			
37			
38		$a_{38,3} \neq a_{37,3}$	
39	±1	$a_{39,1} = m_{38,1}$	
40		$a_{40,31} \neq a_{38,1}$, $a_{40,3} \neq a_{39,3}$	$m_{39,6} \neq m_{38,1}$
41	±1	$a_{41,1} = m_{38,1}$, $a_{41,31} \neq a_{40,1}$	
42		$a_{42,31} \neq a_{40,1}$, $a_{42,3} \neq a_{41,3}$	$m_{41,6} \neq m_{38,1}$
43	±1	$a_{43,1} = m_{38,1}$, $a_{43,31} \neq a_{42,1}$	
44		$a_{44,31} \neq a_{42,1}$, $a_{44,2} \neq a_{43,2}$	$m_{43,6} \neq m_{38,1}$
45	±0	$a_{45,0} \neq m_{44,0}$, $a_{45,31} \neq a_{44,1}$	$m_{44,0} \neq m_{38,1}$
46		$a_{46,30} \neq a_{44,0}$	$m_{45,6} \neq m_{45,5}$, $m_{45,6} \neq m_{38,1}$
47		$a_{47,30} \neq a_{46,0}$, $a_{47,3} \neq a_{46,3}$	$m_{46,1} \neq m_{46,0}$, $m_{46,1} \neq m_{38,1}$
48	±1	$a_{48,1} = m_{47,1}$	$m_{47,30} = m_{38,1}$
49		$a_{49,31} \neq a_{47,1}$	$m_{48,6} \neq m_{47,1}$, $m_{48,30} = m_{38,1}$
50		$a_{50,31} \neq a_{49,1}$	$m_{49,1} \neq m_{47,1}$, $m_{49,30} = m_{38,1}$
51			
52		$a_{52,3} \neq a_{51,3}$	
53	±1	$a_{53,1} = m_{52,1}$	
54		$a_{54,31} \neq a_{52,1}$	$m_{53,6} \neq m_{52,1}$
55		$a_{55,31} \neq a_{54,1}$	$m_{54,1} \neq m_{52,1}$
56			
57			
58		$a_{58,3} = a_{57,3}$	
59	±1	$a_{59,1} = m_{58,1}$	
60		$a_{60,3} = a_{59,3}$	$m_{59,6} \neq m_{58,1}$
61	±1	$a_{61,1} = m_{58,1}$	
62			$m_{61,6} \neq m_{58,1}$
63			$m_{62,1} \neq m_{58,1}$
64			
65			
66			
67			
68			
69			
70			
71			
72		$a_{72,4} = a_{71,4}$	
73	±2	$a_{73,2} = m_{72,2}$	
74		$a_{74,0} = a_{72,2}$	$m_{73,7} \neq m_{72,2}$
75		$a_{75,5} = a_{74,5}$, $a_{75,0} = a_{74,2}$,	$m_{74,2} \neq m_{72,2}$
76	±3	$a_{76,4} = a_{75,4}$, $a_{76,3} = m_{75,3}$	$m_{75,0} \neq m_{72,2}$
77	±2	$a_{77,1} = a_{75,3}$, $a_{77,2} = m_{76,2}$	$m_{76,0} \neq m_{72,2}$, $m_{76,8} \neq m_{75,3}$
78		$(a_{78,0} = a_{76,2})$, $(a_{78,1} = a_{77,3})$	$m_{77,3} \neq m_{75,3}$, $m_{77,7} \neq m_{76,2}$, $m_{77,0} \neq m_{72,2}$
79	±4		$m_{78,2} \neq m_{76,2}$, $m_{78,1} \neq m_{75,3}$
80	±3		$(m_{79,1} \neq m_{75,3})$, $(m_{79,0} \neq m_{76,2})$, $m_{79,9} \neq m_{78,4}$

The CVCs and MCs appeared in step 16-20 on the above Table are a part of the necessary conditions for the collision search, on which the CVCs in step 21-80 depend. (Conditions for prevent carries at step 79 and 80 has been removed [4].)

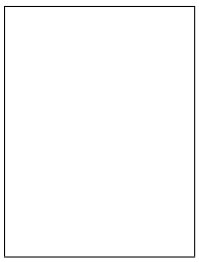
We propose a strict evaluation method on (i) by improving the previous method. We implemented our algorithm on PC, and executed computer experiment. As a result, we confirmed our algorithm worked successfully, and several candidates of DVs are found that may be used to efficient collision search.

In order to find a collision message pair on SHA-1, studies of effectiveness on found DVs are marked as future works. The studies include consideration on applicability of message modification, other differential paths and etc,... Our algorithm is extendable to other hash functions, i.e. SHA-256, SHA-512.

References

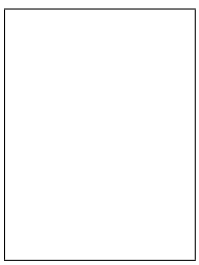
[1] NIST, "Secure hash standard," Federal Information Processing Standard, National Institute of Standards and Technology, April 1995.
[2] E. Biham, R. Chen, A. Joux, P. Carribault, C. Lemuet, and W. Jalby, "Collisions in SHA-0 and reduced SHA-1," EUROCRYPT2005, pp.36–57, International Association for Cryptologic Research (IACR), May 2005.
[3] A. Joux and T. Peyrin, "Hash functions and the (amplified) boomerang attack," CRYPTO2007, pp.244–263, International Association for Cryptologic Research (IACR), August 2007.
[4] X. Wang, Y.L. Yin, and H. Yu, "Finding collisions in the full SHA-1," CRYPTO2005, pp.17–36, International Association for Cryptologic Research (IACR), August 2005.
[5] X. Wang, A.C. Yao, and F. Yao, "Cryptanalysis on SHA-1 hash function," CRYPTOGRAPHIC HASH WORKSHOP, National Institute of Standards and Technology, November 2005.
[6] X. Wang, "Cryptanalysis of hash functions and potential dangers," Invited Talk at the Cryptographer's Track at RSA Conference 2006, RSA, February 2006.
[7] X. Wang and H. Yu, "How to break MD5 and other hash functions," EUROCRYPT2005, pp.19–35, International Association for Cryptologic Research (IACR), May 2005.
[8] F. Chabaud and A. Joux, "Differential collisions in SHA-0," CRYPTO'98, pp.56–71, International Association for Cryptologic Research (IACR), August 1998.
[9] X. Wang, "The collision attack on SHA-0," available at http://www.infosec.sdu.edu.cn/people/wangxiaoyun. htm, 1997.
[10] C.D. Cannière and C. Rechberger, "Finding SHA-1 characteristics:

General results and applications," ASIACRYPT2006, International Association for Cryptologic Research (IACR), December 2006.
[11] P. Hawkes, M. Paddon, and G. Rose, "Automated search for round 1 differentials for SHA-1: Work in progress," NIST SECOND CRYPTOGRAPHIC HASH WORKSHOP, National Institute of Standards and Technology, August 2006.
[12] J. Yajima, Y. Sasaki, Y. Naito, T. Iwasaki, T. Shimoyama, N. Kunihiro, and K. Ohta, "A new strategy for finding a differential path of SHA-1," ACISP2007, pp.45–58, International Association for Cryptologic Research (IACR), July 2007.
[13] Y. Naito, Y. Sasaki, T. Shimoyama, J. Yajima, N. Kunihiro, and K. Ohta, "Improved collision search for SHA-0," ASIACRYPT2006, pp.21–36, International Association for Cryptologic Research (IACR), December 2006.
[14] M. Sugita, M. Kawazoe, and H. Imai, "Gröbner basis based cryptanalysis of SHA-1," Fast Software Encryption 2007, IACR, March 2007.
[15] C.D. Cannière, F. Mendel, and C. Rechberger, "On the full cost of collision search for SHA-1," ECRYPT Hash Workshop, ECRYPT Network of Excellence in Cryptology, May 2007.
[16] "SHA-1 collision search graz," August 2007. IAIK Krypto Group, Graz University of Technology, http://boinc.iaik.tugraz.at/sha1_coll_search/.

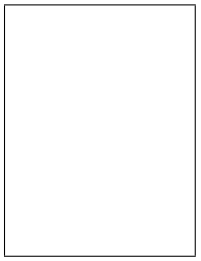


Jun Yajima received his B.E. and M.E. degrees in information and system engineering in 1997 and 1999, respectively, from Chuo University. Since 1999, he has been a researcher at FUJITSU LABORATORIES LTD. His current research interest includes information security and cryptography. He was awarded the SCIS2007 paper prize.

Terutoshi Iwasaki received his B.E. and M.E. degrees in information and system engineering in 2005 and 2007, respectively, from Chuo University. Since 2007, he has been a system engineer at Nomura Research Institute, Ltd. His current research interest includes information security and cryptography.



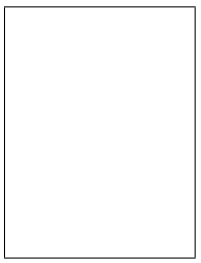
Yusuke Naito XXX



Yu Sasaki XXX

Takeshi Shimoyama received his B.S., M.S. degrees in mathematics from Yokohama City University. in 1989 and 1991, respectively, and D.E. degrees in information and system engineering from Chuo University in 2000. He is a research engineer of FUJITSU LABORATORIES Ltd from 1991. He joined the Research Project of Info Communication Security under Telecommunications Advancement Organization of Japan from 1996 to 1998. His current research interests are in cryptanalysis and information security. He was awarded SCIS paper prize in 1997, IWSEC paper prize in 2007, and OHM Technology Award in 2007. He attained the world record of an integer factoring by GNFS in 2006.

Thomas Peyrin received his M.S. degree in computer science from Ecole Polytechnique in France in 2005. He is a PhD student at University of Versailles in France from 2005 and was a JSPS Fellow at AIST in Japan from September 2007 to March 2008. His current research interests are cryptanalysis and design of cryptographic hash functions. He was awarded Asiacrypt 2007 best paper award.



Noboru Kunihiro received his B. E., M. E. and Ph. D. in mathematical engineering and information physics from the University of Tokyo in 1994, 1996 and 2001, respectively. He is an Associate Professor of the University of Tokyo. He was a researcher of NTT Communication Science Laboratories from 1996 to 2002. He was a associate professor of the University of Electro-Communications from 2002 to 2008. His research interest includes cryptography, information security and quantum computations. He was awarded the SCIS'97 paper prize.

Kazuo Ohta received his B.S., M.S., and Dr.S. degrees from Waseda University, Tokyo, Japan, in 1977, 1979, and 1990, respectively. He has been a professor at the University of Electro-Communications since 2001. He was a researcher at NTT Laboratories between 1979 and 2001. He is presently engaged in research on information security. Dr. Ohta is a member of the International Association for Cryptologic Research,, IEICE, IPSJ and IEEE.