Pushing the Limit of Memory-efficient Collision Attack Framework for SHA-2

Yingxin Li[1], Fukang Liu[2], Gaoli Wang[3,5], and Jiali Shi[4]

[1] School of Cryptology, East China Normal University, Shanghai, China
yxli@sc.ecnu.edu.cn,
[2] Institute of Science Tokyo, Tokyo, Japan
liu.f.ad@m.titech.ac.jp
[3] Shanghai Key Laboratory of Trustworthy Computing, Software Engineering Institute, East China Normal University, Shanghai, China
glwang@sei.ecnu.edu.cn
[4] School of Computer Science and Technology, Henan Polytechnic University, Jiaozuo, China,
jiali@hpu.edu.cn
[5] Advanced Cryptography and System Security Key Laboratory of Sichuan Province, Chengdu 610103, China

Abstract. The SHA-2 family hash is standardized by NIST and mainly includes two variants, SHA-256 and SHA-512. Due to its widespread deployment, its security has attracted continuous attention from various parties. Although Li *et al.* have developed open-source SAT/SMT-based tools and proposed new memory-efficient collision attack frameworks for SHA-2 in recent two years, practical collision attacks are only achieved for 31-step SHA-256 and 29-step SHA-512, respectively. To push the limit of such an attack framework for SHA-2, we carefully investigate existing strategies to choose message differences used in 38/39-step semi-free-start collision attacks. We found that by selecting message words $(W_{4+i}, \ldots, W_{8+i}, W_{12+i}, W_{13+i}, W_{20+i}, W_{22+i})_{0\leq i\leq 3}$ to inject differences, and employing the open-source SAT/SMT-based automated tools to search for the corresponding differential characteristics, notable improvement can be achieved for practical and theoretical collision attacks. Specifically, the first practical collision attacks on 35-step SHA-256 and SHA-512 can be achieved for $i = 0$, improving the best practical collision attacks on SHA-256 and SHA-512 by 4 and 6 steps, respectively. When $i \in \{1, 2\}$, theoretical collision attacks on both SHA-256 and SHA-512 can reach up to 36/37 steps. We have also tried collision attack up to 38 steps by setting $i = 3$, but the uncontrolled differential probability is too low to be used for effective attacks.

Keywords: Hash functions · SHA-2 · Collision attack · Message difference.

1 Introduction

The MD-SHA hash family refers to a class of cryptographic hash functions constructed based on the Merkle-Damgård (MD) [1,16] structure. Typical represen-

tatives include MD5, MD4, RIPEMD, RIPEMD-128, RIPEMD-160, SHA-0, SHA-1, and SHA-2. In 2005, Wang *et al.* used new techniques to demonstrate that MD4 [21], MD5 [23], RIPEMD [21], SHA-0 [24], and SHA-1 [22] cannot resist collision attacks, and the security of MD-SHA hash family was seriously questioned. Subsequently, RIPEMD-128 was also shown to be weak against the semi-free-start (SFS) collision attacks [5] in 2013, and its practical collision attack reached 54 steps [12]. Although SHA-1 was proven to be vulnerable to collision attacks in 2005, a practical full-round collision attack on SHA-1 was not achieved until 2017 [20]. However, when this technique is applied to RIPEMD-160 and SHA-2, the increased complexity of their round functions significantly limits the effectiveness of existing attack strategies. Although some progress has recently been made in collision attacks on RIPEMD-160 [6,11,26], it remains secure for the full number of rounds. As for SHA-2, which is currently the most widely deployed cryptographic hash standard in industry, an in-depth analysis of its collision resistance is of great theoretical and practical importance.

For collision attacks on SHA-2, before 2008, many different researchers published results on collision attacks [4,15,18,19]. However, due to the lack of effective tools, the progress was relatively slow. The change came at ASIACRYPT 2011 [13], where Mendel *et al.* developed the first automated tool to search for collision-generating differential characteristics for SHA-2 based on the guess-and-determine (GnD) technique. Using this tool, they found practical collision message pairs for 27-step SHA-256, as well as practical SFS collision message pairs for 32-step SHA-256, by choosing different message words to inject differences. Subsequently, a series of collision attacks on SHA-2 were obtained by choosing different message differences, including the practical collision attack on 28-step SHA-256 [14], theoretical collision attack on 31-step SHA-256 [14], practical SFS collision attack on 38-step SHA-256 [14], practical collision attacks on 38/39-step SHA-512 [2,3], and practical collision attack on 27-step SHA-512 [2].

After Liu *et al.* made a breakthrough by inventing the first automatic MILP-based method to search for collision-generating signed differential characteristics at EUROCRYPT 2023 [11], Li *et al.* [7] further proposed at EUROCRYPT 2024 the first SAT/SMT-based automated tool to search for collision-generating differential characteristics for SHA-2, achieving the first practical SFS collision attack on 39-step SHA-256, improved theoretical collision attack on 31-step SHA-512, and practical free-start (FS) collision attack on 40-step SHA-256. They subsequently optimized the 31-step SHA-256 differential characteristics and proposed a memory-efficient collision attack framework at ASIACRYPT 2024 [8], achieving the first practical collision attack on 31-step SHA-256. Furthermore, Li *et al.* refined the SAT/SMT-based automated tool by introducing marked variables to capture the number of 2-bit conditions, leading to the first practical collision attack on 29-step SHA-512 and improved collision attack on 31-step SHA-512 at CRYPTO 2025 [9]. A summary of these results is shown in Table 1.

Collision attacks on SHA-2 have continuously achieved significant breakthroughs. By exploiting the GnD-based automated tools [2,13,14], two-block approach [14], Meet-in-the-Middle (MitM) techniques [14], memory-efficient colli-

sion attack framework [8] and SAT/SMT-based automated tools [7,9], researchers have successively realized theoretical or practical collision attacks on reduced-round versions ranging from 27 to 31 steps. However, due to the complex interdependencies inherent in the message expansion of SHA-2, the problem of choosing appropriate message differences and controlling the propagation of differences in internal states remains a key bottleneck that limits further extension of attacks on more rounds.

Our contributions. We briefly summarize our contributions below and our results are summarized in Table 1.

1. We revisit practical collision attacks on 27/28/29-step SHA-256 [9,13,14] in the literature and observe that the expanded message words can be divided into three parts, i.e., $t = t_0 \| t_1 \| t_2$, where the step ranges covered by t_0 and t_2 contain no differences, while the step range covered by t_1 contains differences, and a local collision can be formed within the steps spanned by t. When the number of steps spanned by t_1 is fixed, adjusting t_0 and t_2 can significantly affect the number of attackable steps. Based on the SAT/SMT-based automated tool proposed by Li *et al.* [7,9], we compare multiple candidates of message differences by searching for the corresponding differential characteristics that are more suitable for collision attacks. Finally, we choose the message words $(W_{4+i}, \ldots, W_{8+i}, W_{12+i}, W_{13+i}, W_{20+i}, W_{22+i})_{0 \le i \le 3}$ to inject differences for the most effective collision attacks. In particular, the case of $i = 4$ has been used in the SFS collision attack on 39-step SHA-2 [2,7].
2. Based on the above message word selection strategy, we perform collision attacks on SHA-256 and SHA-512 and obtain the following results:
   - By introducing differences into the expanded message words $(W_4, \ldots, W_8, W_{12}, W_{13}, W_{20}, W_{22})$, we achieve the first practical collision attack on 35-step SHA-256, improving the previous best practical collision attack by 4 steps. Such a selection of message differences allows us to achieve the first practical collision attack on 35-step SHA-512, improving the previous best practical collision attack by 6 steps.
   - By introducing differences into the expanded message words $(W_5, \ldots, W_9, W_{13}, W_{14}, W_{21}, W_{23})$, we also achieve the first theoretical collision attacks on 36-step SHA-256 and SHA-512 with time complexity 2^{71} and 2^{136}, respectively.
   - Pushing the limit of our strategy, by introducing differences into the expanded message words $(W_6, \ldots, W_{10}, W_{14}, W_{15}, W_{22}, W_{24})$, the theoretical collision attack on SHA-256 and SHA-512 can reach up to 37 steps.

We also have tried to extend the attack to 38 steps by setting $i = 3$, but the uncontrolled differential probability is too low to mount an effective attack.

Outline. The remainder of this paper is organized as follows. A description of SHA-2 is given in Sect. 2. In Sect. 3, we review the memory-efficient collision

attack framework and explain how new message words are selected. The new 35-step practical collision attack on SHA-2 is presented in Sect. 4. Sect. 5 presents the 36/37-step collision attacks on SHA-2. Finally, we conclude this paper in Section 6.

Table 1. Summary of (semi-free-start) collision attacks on SHA-2.

Version	Attack type	Steps	Time	Memory	References
SHA-256	collision	28	*practical*	–	[14]
		29	*practical*	–	[9]
		31	$2^{65.5}$	2^{34}	[14]
		31	$2^{49.8}$	2^{48}	[7]
		31	*practical*	$2^{19.8}$	[8]
		35	***practical***	$\mathbf{2^{29.1824}}$	**Sect. 4**
		36	$2^{94.4}$	$2^{12.6}$	[25]
		36	$\mathbf{2^{71}}$	$\mathbf{2^{38.585}}$	**Sect. 5.1**
		37	$2^{119.1}$	$2^{39.9}$	[25]
		37	$2^{122.2}$	$2^{34.9}$	[25]
		37	$\mathbf{2^{81}}$	$\mathbf{2^{53.047}}$	**Sect. 5.2**
	SFS collision	38	*practical*	–	[14]
		39	*practical*	–	[7]
SHA-512	collision	27	*practical*	–	[2]
		28	*practical*	–	[7]
		29	*practical*	–	[9]
		31	$2^{115.6}$	$2^{77.3}$	[7]
		31	$2^{97.3}$	$2^{35.2}$	[8]
		31	$2^{85.5}$	$2^{44.4}$	[9]
		35	***practical***	–	**Sect. 4**
		36	$\mathbf{2^{136}}$	$\mathbf{2^{87}}$	**Sect. 5.1**
		37	$\mathbf{2^{185}}$	$\mathbf{2^{132}}$	**Sect. 5.2**
	SFS collision	38	*practical*	–	[3]
		39	*practical*	–	[2]

2 Preliminaries

2.1 Notations

For a better understanding of this paper, we introduce the following notations.

1. $\boxplus$ and $\boxminus$ represent modular addition and modular subtraction on 32/64 bits, respectively.
2. $\gg$, $\ggg$, $\oplus$, $\neg$, $\vee$ and $\wedge$ represent *shift right*, *rotate right*, *exclusive or*, *not*, *or*, and *and*, respectively.

3. $x[i]$ denotes the i-th bit of x and $x[0]$ is the least significant bit.
4. δx denotes the modular difference, i.e., $\delta x = x' \boxminus x$.
5. ∇x denotes the signed difference between x' and x. We use the same notation as in [10,11], i.e.,

$$\nabla x[i] = \begin{cases} \texttt{n} & (x[i] = 0, x'[i] = 1) \\ \texttt{u} & (x[i] = 1, x'[i] = 0) \\ \texttt{=} & (x[i] = x'[i]) \\ \texttt{0} & (x[i] = x'[i] = 0) \\ \texttt{1} & (x[i] = x'[i] = 1) \end{cases} \tag{1}$$

6. $M = (m_0, m_1, \ldots, m_{15})$ and $M' = (m'_0, m'_1, \ldots, m'_{15})$ represent two message blocks.

Definition 1. *[11] The signed difference ∇x is said to be an expansion of the modular difference δx only when ∇x corresponds to the modular difference δx.*

Definition 2. *[11] The hamming weight of the signed difference ∇x is denoted by $\boldsymbol{H}(\nabla x)$, where $\boldsymbol{H}(\nabla x)$ is the number of indices i such that $\nabla x[i] \in \{\boldsymbol{n}, \boldsymbol{u}\}$.*

For example, let

$$\nabla x_0 = [\texttt{==== nu== ==== ==== ==== ==== ==== ====}],$$
$$\nabla x_1 = [\texttt{==== =n== ==== ==== ==== ==== ==== ====}].$$

Then, both ∇x_0 and ∇x_1 are the expansions of $\delta x = 2^{26}$. Moreover, we have $\mathbf{H}(\nabla x_0) = 2$ and $\mathbf{H}(\nabla x_1) = 1$. As each signed difference corresponds to a unique modular difference, for convenience, when computing $\delta x \boxplus \delta y$ for a given $(\nabla x, \nabla y)$, we also simply denote $\delta x \boxplus \delta y$ by $\nabla x \boxplus \nabla y$. For the above example, we have $\nabla x_0 \boxplus \nabla x_1 = 2^{27}$.

2.2 Description of SHA-2

The SHA-2 family is a series of hash functions standardized by NIST as part of the Secure Hash Standard (SHS) [17]. SHA-2 adopts the well-known Merkle-Damgård construction [1,16], and its compression functions employ the Davies-Meyer construction. We use n to represent the word size of SHA-2. The compression function of SHA-2, denoted by $H(CV, M)$, takes a $16n$-bit message block M and an $8n$-bit chaining value CV as inputs, and produces an $8n$-bit output. In SHA-2, the two main versions, SHA-256 and SHA-512, use 32-bit and 64-bit word sizes, respectively (i.e., $n = 32$ or 64).

For our collision attacks on SHA-2, we aim to find two message blocks (M_0, M_1) and (M_0, M'_1) such that

$$H(H(CV_0, M_0), M_1) = H(H(CV_0, M_0), M'_1)$$

where $M_1 \neq M'_1$ and CV_0 is a prefixed constant.

The compression functions H are computed through iterative updates to internal states. The number of steps, which is denoted by r, is 64 for SHA-256 and

80 for SHA-512. In the following, we provide a brief overview of the compression functions H. They consist of two main parts: the message expansion and the state update transformation. A complete description of SHA-2 is given in [17].

Message Expansion. The 512-bit message block for SHA-256 and the 1024-bit message block for SHA-512 are divided into 16 message words of sizes 32 bits and 64 bits, respectively, which are denoted by $(m_0, \ldots, m_{15})$. Then, the 16 message words are expanded to r expanded message words W_i, i.e., $W_0, W_1, \ldots, W_{r-1}$:

$$W_i = \begin{cases} m_i & 0 \le i \le 15, \\ \sigma_1(W_{i-2}) \boxplus W_{i-7} \boxplus \sigma_0(W_{i-15}) \boxplus W_{i-16} & 16 \le i \le r-1. \end{cases}$$

The functions $\sigma_0(x)$ and $\sigma_1(x)$ in SHA-256 are given by

$$\sigma_0(x) = (x \ggg 7) \oplus (x \ggg 18) \oplus (x \gg 3),$$
$$\sigma_1(x) = (x \ggg 17) \oplus (x \ggg 19) \oplus (x \gg 10).$$

The functions $\sigma_0(x)$ and $\sigma_1(x)$ in SHA-512 are given by

$$\sigma_0(x) = (x \ggg 1) \oplus (x \ggg 8) \oplus (x \gg 7),$$
$$\sigma_1(x) = (x \ggg 19) \oplus (x \ggg 61) \oplus (x \gg 6).$$

State update transformation. We utilize the alternate description for the state update of SHA-256 and SHA-512, as illustrated in Figure 1.

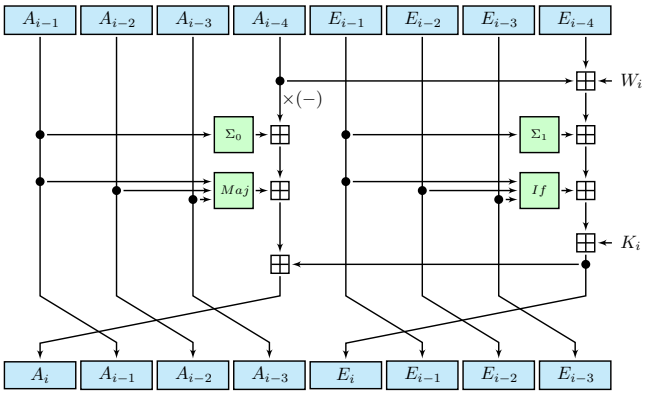

Fig. 1. The state update transformation of SHA-2 [7].

The state update transformation starts from a 256-bit (resp. 512-bit) chaining value $iv = (A_{-1}, \ldots, A_{-4}, E_{-1}, \ldots, E_{-4})$ for SHA-256 (resp. SHA-512), and updates it by applying the step function r times. In each step $i = 0, \ldots, r-1$,

one expanded message word W_i is used to compute the two state words E_i and A_i as follows, where K_i is a predefined constant and can be referred to [17].

$$E_i = A_{i-4} \boxplus E_{i-4} \boxplus \Sigma_1(E_{i-1}) \boxplus \mathrm{IF}(E_{i-1}, E_{i-2}, E_{i-3}) \boxplus K_i \boxplus W_i,$$
$$A_i = E_i \boxminus A_{i-4} \boxplus \Sigma_0(A_{i-1}) \boxplus \mathrm{MAJ}(A_{i-1}, A_{i-2}, A_{i-3}).$$

Both SHA-256 and SHA-512 utilize the same Boolean functions IF and MAJ, as defined below:

$$\mathrm{IF}(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus z,$$
$$\mathrm{MAJ}(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z).$$

However, the linear functions Σ_0 and Σ_1 are different for SHA-256 and SHA-512. For SHA-256, they are defined below:

$$\Sigma_0(x) = (x \ggg 2) \oplus (x \ggg 13) \oplus (x \ggg 22),$$
$$\Sigma_1(x) = (x \ggg 6) \oplus (x \ggg 11) \oplus (x \ggg 25).$$

For SHA-512, they are defined below:

$$\Sigma_0(x) = (x \ggg 28) \oplus (x \ggg 34) \oplus (x \ggg 39),$$
$$\Sigma_1(x) = (x \ggg 14) \oplus (x \ggg 18) \oplus (x \ggg 41).$$

After the last step of the state update transformation, the previous chaining value is added to the output of the state update. The result of this feed-forward sum is the chaining value h:

$$h = (A_{r-1} \boxplus A_{-1}, \ldots, A_{r-4} \boxplus A_{-4}, E_{r-1} \boxplus E_{-1}, \ldots, E_{r-4} \boxplus E_{-4}).$$

3 A Novel Strategy for Selecting Non-Zero Message Differences

The collision attack based on the two-block approach plays a crucial role in collision attacks on the MD-SHA family of hash functions. In some milestone studies, for example, Wang *et al.* utilized two-block approach to achieve the first practical collision attack on full MD5 [23] and a theoretical collision attack on full SHA-1 [22], and Stevens *et al.* achieved the first colliding message pair for full SHA-1 using the two-block method [20]. This breakthrough result clearly demonstrates the effectiveness of the two-block approach in collision attacks on MD-SHA.

As for SHA-2, Mendel *et al.* [14] used a two-block approach combined with a MitM technique to turn a SFS collision into a collision on 31-step SHA-256. Subsequently, Li *et al.* [8] proposed a memory-efficient collision attack framework that replaces the MitM technique and achieved the first practical collision attack on 31-step SHA-256. However, [14] and [8] both used the same message words to inject differences. At CRYPTO 2025, Li *et al.* [9] selected new message words

to achieve a practical collision attack on 29-step SHA-512. This result provided us with new inspiration: is it possible to further increase the number of steps of SHA-2 collision attacks by selecting new message words? In other words, can a semi-free-start collision attack with more steps be turned into a collision attack? To achieve this goal, we first review the memory-efficient two-block collision attack framework and extract the requirements that such attacks need to satisfy. The basic steps are outlined below.

Step 1: **Pre-processing Phase.** Find valid solutions of $(A_1, \ldots, A_s, E_5, \ldots, E_s, W_9, ..., W_s)_{9\le s\le 15}$. Then choose N_{start} solutions with distinct $(A_1, \ldots, A_4, E_5, \ldots, E_8)$. Finally, according to the state (A_i, E_i) update function and each starting point $(A_1, \ldots, A_4, E_5, \ldots, E_8)$, first exhaust all possible (W_8, E_4) to obtain A_0. Then exhaust all possible (W_7, E_3) to obtain A_{-1} from each tuple (W_8, E_4, A_0). Based on such a process, we can obtain all valid tuples $(A_{-1}, \ldots, A_s, E_3, \ldots, E_s, W_7, \ldots, W_s)_{9\le s\le 15}$, and store them in a table denoted by $\mathtt{TAB}_2$.

Step 2: **Matching Phase.** Try an arbitrary M_0, and get the corresponding chaining input $(A_{-4}, A_{-3}, A_{-2}, A_{-1}, E_{-4}, E_{-3}, E_{-2}, E_{-1})$ to match A_{-1} from $\mathtt{TAB}_2$. Once a match is found, perform the on-the-fly detection of the validity of A_{-4}, A_{-3}, and A_{-2}, which is indeed to test the conditions on (W_4, W_5, W_6). More details can be found in [8].

Step 3: **Fulfill the Conditions on** $(A_i, E_i, W_i)_{i\ge 16}$**.** Up to this step, $(W_i)_{0\le i\le s}$ have been fixed. Use the degrees of freedom in $(W_i)_{s\le i\le 15}$ to fulfill the remaining uncontrolled conditions on $(A_i, E_i, W_i)_{i\ge 16}$. If it fails, go to Step 2.

In the **Pre-processing Phase**, suppose that there are n_1 and n_2 bit conditions on W_8 and E_4, respectively, and the expected number of (A_0, E_4, W_8) valid pairs is $2^{n-n_1-n_2}$. Assuming that W_7 and E_3 have n_3 and n_4 conditions, respectively, and expect to obtain $2^{n-n_3-n_4}$ possible (A_{-1}, E_3, W_7) from each valid (A_0, E_4, W_8). Therefore, it is expected to obtain about $N_{start} \cdot 2^{2n-n_1-n_2-n_3-n_4}$ valid tuples of $(A_{-1}, \ldots, A_s, E_3, \ldots, E_s, W_7, \ldots, W_s)_{9\le s\le 15}$ from N_{start} starting points.

Denote the time complexity to obtain one starting point by T_{sat}. Then, the time complexity of the pre-processing phase denoted by T_{pre} can be estimated as

$$T_{pre} = N_{start} \times (T_{sat} + min(2^{n-n_1}, 2^{n-n_2}) + 2^{n-n_1-n_2} \times min(2^{n-n_3}, 2^{n-n_4})).$$

The memory complexity denoted by M is $M = N_{start} \cdot 2^{2n-n_1-n_2-n_3-n_4}$.

In the **Matching Phase**, denote the number of all conditions on (W_4, W_5, W_6) by N_{pro}, and the success probability of Step 3 by β. Then, when an A_{-1} is matched, the probability to output a colliding message pair is

$$P = 2^{-N_{pro}-\beta}$$

for each valid $(CV_1, A_0, \ldots, A_s, E_3, \ldots, E_s, W_7, \ldots, W_s)_{9\le s\le 15}$.

If N_{test} different M_0 are tested, we then expect to test on average

$$N_{test} \times \frac{N_{start} \cdot 2^{2n-n_1-n_2-n_3-n_4}}{2^n}$$

different values of $(CV_1, A_0, \ldots, A_s, E_3, \ldots, E_s, W_7, \ldots, W_s)_{9\leq s\leq 15}$ for Step 2 and 3. Hence, the following condition should hold in order to find a collision:

$$N_{test} \times \frac{N_{start} \cdot 2^{2n-n_1-n_2-n_3-n_4}}{2^n} \geq \frac{1}{P} = 2^{N_{pro}+\beta}.$$

In other words, this is equivalent to

$$N_{test} \geq \frac{2^{n+N_{pro}+\beta}}{N_{start} \cdot 2^{2n-n_1-n_2-n_3-n_4}} = \frac{2^{N_{pro}+\beta+n_1+n_2+n_3+n_4-n}}{N_{start}}.$$

Therefore, the time complexity of Step 2-3 is estimated as

$$T_{match} = \frac{2^{N_{pro}+\beta+n_1+n_2+n_3+n_4-n}}{N_{start}} + 2^{N_{pro}+\beta}.$$

The total time complexity of the memory-efficient collision attack framework is

$$T_{pre} + T_{match}$$

and the memory complexity is

$$N_{start} \cdot 2^{2n-n_1-n_2-n_3-n_4}.$$

The time complexity of the memory-efficient collision attack framework, $T_{\text{pre}} + T_{\text{match}}$, mainly depends on the sizes of N_{pro} and β. In other words, the bit conditions on (W_4, W_5, W_6) and $(A_i, E_i, W_i)_{i\geq 16}$ will affect the overall time complexity of the collision attack.

To select appropriate non-zero message words and further increase the number of steps achieved in SHA-2 collision attacks, it is necessary to conduct a detailed analysis of the expanded message words. However, due to the complex interdependencies among the expanded message words in SHA-2, determining suitable non-zero message words becomes a challenging task.

In the previous selection of expanded message words, the main idea is to search for the size t of a relatively complex local collision within the message expansion. To facilitate better understanding, the local collision size t is divided into three parts, i.e., $t = t_0 \| t_1 \| t_2$, where the segment in which non-zero message differences appear in the intermediate steps is defined as t_1; the segment in which these differences are cancelled over as many consecutive backward steps as possible is defined as t_2; and the segment in the forward direction, which must contain no differences at all, is defined as t_0. As the value of t_1 increases, the number of steps achieved in SFS collision attacks on SHA-2 also increases significantly. For example, the researchers chose $t_1 = 11$ to achieve 27/28/29-step practical collision attacks [2,9,13,14], $t_1 = 14$ for a 31-step collision attack [7,8,14], $t_1 = 15$

for a 32-step SFS collision attack [13], $t_1 = 18$ for a 38-step SFS collision attack [3,14], and $t_1 = 19$ for a 39-step SFS collision attack [2,7]. However, among these results, when t_1 is fixed to 11, 27/28/29-step practical collision attacks were achieved by adjusting t_0 and t_2. This indicates that not only t_1 but also t_0 and t_2 can influence the number of steps reached in SHA-2 collision attacks.

To provide a clearer understanding, we analyze in detail the three different expanded message-word selection strategies corresponding to $t_1 = 11$.

27-step Collision Attack on SHA-2 for $t_1 = 11$ [2,13]. In the 27-step SHA-2 collision attack, $t_0 = 7$, spanning over steps 0–6. Similarly, $t_2 = 9$, spanning over steps 18–26. Within $t_1 = 11$, spanning over steps 7–17, where the expanded message words $(W_7, W_8, W_{12}, W_{15}, W_{17})$ are selected to inject differences, eventually forming the 27-step (steps 0-26) SHA-2 collision attack, as shown in Figure 2. Although the middle steps in t_1 with differences can induce differences over steps 7–13 in E_i and steps 7–9 in A_i, a substantial amount of freedom is still required to satisfy all the conditions imposed by the differential characteristic. In particular, when performing message modifications, all bit conditions after step 16 must be satisfied. However, in the 27-step collision attack, only the bit conditions on W_{17} need to be satisfied through exhaustive search.

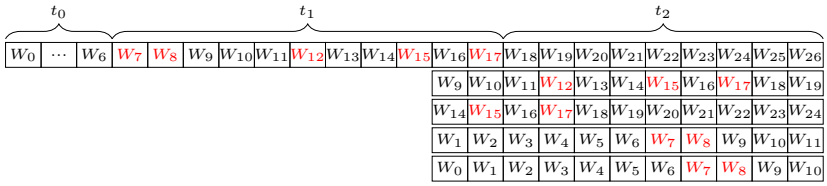

Fig. 2. Nonzero message words of the 27-step collision attack

28-step Collision Attack on SHA-2 for $t_1 = 11$ [7,14]. In the 28-step SHA-2 collision attack, $t_0 = 8$, spanning over steps 0–7. Similarly, $t_2 = 9$, spanning over steps 19–27. Within $t_1 = 11$, spanning over steps 8–18 ,where the expanded message words $(W_8, W_9, W_{13}, W_{16}, W_{18})$ are selected to inject difference, eventually forming the 28-step (steps 0-27) SHA-2 collision attack, as shown in Figure 3. Although t_1 middle steps with differences can induce differences spanning steps 8–14 in E_i and steps 8–10 in A_i, when performing message modifications, the bit conditions on E_{16}, W_{16} and W_{18} need to be satisfied through exhaustive.

29-step Collision Attack on SHA-2 for $t_1 = 11$ [9]. In the 29-step SHA-2 collision attack, $t_0 = 9$, spanning over steps 0–8. Similarly, $t_2 = 9$, spanning over steps 20–28. Within $t_1 = 11$, spanning over steps 9–19 ,where the expanded message words $(W_9, W_{10}, W_{14}, W_{17}, W_{19})$ are selected to inject difference, eventually

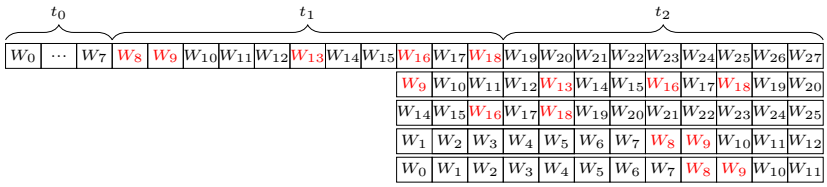

Fig. 3. nonzero message words of the 28-step collision attack

forming the 29-step (0-28 steps) SHA-2 collision attack, as shown in Figure 4. Although t_1 middle steps with differences can induce differences spanning steps 9–15 in E_i and steps 9–11 in A_i, when performing message modifications, the bit conditions on E_{16-18}, W_{17} and W_{19} need to be satisfied through exhaustive. However, Li et al. further extended the attack to 29 steps and selected $(W_{10}, W_{11}, W_{15}, W_{18}, W_{20})$ to search for a valid 30-step differential trail. Nevertheless, due to the excessive number of conditions that must be satisfied through exhaustive search, a practical 30-step collision attack could not be achieved [9].

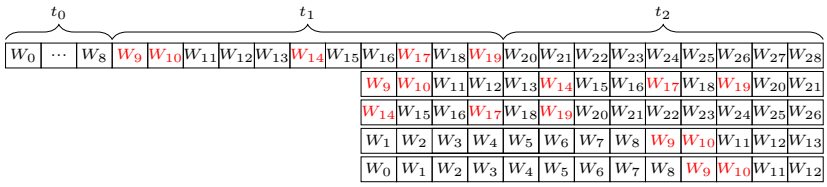

Fig. 4. nonzero message words of the 29-step collision attack

In the case of $t_1 = 11$, different non-zero message words are selected to study practical collision attacks. These three selection strategies for non-zero message words all keep t_1 and t_2 unchanged while gradually increasing t_0. In this series of message-word selection methods, the overall complexity of the collision attack depends on the bit conditions after step 16. If there are too many bit conditions from step 16 onward, it becomes difficult to realize such an attack. For example, the authors in [9] pointed out that the practical collision attack on 30-step SHA-512 cannot be achieved because there are too many bit conditions from step 16 and beyond.

Based on the analysis of the above three different non-zero message words selection strategies, we investigate the message selection strategies for a 31-step SHA-2 collision attack with $t_1 = 14$, a 32-step SFS collision attack with $t_1 = 16$, a 38-step SFS collision attack with $t_1 = 18$, and a 39-step SFS collision attack with $t_1 = 19$. However, at CRYPTO 2023, Hosoyamada and Sasaki pointed out that due to the lack of message freedom, the two-block approach cannot be used to transform the 38-step and 39-step SFS collision attacks into collision attacks. Our detailed analysis shows that there exist five different non-zero message words

selection strategies to further increase the number of steps in the collision attack, namely by keeping t_1 unchanged while changing t_0 and t_2, as shown in Table 2.

Table 2. Different choices of message words

t_1	t_0	t_2	Message word	Attack Step	Attack type
11	7	7	$W_7, W_8, W_{12}, W_{15}, W_{17}$	27 [2,13]	Collision Attack
	8	7	$W_8, W_9, W_{13}, W_{16}, W_{18}$	28 [7,14]	
	9	7	$W_9, W_{10}, W_{14}, W_{17}, W_{19}$	29 [9]	
14	5	12	$W_5, \ldots, W_9, W_{16}, W_{18}$	31 [7,8,9,14]	Collision Attack
16	2	14	$W_2, \ldots, W_8, W_{17}$	32 [13]	SFS Collision Attack
18	4	13	$W_4, W_5, W_7, W_{12}, W_{20}, W_{21}$	35	SFS Collision Attack
	5	13	$W_5, W_6, W_8, W_{13}, W_{21}, W_{22}$	36	
	6	13	$W_6, W_7, W_9, W_{14}, W_{22}, W_{23}$	37	
	7	13	$W_7, W_8, W_{10}, W_{15}, W_{23}, W_{24}$	38 [3,14]	
19	4	12	$W_4, \ldots, W_8, W_{12}, W_{13}, W_{20}, W_{22}$	35	SFS Collision Attack
	5	12	$W_5, \ldots, W_9, W_{13}, W_{14}, W_{21}, W_{23}$	36	
	6	12	$W_6, \ldots, W_{10}, W_{14}, W_{15}, W_{22}, W_{24}$	37	
	7	12	$W_7, \ldots, W_{11}, W_{15}, W_{16}, W_{23}, W_{25}$	38	
	8	12	$W_8, \ldots, W_{12}, W_{16}, W_{17}, W_{24}, W_{26}$	39 [2,7]	

- **Case-I:** Based on $t_1 = 14$, $t_0 = 5$, $t_2 = 10$, and the selected non-zero message words $(W_5, \ldots, W_9, W_{16}, W_{18})$, the 31-step SFS collision attack is converted into a 31-step collision attack [14]. When keeping $t_1 = 14$ unchanged while changing t_0 and t_2, we conducted extensive attempts but were ultimately unable to find better non-zero message words. As a result, it is not possible to further increase the number of attack steps under the condition $t_1 = 14$.
- **Case-II:** Based on $t_1 = 16$, $t_0 = 2$, $t_2 = 14$, and the selected non-zero message words $(W_2, \ldots, W_8, W_{17})$, which can easily achieve the 32-step SFS collision attack [14]. Even with $t_1 = 16$ fixed and t_0 and t_2 changed, no better non-zero message words can be found to further increase the attack steps.
- **Case-III:** Based on $t_1 = 18$, $t_0 = 7$, $t_2 = 13$, and the selected non-zero message words $(W_7, W_8, W_{10}, W_{15}, W_{23}, W_{24})$, which can easily achieve the 38-step SFS collision attack [14]. When $t_1 = 18$ is fixed, the 34-, 35-, 36- and 37-step SFS collision attacks can be achieved by selecting $t_0 = i$, $t_2 = 13$, and $(W_i, W_{i+1}, W_{i+3}, W_{i+8}, W_{i+16}, W_{i+17})$, where $i \in \{4, 5, 6, 7\}$.
- **Case-IV:** Based on $t_1 = 19$, $t_0 = 8$, $t_2 = 12$, and the selected non-zero message words $(W_8, \ldots, W_{12}, W_{16}, W_{17}, W_{24}, W_{26})$, which can easily achieve the 39-step SFS collision attack [2,7]. When $t_1 = 19$ is fixed, the 34-, 35-, 36-,

37- and 38-step SFS collision attacks can be achieved by selecting $t_0 = i$, $t_2 = 12$, and $(W_i, \ldots, W_{i+4}, W_{i+8}, W_{i+9}, W_{i+16}, W_{i+18})$, where $i \in \{3, 4, 5, 6, 7\}$.

Following Li *et al.*'s two-block method, we can easily convert the SFS collision attacks in **Case-II**, **Case-III**, and **Case-IV** into the corresponding collision attacks. However, because **Case-II** converts a 32-step SFS collision attack into a 32-step collision attack, the number of steps differs significantly from those in **Case-III** and **Case-IV**. Therefore, our focus is mainly on converting the SFS collision attacks in **Case-III** and **Case-IV** into collision attacks. We compare the message words selection strategies in **Case-III** and **Case-IV**, and identify a suitable **Case** to conduct collision attacks on SHA-2. We illustrate the approach by using the 35-step non-zero message words selection strategies from **Case-III** and **Case-IV** as examples.

Based on the non-zero message words at 35-step in **Case-III** and **Case-IV**, we use the SAT/SMT-based automated tools to search for the corresponding 35-step differential characteristics of the expanded message. In order to obtain differential trails that are more suitable for collision attacks, in other words, the differential characteristics in the first 7 steps of the expanded message words and those after step 16 should be as few as possible. **Model-1** searches for differential characteristics by taking the overall minimum as the objective function for **Case-III**. **Model-2** searches for differential characteristics by taking the overall minimum as the objective function under the constraint that the differences in W_{20} and W_{21} are minimized for **Case-III**. **Model-3** likewise searches for differential trails by taking the overall minimum as the objective function for **Case-IV**. The detailed search procedure is as follows:

- **Model-1:** Minimize the Hamming weight of ∇W_i. Specifically, find the minimal value of $t_w = \Sigma_0^{34} \mathbf{H}(\nabla W_i)$ such that the nonzero differences only exist in the 6 expanded message words $(W_4, W_5, W_7, W_{12}, W_{20}, W_{21})$. Note that the concrete message differences are not specified at this step and the only goal is to find the minimal value t_w.
- **Model-2:** Find a solution of $(\nabla W_i)_{0 \le i \le 34}$ with the minimal $\mathbf{H}(\nabla W_{20})$ and the minimal $\mathbf{H}(\nabla W_{21})$ by inject difference in $(W_4, W_5, W_7, W_{12}, W_{20}, W_{21})$. After testing, we can get $\mathbf{H}(\nabla W_{21}) = 1$, and $\mathbf{H}(\nabla W_{20}) = 2$. Finally, by maintaining the Hamming weight of $\mathbf{H}(\nabla W_{20})$ and $\mathbf{H}(\nabla W_{21})$, we seek an overall solution of $(\nabla W_i)_{0 \le i \le 34}$ with the minimum Hamming weight.
- **Model-3:** Minimize the Hamming weight of ∇W_i. Specifically, find the minimal value of $t_w = \Sigma_0^{34} \mathbf{H}(\nabla W_i)$ such that the nonzero differences only exist in the 9 expanded message words $(W_4, \ldots, W_8, W_{12}, W_{13}, W_{20}, W_{22})$. Note that the concrete message differences are not specified at this step and the only goal is to find the minimal value t_w.

Based on the above **Model-1**, **Model-2**, and **Model-3**, we obtained three results, which are named **Case-III-1**, **Case-III-2**, and **Case-IV**, respectively, as shown in Figure 5.

From the differential characteristic obtained by **Model-1**, the differences at W_{20} and W_{21} are relatively large, which leads to excessive differences in E_i,

thereby increasing the complexity of the collision attack. In **Model-2**, although the Hamming weights at W_{20} and W_{21} are reduced, the Hamming weights in the first 7 steps increase. In contrast, for the differential characteristic obtained by **Model-3**, the Hamming weights in the first 7 steps as well as those of W_{20} and W_{22} are relatively moderate.

From an overall comparison, selecting **Case-IV** is more suitable for launching collision attacks on SHA-2.

step	Case-III-1	Case-III-2	Case-IV
0	================================	================================	================================
⋮	⋮	⋮	⋮
3	================================	================================	================================
4	================================n	=uunun==nn==un====nuun=nn=======	==n=============================
5	======n======unnnn==============	================nuuuuuuuuuu===	=====u===u==========n===========
6	================================	================================	==n=============================
7	===uu==========n======n=n=n====	=unnnnnnnnnnnnun================	=======n=======u===u====u===u=u=
8	================================	================================	===========u=======uu==========
9	================================	================================	================================
⋮	⋮	⋮	⋮
11	================================	================================	================================
12	======u========nu==============	==============unnnnnnnnnnnnnn===	=====n===n==========u==========
13	================================	================================	==u=============================
14	================================	================================	================================
⋮	⋮	⋮	⋮
19	================================	================================	================================
20	===n=====u===u======u==u========	==============u=====n===========	=======nn=======u===============
21	======n==========u==============	================================n===	================================
22	================================	================================	==n=============================
23	================================	================================	================================
⋮	⋮	⋮	⋮
34	================================	================================	================================

Fig. 5. The 35-step differential trail for the message expansion

4 The First Practical Collision Attack on 35-step SHA-2

To further advance practical collision attacks on SHA-2, based on the above detailed analysis, we chose the message words $(W_4, \ldots, W_8, W_{12}, W_{13}, W_{20}, W_{22})$ to inject differences, enabling a 35-step collision attack on SHA-2. We use the SAT/SMT-based automated tool described in [7] to search for the corresponding differential characteristics. The detailed search procedure is presented below.

Step 1: **Minimize the Hamming weight of** ∇W_i. Specifically, find a solution of $(\nabla W_i)_{0 \le i \le 34}$ with the minimal value of $\sum_{i=0}^{34} \mathbf{H}(\nabla W_i)$ such that the nonzero differences only exist in the 9 expanded message words $(W_4, \ldots, W_8, W_{12}, W_{13}, W_{20}, W_{22})$.

Step 2: **Find the minimal number of differences on** $(E_i)_{14\leq i\leq 18}$. Specifically, with the fixed solution of $(\nabla W_i)_{0\leq i\leq 34}$ and under the condition

$$\forall i \in [0,34] \text{ and } i \notin \{4,\ldots,14\} : \delta A_i = 0,$$
$$\forall i \in [0,34] \text{ and } i \notin \{4,\ldots,18\} : \delta E_i = 0,$$

we take $t_E = \sum_{i=14}^{18} \mathbf{H}(\nabla E_i)$ as the objective function and search for the minimum value of t_E. At this stage, the differential trails found by the search are not fixed; only the value of t_E needs to be recorded.

Step 3: **Find the minimal Hamming weight of** $\mathbf{H}(\nabla A_i)_{4\leq i\leq 14}$. With the fixed solution of $(\nabla W_i)_{0\leq i\leq 34}$ obtained at Step 1, we aim to find a valid solution of $(\nabla A_i, \nabla E_i)_{0\leq i\leq 34}$ such that $\sum_{i=14}^{18} \mathbf{H}(\nabla E_i) = t_E$ holds. Meanwhile, we set a threshold on $\sum_{i=0}^{34} \mathbf{H}(\nabla A_i)$. Specifically, we choose an integer t_r and add the constraint

$$\sum_{i=0}^{34} \mathbf{H}(\nabla A_i) \leq t_r$$

to the model. If the solver can output a solution, we gradually reduce t_r until no solution exists for $(\nabla A_i, \nabla E_i)_{0\leq i\leq 34}$. We then keep the value of t_r.

Step 4: **Find the minimal Hamming weight of** $\mathbf{H}(\nabla E_i)_{4\leq i\leq 18}$. With the fixed solution of $(\nabla W_i)_{0\leq i\leq 34}$ and the values of t_A and t_E, under the above conditions, we find the minimal value of $\sum_{i=4}^{18} \mathbf{H}(\nabla E_i)$.

Application to SHA-256. With the above steps, we successfully find a high-probability valid differential characteristic for 35-step SHA-256, as shown in Figure 6.

message modification. Based on the 35-step differential trail shown in Figure 6, and by using message modification techniques similar to those in the memory-efficient collision attack framework, a practical collision attack on 35-step SHA-256 is enabled.

The message modification methods in the first two steps are the same as those in the memory-efficient collision attack framework, while the last step is slightly different and will be described in detail below.

Step 1: **Pre-processing Phase.** Find valid solutions of $(A_1,\ldots,A_{13},E_5,\ldots,E_{13},W_9,...,W_{13})$. Then choose N_{start} solutions with distinct $(A_1,\ldots,A_4,E_5,\ldots,E_8)$. Finally, according to the state (A_i,E_i) update function and each starting point $(A_1,\ldots,A_4,E_5,\ldots,E_8)$, first exhaust all possible (W_8,E_4) to obtain A_0. Then exhaust all possible (W_7,E_3) to obtain A_{-1} from each tuple (W_8,E_4,A_0). Based on such a process, we can obtain all valid tuples $(A_{-1},\ldots,A_{13},E_3,\ldots,E_{13},W_7,\ldots,W_{13})$, and store them in a table denoted by $\mathtt{TAB}_2$.

i	∇A_i	∇E_i	∇W_i
-4	==========	==========	
-3	==========	==========	
-2	==========	==========	
-1	==========	==========	
0	==========	==========	==========
1	==========	==========	==========
2	==========	==========	==========
3	==========	=====1=====011======0======0===	==========
4	==n========	==n0=0=1===100=0==0=1===0==1=0=1	==n========
5	=====n===n===n=u====n===u==u====	01011u001n=nuu=11000n=1=101u=100	=====u===u=========n==========
6	==========	101n=0=1=1=n1111==n0u===n=0n=n=u	==n========
7	==========	10u0=1=101==00=n==0=0===0==0=0=0	=======n=======u===u====u=1=u=u=
8	==========	uuu1=0=111=0=0=01=1=01==1==1=1=0	==========u=======uu==========
9	==========u==========u	11=10n0nuuu00n0u101=n11=u01u=unn	==========
10	==========	un1111110010011111=010u=u0+001100	==========
11	====n========u=u=======u==n====	1011n111100010u1u0111111u+nu1001	==========
12	=un===u=n=======n==========	001uuu11uuuuuuuu11n1000n1uuu1001	=====n===n==========u==========
13	==========	=n1111uu1n00000u1u0=nnn01111010n	==u========
14	==u========	=0=100110000000=101=0000=110===0	==========
15	==========	=1====0011===u10001=011===0n===1	==========
16	==========	======u=n====1==n=====0===01====	==========
17	==========	======0=0====1==0==========1====	==========
18	==========	==u===1=0=======1====1==========	==========
19	==========	==0==========	==========
20	==========	==1==========	=====0=nn=====0=u=1==========
21	==========	==========	==========
22	==========	==========	==n========
23	==========	==========	==========
24	==========	==========	==========
25	==========	==========	==========
26	==========	==========	==========
27	==========	==========	==========
28	==========	==========	==========
29	==========	==========	==========
30	==========	==========	==========
31	==========	==========	==========
32	==========	==========	==========
33	==========	==========	==========
34	==========	==========	==========

$W_4[1,8] \neq W_4[12,25], W_4[18] = W_4[14], W_5[0,1,30] = W_5[28,18,9], W_6[1,8] = W_6[12,25], W_6[18] \neq W_6[14]$.

$W_7[22,13,23] \neq W_7[18,9,8], W_7[11,14,20] = W_7[22,31,31], W_8[0,14,21] = W_8[28,25,6]$.

$W_8[31,23,30,15,22,8] \neq W_8[27,2,15,26,7,4], W_{20}[4,31] = W_{20}[6,22], W_{20}[31,30,25,21] \neq W_{20}[1,0,16,14]$.

$W_{22}[4,31] = W_{22}[6,22], W_{22}[27] \neq W_{22}[20]$.

$E_4[10] \neq E_4[15], E_5[3,21] = E_5[8,8], E_6[9,27,9,8] \neq E_6[23,14,14,27], E_6[1,1,23,6] = E_6[6,15,10,25]$.

$E_7[21,10] \neq E_7[3,15], E_{16}[28,20,20,6] = E_{16}[1,7,2,11], E_{16}[30,28,10] \neq E_{16}[12,10,29], E_{18}[24] \neq E_{18}[11]$.

$E_{18}[2] = E_{18}[16]$.

$A_3[29] = A_2[29], A_3[29] \neq A_5[29], A_3[26,4] = A_4[26,4], A_3[22,18,16,11,7] \neq A_4[22,18,16,11,7], A_{14}[9] = A_{14}[20]$.

$A_{14}[18,8] = A_{14}[6,17], A_{13}[30,25,23] = A_{14}[30,25,23], A_{13}[15] \neq A_{14}[15], A_{13}[29] \neq A_{15}[29], A_{15}[29] = A_6[29]$.

Fig. 6. The differential characteristic for 35-step SHA-256

Step 2: Try an arbitrary M_0, and get the corresponding chaining input (A_{-4}, A_{-3} A_{-2}, $A_{-1}, E_{-4}, E_{-3}, E_{-2}, E_{-1}$) to match A_{-1} from $\texttt{TAB}_2$. Once a match is found, perform the on-the-fly detection of the validity of A_{-4}, A_{-3}, and A_{-2}, which is indeed to test the conditions on (W_4, W_5, W_6). More details can be referred to [8].

Step 3: After $(E_i, A_i, W_i)_{0 \le i \le 13}$ are all determined, we use the degrees of freedom in W_{14} and W_{15} to satisfy the remaining bit conditions after step 16.

Based on the message modification method, we first need to find a valid solution that satisfies the differential characteristic. Then, based on this solution, we fix $(E_i)_{5 \le i \le 13}$, $(A_i)_{1 \le i \le 13}$, and $(W_i)_{9 \le i \le 13}$. To obtain as many valid tuples as possible

$$(A_{-1}, \ldots, A_{13}, E_3, \ldots, E_{13}, W_7, \ldots, W_{13}).$$

Here, we use a small trick: we re-enumerate the already fixed E_5, E_6, and E_7, obtaining 2^3, 2^4, and 2^{11} possible values, respectively. We then sequentially enumerate W_4 to verify E_8, and subsequently enumerate W_7 to verify E_3. Finally, we obtain $2^{29.1824}$ valid tuples

$$(A_{-1}, \ldots, A_{13}, E_3, \ldots, E_{13}, W_7, \ldots, W_{13}).$$

In Step 2, experimental results show that the probability of obtaining a valid M_0 is approximately $2^{-20.92}$, implying that about $2^{20.92}$ evaluations of the 35-step SHA-256 compression function are required to obtain one valid

$$(A_{-4}, \ldots, A_{13}, E_{-4}, \ldots, E_{13}, W_0, \ldots, W_{13}).$$

Complexity Evaluation. The time complexity of finding a valid solution in Step 1 is approximately $2^{34.3}$. In Step 2, the time complexity of obtaining one valid $(A_{-4}, \ldots, A_{13}, E_{-4}, \ldots, E_{13}, W_0, \ldots, W_{13})$ is about $2^{20.92}$. In Step 3, there are $2^{17.585}$ possible values for (W_{14}, W_{15}), and there are 45 conditions for $(A_{16}, E_{16}, E_{17}, E_{18}, E_{19}, E_{20}, W_{20}, W_{22})$. Thus, we need to find $2^{27.415}$ valid tuples $(A_{-1}, \ldots, A_{13}, E_3, \ldots, E_{13}, W_0, \ldots, W_{13})$. Since the time complexity of finding one such value is $2^{20.92}$, the time complexity of finding $2^{27.415}$ valid tuples $(A_{-4}, \ldots, A_{13}, E_{-4}, \ldots, E_{13}, W_0, \ldots, W_{13})$ is $2^{48.335}$.

Experimental verification. On a server with 128 threads, 378 GB of memory, and two AMD EPYC 9354 CPUs, we generated $2^{29.1824}$ valid tuples $(A_{-1}, \ldots, A_{13}, E_3, \ldots, E_{13}, W_7, \ldots, W_{13})$. We use the freedom in the message words (W_{14}, W_{15}) to fulfill the conditions on $(A_{16}, E_{16}, E_{17}, E_{18}, E_{19}, E_{20}, W_{20}, W_{22})$, and eventually find a practical colliding message pair, as shown in Table 3.

Application to SHA-512. With the above steps, we also successfully find a high-probability valid differential characteristic for 35-step SHA-512, as shown in Figure 7.

Fig. 7. The differential characteristic for 35-step SHA-512

Table 3. The colliding message pair for 35 steps of SHA-256

M_0	a8850273 c0f4a504 5d3ad7b5 6e5f5026 535cc256 e92ef7a5 436f70df 7d7e236a cadc14e8 d59ac191 6874f1ba 6b83960d f6dfe9de 6a013df2 f856b739 237894e8
M_1	c0008214 ae65f3bf e93c006a 5f195aa9 a4d6cd0f 21811cec ea897317 db9ec665 6ec17218 5100da8a 0912e57b a96b2054 45f2222c 4d12f88a d2701ecc 140976d1
M_1'	c0008214 ae65f3bf e93c006a 5f195aa9 84d6cd0f 25c114ec ca897317 da9fd6ef 6ec97e18 5100da8a 0912e57b a96b2054 41b22a2c 6d12f88a d2701ecc 140976d1
$hash$	c6209b2b 5e3fd4c8 96087364 046304ab bbc6dad9 403a26d1 018e351f e444451f

Due to the similarity between SHA-256 and SHA-512, the message modification techniques applied to SHA-256 can be naturally extended to SHA-512. Then, based on the 35-step differential trail on SHA-512 in Figure 7, W_8, E_4, W_7, and E_3 have 12, 37, 14, and 16 bit conditions, respectively. Due to the larger state size of SHA-512 compared to SHA-256, with a state word length of 64 bits, we cannot precompute all valid values for W_8, E_4, W_7, and E_3 in the same way as for SHA-256. Therefore, we propose a new message modification method, which is similar to the generic message modification method, except that it lacks the precomputation phase.

Step 1: Find a valid solution in the expanded message words $W_8 - W_{13}$, as well as the state words $(A_i)_{0\le i\le 13}$ and $(E_i)_{4\le i\le 13}$ by the step update function.

Step 2: Try an arbitrary M_0 and obtain the corresponding chaining input $(A_{-4}, \ldots, A_{-1}, E_{-4}, \ldots, E_{-1})$ and check the validity of A_{-4}, A_{-3}, A_{-2}, and A_{-1} on-the-fly. Then, based on the step update function, we can easily transform the validity check of A_{-4}, A_{-3}, A_{-2}, and A_{-1} into checking the conditions on (E_3, E_2) and (W_4, W_5, W_6, W_7).

$$
\begin{aligned}
E_3 &= A_3 \boxplus A_{-1} \boxminus \Sigma_0(A_2) \boxminus \mathrm{MAJ}(A_2, A_1, A_0),\\
E_2 &= A_2 \boxplus A_{-2} \boxminus \Sigma_0(A_1) \boxminus \mathrm{MAJ}(A_1, A_0, A_{-1}),\\
W_7 &= E_7 \boxminus A_3 \boxminus E_3 \boxminus \Sigma_1(E_6) \boxminus \mathrm{IF}(E_6, E_5, E_4) \boxminus K_7,\\
W_6 &= E_6 \boxminus A_2 \boxminus E_2 \boxminus \Sigma_1(E_5) \boxminus \mathrm{IF}(E_5, E_4, E_3) \boxminus K_6,\\
W_5 &= E_5 \boxminus A_1 \boxminus E_1 \boxminus \Sigma_1(E_4) \boxminus \mathrm{IF}(E_4, E_3, E_2) \boxminus K_5,\\
W_4 &= E_4 \boxminus A_0 \boxminus E_0 \boxminus \Sigma_1(E_3) \boxminus \mathrm{IF}(E_3, E_2, E_1) \boxminus K_4.
\end{aligned}
$$

Step 3: **Fulfill the Conditions on $(A_i, E_i, W_i)_{i\ge 16}$.** Up to this step, $(W_i)_{0\le i\le 13}$ have been fixed. Use the degrees of freedom in $(W_i)_{14\le i\le 15}$ to fulfill the remaining uncontrolled conditions on $(A_i, E_i, W_i)_{i\ge 16}$. If it fails, go to Step 2.

Complexity Evaluation. The time complexity of Step 1 is approximately $2^{37.56}$. In Step 2, there are 21 and 30 conditions on (E_2, E_3) and (W_4, W_5, W_6, W_7), respectively. Based on experimental evaluation, the complexity of finding a valid

tuple $(A_{-4}, \ldots, A_{-1}, E_{-4}, \ldots, E_{-1})$ is approximately 2^{49}. In Step 3, there are 2^{34} possible values for (W_{14}, W_{15}), and there are 46 conditions for (A_{16}, E_{16}, E_{17}, E_{18}, E_{19}, E_{20}, W_{20}, W_{22}). Thus, we need to find $2^{46-34} = 2^{12}$ valid (A_{-4}, $\ldots$, A_{-1}, $E_{-4}, \ldots, E_{-1}$). Hence, the time complexity of the 35-step SHA-512 collision attack is $2^{12+49} = 2^{61}$.

Experimental verification. The experiments were carried out on a computer with an NVIDIA RTX 5060 Ti GPU (16 GB memory). After 14 days of computation, we obtained 31 valid candidates tuples $(A_{-4}, \ldots, A_{-1}, E_{-4}, \ldots, E_{-1})$. These candidates were subsequently used to perform Step 3, which was executed on a regular personal computer. However, we did not find a practical colliding message pair.

In order to find a practical colliding message pair for 35-step SHA-512, we can elaborate more on our practical implementations. Specifically, as in the attack on 35-step SHA-256, after fixing one valid tuple

$$T = (A_1, \ldots, A_{13}, E_5, \ldots, E_{13}, W_9, \ldots, W_{13}),$$

we use Step 1 to find valid M_0. Once such a valid M_0 is found, we can further change some internal state words in T while keeping ($A_4, \ldots$, A_{13}, $E_8, \ldots, E_{13}$, W_{12}, W_{13}) the same as in the original solution. Then, we use a SAT solver to find a valid solution of $(A_0, \ldots, A_3, E_0, \ldots, E_7, W_0, \ldots, W_{11})$ to match the fixed IV and $(A_4, \ldots, A_{13}, E_8, \ldots, E_{13}, W_{12}, W_{13})$. We know there is at least one such valid solution. The goal is to find more such valid solutions with SAT solvers. This has the same effect as using more solutions of T (note that in the attack on SHA-256, we use only 1 solution). In this way, we do not need as many as 2^{12} valid M_0 as expected by reusing the degrees of freedom in other state words. Therefore, by using only $31 \approx 2^5$ valid M_0, we can find a practical collision. Equivalently speaking, we get about 2^7 valid T for each valid M_0.

In this process, the time complexity to find the original valid T is $2^{37.56}$. In Step 2, the time complexity for finding 31 valid M_0 is $2^{53.95}$. In Step 3, the time complexity for verifying one candidate is $2^{35.49}$ for each valid M_0, i.e., this time complexity $2^{35.49}$ is the cost to find a valid T and (W_{14}, W_{15}) with a SAT solver for each valid M_0, which was repeated 31 times in our practical implementation. This also indicates that generating more valid T for each candidate M_0 is very efficient with the solver. So, in our practical attack, the complexity is dominated by finding many valid M_0, and it is about 2^{54}. Finally, we find a practical colliding message pair for 35-step SHA-512, as shown in Table 4.

5 Collision Attacks on SHA-2 Up to 37 Steps

Similarly, by following the message word selection strategy described in Section 3, we can choose message differences at $(W_5, \ldots, W_9, W_{13}, W_{14}, W_{21}, W_{23})$ and $(W_6, \ldots, W_{10}, W_{14}, W_{15}, W_{22}, W_{24})$ to mount collision attacks on 36-step and 37-step SHA-2, respectively.

Table 4. The colliding message pair for 35 steps of SHA-512

M_0	6beb0fc6f8eff3a2 b49456bcaa893186 36acaa4450b7246e ad4a3956764275da 5adb698dd62486e7 9c75f643ea4ef25c f2980a1c55495f5a 7fb6116784a9d528 08158fb6f3811608 45acde2298893a54 d29aa6a38c56f297 a8639ecf5d3d9492 47d507346d25caf7 adf36dd973e8826b c637daf1daee9908 61ccb0377a32ef45
M_1	407fb97681b9b094 1afdd90f2b8ca40f 054612e51ce747af e4a40cb66ebf7a1e d1309321eff346b4 3ac36bbde8a4b898 9002c9bb96d87455 5e7d9e15332aeef6 eca644f1b8c8f9d7 fa7f118d64cf673b e6bf7c3a4c6d30cd 5fd65a016bf67839 110db279285b171e 8b2ae0013e0eb088 06353916c1dcf4e7 af64f9c23459e344
M_1'	407fb97681b9b094 1afdd90f2b8ca40f 054612e51ce747af e4a40cb66ebf7a1e d9309321eff346b4 3ecb6bbde8a4b898 9802c9bb96d87455 7e2dbe94332aeef6 ac8645f1b8c8f9d7 fa7f118d64cf673b e6bf7c3a4c6d30cd 5fd65a016bf67839 1505b279285b171e 832ae0013e0eb088 06353916c1dcf4e7 af64f9c23459e344
$hash$	e3c051778d486142 428bee20bed09588 a3c7439379fda549 f7728858582c163b e631e7f7fa1819f1 f2139d9c35f92536 83c7cd3f1a1917ee a708fc5646aa9087

5.1 The Collision Attack on 36-Step SHA-2

The detailed procedure for searching the 36-step differential characteristic is as follows.

Step 1: **Minimize the Hamming weight of ∇W_i.** Specifically, find a solution of $(\nabla W_i)_{0\le i\le 35}$ with the minimal value of $\sum_{i=0}^{35}\mathbf{H}(\nabla W_i)$ such that the nonzero differences only exist in the 9 expanded message words $(W_5,\ldots,W_9,W_{13},W_{14},W_{21},W_{23})$.

Step 2: **Find the minimal number of differences on $(E_i)_{14\le i\le 19}$.** Specifically, with the fixed solution of $(\nabla W_i)_{0\le i\le 35}$ and under the condition

$$\forall i\in[0,34] \text{ and } i\notin\{5,\ldots,14\}:\delta A_i=0,$$
$$\forall i\in[0,34] \text{ and } i\notin\{5,\ldots,19\}:\delta E_i=0,$$

we take $t_E=\sum_{i=14}^{19}\mathbf{H}(\nabla E_i)$ as the objective function and search for the minimum value of t_E. At this stage, the differential trails found by the search are not fixed; only the value of t_E needs to be recorded.

Step 3: **Find the minimal Hamming weight of $\mathbf{H}(\nabla A_i)_{5\le i\le 14}$.** With the fixed solution of $(\nabla W_i)_{0\le i\le 35}$ obtained at Step 1, we aim to find a valid solution of $(\nabla A_i,\nabla E_i)_{0\le i\le 35}$ such that $\sum_{i=14}^{19}\mathbf{H}(\nabla E_i)=t_E$ holds. Meanwhile, we set a threshold on $\sum_{i=0}^{35}\mathbf{H}(\nabla A_i)$. Specifically, we choose an integer t_r and add the constraint

$$\sum_{i=0}^{35}\mathbf{H}(\nabla A_i)\le t_A$$

to the model. If the solver can output a solution, we gradually reduce t_r until no solution exists for $(\nabla A_i,\nabla E_i)_{0\le i\le 35}$. We then keep the value of t_A.

Step 4: **Find the minimal Hamming weight of $\mathbf{H}(\nabla E_i)_{5\le i\le 19}$.** With the fixed solution of $(\nabla W_i)_{0\le i\le 35}$ and the values of t_A and t_E, under the above conditions, we find the minimal value of $\sum_{i=5}^{19}\mathbf{H}(\nabla E_i)$.

We use open-source tools developed in [7,8,9] to search for the corresponding differential characteristics of SHA-2, and the results are shown in Figure 8 and Figure 9. We also provide 36-step SFS collisions in Table 9 and Table 10 (refer to Appendix A) to demonstrate that the two differential characteristics are valid.

i	∇A_i	∇E_i	∇W_i
-4	================================	================================	
-3	================================	================================	
-2	================================	================================	
-1	================================	================================	
0	================================	================================	================================
1	================================	================================	================================
2	================================	================================	================================
3	================================	+++==============================	================================
4	================================	+++=0====1==01======0===1==0====	================================
5	=un============================	nnn=10=110=010=0==1=1===0==1=100	==u=============================
6	========n=====u====n======n====	111=u1101n=1un=0=111n=10u11n0=11	=====u===u=========n==========
7	================================	10000n0un0=n01001010u1==011u=0nn	==u=============================
8	================================	10u=10=0010000=n110001==10010000	=======n=======u===n====n=1=n=n=
9	================================	nnn==011001111000010010=1110001	============n=======uu==========
10	==========n==================nu=	11010000n0u=1000uuuuu1111nu0uuu1	================================
11	================================	100000010nu1n01111100uuu000111nu	================================
12	====n========u========u==n====	1001n00101u111u100111100u01n1110	================================
13	=nu===u=n======nu==============	10nnuuuuuuuuuu111000100n1uuu1001	=====u===n==========n==========
14	================================	=n1001101u11100n1n=0nnn010111000	==n=============================
15	==n=============================	=01100000=0111=000==0000=110====	================================
16	================================	=1=====001===u1111==011===0n====	================================
17	================================	==10===uu====1==n===000===01==0=	================================
18	================================	==1====00====1==0==========1====	================================
19	================================	==n====10=======1===10==========	================================
20	================================	==0=============================	================================
21	================================	==1=============================	=====1=nn=====0=u=1=============
22	================================	================================	================================
23	================================	================================	==u=============================
24	================================	================================	================================
25	================================	================================	================================
26	================================	================================	================================
27	================================	================================	================================
28	================================	================================	================================
29	================================	================================	================================
30	================================	================================	================================
31	================================	================================	================================
32	================================	================================	================================
33	================================	================================	================================
34	================================	================================	================================
35	================================	================================	================================

$E_{16}[13]=E_{16}[0], E_{17}[19]\neq E_{17}[6], E_{17}[20]\neq E_{17}[2], E_{18}[2]=E_{18}[16]$

$A_{14}[29]\neq A_{16}[29], A_{16}[29]=A_{17}[29]$

$W_5[1]\neq W_5[12], W_5[8]=W_5[25], W_5[18]=W_5[14], W_6[0]=W_6[28], W_6[30]=W_6[9], W_6[1]=W_6[18].$

$W_{21}[31,30]\neq W_{21}[1,0], W_{21}[25]=W_{21}[16], W_{21}[21]\neq W_{21}[14], W_{23}[4]=W_{23}[6], W_{23}[31]\neq W_{23}[22], W_{23}[27]=W_{23}[20].$

Fig. 8. The differential characteristic for 36-step SHA-256

Complexity of the collision attack on 36-step SHA-256. According to the memory-efficient collision attack framework [8], we have

$$n = 32,\ n_1 = 13,\ n_2 = 7,\ n_3 = 4,\ n_4 = 3,\ \mathcal{N}_{\mathtt{pro}} = 11,\ \beta \approx 60.$$

Therefore, the expected time complexity of the collision attack on 36-step SHA-256 becomes

$$\mathcal{T}_{\mathtt{pre}} + \mathcal{T}_{\mathtt{match}} = \mathcal{N}_{\mathtt{start}} \times (\mathcal{T}_{\mathtt{sat}} + 2^{19} + 2^{40}) + \frac{2^{32+10+60}}{\mathcal{N}_{\mathtt{start}} \cdot 2^{2n-n_1-n_2-n_3-n_4}} + 2^{71}.$$

In our experiments, the time complexity for finding a valid solution using a SAT solver is approximately $2^{35.8}$. When $\mathcal{N}_{\mathtt{start}} = 1$, we check all bit conditions on (E_3, E_4, W_7, W_8) and obtain in total $2^{38.585}$ valid tuple

$$(A_{-1}, \ldots, A_{15}, E_3, \ldots, E_{15}, W_7, \ldots, W_{15})$$

in the pre-processing phase, that is

$$\mathcal{T}_{\mathtt{pre}} + \mathcal{T}_{\mathtt{match}} = \frac{2^{32+10+60}}{2^{38.585}} + 2^{71}.$$

Hence, the time complexity of the 36-step collision attack is 2^{71}, and the memory complexity is $2^{38.585}$.

Experimental verification. On a server with 128 threads, 378 GB of memory, and two AMD EPYC 9354 CPUs. According to the message modification procedure, in the preprocessing phase we enumerate all possible (E_3, E_4, W_7, W_8) and store

$$(A_{-1}, \ldots, A_{12}, E_3, \ldots, E_{15}, W_7, \ldots, W_{15})$$

in a table. However, since the size of the table reaches $2^{38.585}$, the available memory resources are insufficient. Therefore, in the experimental validation phase, we fix in advance a valid pair (W_8, E_4). With the fixed values, we re-evaluate all constraints on (W_7, E_3, E_{-1}) and obtain $2^{25.4044}$ possible

$$(A_{-1}, \ldots, A_{15}, E_3, \ldots, E_{15}, W_7, \ldots, W_{15}),$$

which are stored in a table. We then try an arbitrary M_0 and use the corresponding output as input to verify the conditions on $W_6, W_5, W_7, W_{22}, W_{24}$, and E_{16}. Finally, we test $2^{43.9203}$ different values of M_0 and find solutions to satisfy part of the conditions, as shown in Table 11.

Complexity of the collision attack on 36-Step SHA-512. According to the memory-efficient collision attack framework, we have

$$n = 64,\ n_1 = 16,\ n_2 = 19,\ n_3 = 4,\ n_4 = 2,\ \mathcal{N}_{\mathtt{pro}} = 10,\ \beta \approx 126.$$

Fig. 9. The differential characteristic for 36-step SHA-512

Therefore, the expected time complexity of the new collision attack on 36-step SHA-512 becomes

$$\mathcal{T}_{\texttt{pre}} + \mathcal{T}_{\texttt{match}} = \mathcal{N}_{\texttt{start}} \cdot (\mathcal{T}_{\texttt{sat}} + 2^{89}) + \frac{2^{41+10+126-64}}{\mathcal{N}_{\texttt{start}}} + 2^{136}.$$

and the expected memory complexity is $\mathcal{N}_{\texttt{start}} \cdot 2^{87}$.

The time complexity for finding a valid solution using a SAT solver is approximately $2^{35.6}$. When $\mathcal{N}_{\texttt{start}} = 1$, the time complexity of the 36-step SHA-512 collision attack is 2^{136}, and the memory complexity is 2^{87}.

Experimental verification. On a server with 128 threads, 378 GB of memory, and two AMD EPYC 9354 CPUs. Similar to the experimental verification method, we are unable to store the 2^{87} valid tuples $(A_{-1}, A_0, E_3, E_4, W_7, W_8)$ in a table, because the available memory resources are insufficient. Therefore, in the experimental validation phase, we fix in advance a valid pair (W_8, E_4). With the fixed values, we then try an arbitrary M_0 and use the corresponding output as input to verify the conditions on $W_5, W_6, W_7, W_{22}, W_{24}$, and $E_{16}[0, 2, 5, 6, 10, 11, 12, 13]$. Finally, we test $2^{35.1341}$ different values of M_0 and find solutions to satisfy part of the conditions, as shown in Table 12.

5.2 Extension to Collision Attack on 37-Step SHA-2

We use a similar strategy to search for 37-step differential characteristics as in the above 36-step collision attack. The high-probability 37-step differential characteristics for SHA-256 and SHA-512 are shown in Figure 10 and Figure 11, respectively. The corresponding SFS collisions are shown in Table 13 and Table 14 (refer to Appendix A), respectively.

Complexity of the collision attack on 37-Step SHA-256. According to the memory-efficient collision attack framework [8], we have

$$n = 32,\ n_1 = 4,\ n_2 = 1,\ n_3 = 6,\ n_4 = 0,\ \mathcal{N}_{\texttt{pro}} = 4,\ \beta \approx 77.$$

Therefore, the expected time complexity of the new collision attack on 37-step SHA-256 becomes

$$\mathcal{T}_{\texttt{pre}} + \mathcal{T}_{\texttt{match}} = \mathcal{N}_{\texttt{start}} \times (\mathcal{T}_{\texttt{sat}} + 2^{28} + 2^{53}) + \frac{2^{32+4+77}}{\mathcal{N}_{\texttt{start}} \cdot 2^{2n-n_1-n_2-n_3-n_4}} + 2^{81}.$$

The time complexity for finding a valid solution using a SAT solver is approximately $2^{37.21}$. When $\mathcal{N}_{\texttt{start}} = 1$, we first exhaustive enumerate (E_4, W_8) and verified all the conditions, obtaining $2^{27.04}$ possible values. For (E_3, W_7), there are only 6 conditions on W_7. Therefore, we can find a total of $2^{27.04+26}$ possible valid tuples $(A_{-1}, A_0, E_3, E_4, W_7, W_8)$, that is

$$\mathcal{T}_{\texttt{pre}} + \mathcal{T}_{\texttt{match}} = 2^{53} + \frac{2^{32+4+77}}{2^{53.047}} + 2^{81}.$$

Hence, the time complexity of the 37-step collision attack is 2^{81}, and the memory complexity is $2^{53.04}$.

i	∇A_i	∇E_i	∇W_i
-4	================================	================================	
-3	================================	================================	
-2	================================	================================	
-1	================================	================================	
0	================================	================================	================================
1	================================	================================	================================
2	================================	================================	================================
3	================================	================================	================================
4	================================	+===============================	================================
5	================================	+=1+===0===1======0======+======	================================
6	n===============================	u=0+=1=1=1=0==1=+=100+0=1+=+==+1	u===============================
7	===u===n====uu====u===n==n======	0=nu=01n=0=u=00=+1u11+1=1n=+==+0	===n===u==========u=============
8	================================	11u01=000n1100n0u11nnnn1u11u00uu	n===============================
9	================================	u=0010=1=110=n0=0=00000=11=0=+00	===n=n=======u===u==n=n=n=u=u===
10	================================	u=1n00110001000=1111011=0u001+11	==========u======n=u============
11	====n===u==========n=======u==n	0110u0000nnn1n00110u1uu11n100uuu	================================
12	n===============================	u0111==u0u01u0010uu11nu111u1n10u	================================
13	==u=========n=n=======u==n======	nu=0u01101001nn11010101nuu111011	================================
14	====n=u=======u================	111u1==011==101=u01010n010111==0	===n===n==========n=============
15	================================	01=nu00u=0=0u11011nnn=010u====1n	n===============================
16	u===============================	===01110===00=+=1=000=0==1=====0	================================
17	================================	===01001===n1=+===011===0n=====0	================================
18	================================	00===nn====0==u===010===00==0===	================================
19	================================	0====00====1==0==========1======	================================
20	================================	n====10=======1===00============	================================
21	================================	0===============================	================================
22	================================	1===============================	=====uu=======n=1===============
23	================================	================================	================================
24	================================	================================	n===============================
25	================================	================================	================================
26	================================	================================	================================
27	================================	================================	================================
28	================================	================================	================================
29	================================	================================	================================
30	================================	================================	================================
31	================================	================================	================================
32	================================	================================	================================
33	================================	================================	================================
34	================================	================================	================================
35	================================	================================	================================
36	================================	================================	================================

$E_{17}[15] \neq E_{17}[2], E_{18}[21,22] \neq E_{18}[8,4], E_{20}[4] \neq E_{19}[18]$

$A_{15}[25,27] = A_{16}[25,27], A_{15}[17] \neq A_{16}[17], A_{16}[11,10,8] = A_{16}[22,19,20], A_{15}[31] = A_{17}[31], A_{17}[31] = A_{18}[31]$

$W_6[3,10,20] = W_6[14,27,16], W_7[2,0] = W_7[30,11], W_7[3] \neq W_7[20], W_8[3] \neq W_8[14], W_8[10,20] = W_8[27,16].$

$W_{22}[1,0,19] \neq W_{22}[3,2,10], W_{22}[28,27,23] = W_{22}[19,18,16], W_{24}[6,29] = W_{24}[8,22], W_{24}[1] \neq W_{24}[24].$

Fig. 10. The differential characteristic for 37-step SHA-256

Experimental verification. On a server with 128 threads, 378 GB of memory, and two AMD EPYC 9354 CPUs. Similar to the experimental verification method for 37-step SHA-256, we are unable to store the $2^{27.04+26}$ valid tuples $(A_{-1}, A_0, E_3, E_4, W_7, W_8)$ in a table, the available memory resources are insufficient. Therefore, in the experimental validation phase, we fix in advance a valid pair (W_8, E_4). With the fixed values, we re-evaluate all constraints on (W_7, E_3, E_{-1}) and obtain 2^{26} possible

$$(A_{-1}, \ldots, A_{15}, E_3, \ldots, E_{15}, W_7, \ldots, W_{15}),$$

which are stored in a table. We then try an arbitrary M_0 and use the corresponding output as input to verify the conditions on W_6, W_7, W_{22}, W_{24}, and E_{16}. Finally, we test $2^{37.6651}$ different values of M_0 and find solutions to satisfy part of the conditions, as shown in Table 15.

Complexity of the collision attack on 37-Step SHA-512. For the 37-step differential characteristic in Figure 11, we have

$$n = 64,\ n_1 = 4,\ n_2 = 1,\ n_3 = 6,\ n_4 = 0,\ \mathcal{N}_{\mathtt{pro}} = 4,\ \beta \approx 181.$$

The time complexity for finding a valid solution using a SAT solver is approximately $2^{32.3}$. Based on the solution that has been found, the time complexity of the 37-step SHA-512 collision attack is

$$\mathcal{T}_{\mathtt{pre}} + \mathcal{T}_{\mathtt{match}} = \mathcal{N}_{\mathtt{start}} \cdot (2^{32.3} + 2^{60} + 2^{59} \times 2^{58}) + \frac{2^{4+181+11-64}}{\mathcal{N}_{\mathtt{start}}} + 2^{185}.$$

When $\mathcal{N}_{\mathtt{start}} = 1$, the time complexity of the 37-step SHA-512 collision attack is 2^{185}, and the memory complexity is 2^{132}.

Experimental verification. On a server with 128 threads, 378 GB of memory, and two AMD EPYC 9354 CPUs. Similar to the experimental verification method, we are unable to store the 2^{132} valid tuples $(A_{-1}, A_0, E_3, E_4, W_7, W_8)$ in a table, the available memory resources are insufficient. Therefore, in the experimental validation phase, we fix in advance a valid pair (W_8, E_4). With the fixed values, we then try an arbitrary M_0 and use the corresponding output as input to verify the conditions on W_6, W_7, W_{22}, W_{24}, and the first 9 bit conditions of E_{16}. Finally, we test $2^{36.1491}$ different values of M_0 and find solutions to satisfy part of the conditions, as shown in Table 16.

5.3 Discussion on the 38-step SHA-2 Collision Attack

Similarly, we can select message differences at $(W_7, \ldots, W_{11}, W_{15}, W_{16}, W_{23}, W_{25})$ to search for 38-step SHA-2 differential characteristics. We obtain the 38-step differential characteristic for SHA-256 and one for SHA-512, as shown in Table 17 and Table 18 (refer to Appendix A), respectively. To demonstrate the validity of

Fig. 11. The differential characteristic for 37-step SHA-512

these differential characteristics, we further present the corresponding practical SFS colliding message pairs for SHA-256 and SHA-512, as shown in Table 19 and Table 20 (refer to Appendix A), respectively.

According to [8], the time complexity of a collision attack mainly depends on the size of N_{pro} and β. For example, for SHA-512, $\beta \approx 257$ and $N_{pro} = 0$, and the corresponding time complexity is approximately 2^{257}, which is also one order of magnitude higher than the complexity bound of SHA-512 collision attacks. Consequently, the 38-step collision attack on SHA-512 cannot be achieved either. Similarly, the same phenomenon occurs for SHA-256: in the 38-step SHA-256 collision differential characteristic, the size of β exceeds the bound of SHA-256 collision attacks.

6 Conclusion

In this work, by analyzing the memory-efficient collision attack framework proposed by Li *et al.* and the message word selection strategies used in existing SFS collision attacks, we select message differences at a series of message words $(W_{4+i}, \ldots, W_{8+i}, W_{12+i}, W_{13+i}, W_{20+i}, W_{22+i})_{0 \le i \le 3}$ to perform collision attacks on SHA-2. With this method, for both SHA-256 and SHA-512, we achieve the first practical collision attacks up to 35 steps, significantly improving practical collisions for SHA-2 over the state-of-the-art. Pushing the limit of this method, theoretical collision attacks on both SHA-256 and SHA-512 can reach 36/37 steps, marking significant progress in the cryptanalysis of SHA-2. We hope that this work will provide new insight into how to achieve practical collision attacks in the SHA-2 family.

Acknowledgments. We thank the reviewers of CRYPTO 2026 to improve the quality of this paper. Gaoli Wang is supported by the National Natural Science Foundation of China (No. 62472172), the Open Fund of Advanced Cryptography and System Security Key Laboratory of Sichuan Province (Grant No. SKLACSS-202301), ECNU Multifunctional Platform for Innovation (001). In addition, Yingxin Li and Gaoli Wang are supported by the Fundamental Research Funds for the Central Universities (NO. 40500-20104-222609). Fukang Liu has been supported by JSPS KAKENHI Grant Numbers 24K20733 and 26K14705. Jiali Shi has been supported by the Natural Science Foundation of Henan (NO. 262300420687), the Key Scientific Research Projects of Henan Province for Institutions of Higher Education (NO. 26B110005), and the Fundamental Research Funds for the Universities of Henan Province (NO. NSFRF2502061).

References

1. Damgård, I.: A design principle for hash functions. In: Brassard, G. (ed.) Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings. Lecture Notes in Computer Science, vol. 435, pp. 416–427. Springer (1989). https://doi.org/10.1007/0-387-34805-0_39, https://doi.org/10.1007/0-387-34805-0_39

2. Dobraunig, C., Eichlseder, M., Mendel, F.: Analysis of SHA-512/224 and SHA-512/256. In: Iwata, T., Cheon, J.H. (eds.) Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9453, pp. 612–630. Springer (2015). https://doi.org/10.1007/978-3-662-48800-3_25, https://doi.org/10.1007/978-3-662-48800-3_25
3. Eichlseder, M., Mendel, F., Schläffer, M.: Branching heuristics in differential collision search with applications to SHA-512. In: Cid, C., Rechberger, C. (eds.) Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers. Lecture Notes in Computer Science, vol. 8540, pp. 473–488. Springer (2014). https://doi.org/10.1007/978-3-662-46706-0_24, https://doi.org/10.1007/978-3-662-46706-0_24
4. Indesteege, S., Mendel, F., Preneel, B., Rechberger, C.: Collisions and other non-random properties for step-reduced SHA-256. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) Selected Areas in Cryptography, 15th International Workshop, SAC 2008, Sackville, New Brunswick, Canada, August 14-15, Revised Selected Papers. Lecture Notes in Computer Science, vol. 5381, pp. 276–293. Springer (2008). https://doi.org/10.1007/978-3-642-04159-4_18, https://doi.org/10.1007/978-3-642-04159-4_18
5. Landelle, F., Peyrin, T.: Cryptanalysis of full RIPEMD-128. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7881, pp. 228–244. Springer (2013). https://doi.org/10.1007/978-3-642-38348-9_14, https://doi.org/10.1007/978-3-642-38348-9_14
6. Li, Y., Liu, F., Wang, G.: Automating collision attacks on RIPEMD-160. IACR Trans. Symmetric Cryptol. **2023**(4), 112–142 (2023). https://doi.org/10.46586/TOSC.V2023.I4.112-142, https://doi.org/10.46586/tosc.v2023.i4.112-142
7. Li, Y., Liu, F., Wang, G.: New records in collision attacks on SHA-2. In: Joye, M., Leander, G. (eds.) Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part I. Lecture Notes in Computer Science, vol. 14651, pp. 158–186. Springer (2024). https://doi.org/10.1007/978-3-031-58716-0_6, https://doi.org/10.1007/978-3-031-58716-0_6
8. Li, Y., Liu, F., Wang, G., Dong, X., Sun, S.: The first practical collision for 31-step SHA-256. In: Chung, K., Sasaki, Y. (eds.) Advances in Cryptology - ASIACRYPT 2024 - 30th International Conference on the Theory and Application of Cryptology and Information Security, Kolkata, India, December 9-13, 2024, Proceedings, Part VII. Lecture Notes in Computer Science, vol. 15490, pp. 237–266. Springer (2024). https://doi.org/10.1007/978-981-96-0941-3_8, https://doi.org/10.1007/978-981-96-0941-3_8
9. Li, Y., Liu, F., Wang, G., Qian, H., Jia, K., Kong, X.: New collision attacks on round-reduced SHA-512. In: Kalai, Y.T., Kamara, S.F. (eds.) Advances in Cryptology - CRYPTO 2025 - 45th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2025, Proceedings, Part V. Lecture Notes in Computer Science, vol. 16004, pp. 200–229. Springer (2025). https://doi.org/10.1007/978-3-032-01901-1_7, https://doi.org/10.1007/978-3-032-01901-1_7
10. Liu, F., Meier, W., Sarkar, S., Wang, G., Ito, R., Isobe, T.: New cryptanalysis of ZUC-256 initialization using modular differences. IACR Trans. Symmetric Cryptol.

2022(3), 152–190 (2022). https://doi.org/10.46586/TOSC.V2022.I3.152-190, https://doi.org/10.46586/tosc.v2022.i3.152-190

11. Liu, F., Wang, G., Sarkar, S., Anand, R., Meier, W., Li, Y., Isobe, T.: Analysis of RIPEMD-160: new collision attacks and finding characteristics with MILP. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 14007, pp. 189–219. Springer (2023). https://doi.org/10.1007/978-3-031-30634-1_7, https://doi.org/10.1007/978-3-031-30634-1_7
12. Lu, Z., Yu, H., Lin, X., Yuan, S.: Collision attacks on reduced RIPEMD-128. IACR Trans. Symmetric Cryptol. **2025**(2), 16–33 (2025). https://doi.org/10.46586/TOSC.V2025.I2.16-33, https://doi.org/10.46586/tosc.v2025.i2.16-33
13. Mendel, F., Nad, T., Schläffer, M.: Finding SHA-2 characteristics: Searching through a minefield of contradictions. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings. Lecture Notes in Computer Science, vol. 7073, pp. 288–307. Springer (2011). https://doi.org/10.1007/978-3-642-25385-0_16, https://doi.org/10.1007/978-3-642-25385-0_16
14. Mendel, F., Nad, T., Schläffer, M.: Improving local collisions: New attacks on reduced SHA-256. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7881, pp. 262–278. Springer (2013). https://doi.org/10.1007/978-3-642-38348-9_16, https://doi.org/10.1007/978-3-642-38348-9_16
15. Mendel, F., Pramstaller, N., Rechberger, C., Rijmen, V.: Analysis of step-reduced SHA-256. In: Robshaw, M.J.B. (ed.) Fast Software Encryption, 13th International Workshop, FSE 2006, Graz, Austria, March 15-17, 2006, Revised Selected Papers. Lecture Notes in Computer Science, vol. 4047, pp. 126–143. Springer (2006). https://doi.org/10.1007/11799313_9, https://doi.org/10.1007/11799313_9
16. Merkle, R.C.: One way hash functions and DES. In: Brassard, G. (ed.) Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings. Lecture Notes in Computer Science, vol. 435, pp. 428–446. Springer (1989). https://doi.org/10.1007/0-387-34805-0_40, https://doi.org/10.1007/0-387-34805-0_40
17. National Institute of Standards and Technology: Fips 180-4: Secure hash standard (shs). Federal Information Processing Standards Publication 180-4, U.S. Department of Commerce (2012). https://doi.org/10.6028/NIST.FIPS.180-4
18. Nikolic, I., Biryukov, A.: Collisions for step-reduced SHA-256. In: Nyberg, K. (ed.) Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers. Lecture Notes in Computer Science, vol. 5086, pp. 1–15. Springer (2008). https://doi.org/10.1007/978-3-540-71039-4_1, https://doi.org/10.1007/978-3-540-71039-4_1
19. Sanadhya, S.K., Sarkar, P.: Attacking reduced round SHA-256. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) Applied Cryptography and Network Security, 6th International Conference, ACNS 2008, New York, NY, USA, June 3-6, 2008. Proceedings. Lecture Notes in Computer Science, vol. 5037, pp. 130–143

(2008). https://doi.org/10.1007/978-3-540-68914-0_8, https://doi.org/10.1007/978-3-540-68914-0_8
20. Stevens, M., Bursztein, E., Karpman, P., Albertini, A., Markov, Y.: The first collision for full SHA-1. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10401, pp. 570–596. Springer (2017). https://doi.org/10.1007/978-3-319-63688-7_19, https://doi.org/10.1007/978-3-319-63688-7_19
21. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the hash functions MD4 and RIPEMD. In: Cramer, R. (ed.) Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3494, pp. 1–18. Springer (2005). https://doi.org/10.1007/11426639_1, https://doi.org/10.1007/11426639_1
22. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: Shoup, V. (ed.) Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3621, pp. 17–36. Springer (2005). https://doi.org/10.1007/11535218_2, https://doi.org/10.1007/11535218_2
23. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Cramer, R. (ed.) Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3494, pp. 19–35. Springer (2005). https://doi.org/10.1007/11426639_2, https://doi.org/10.1007/11426639_2
24. Wang, X., Yu, H., Yin, Y.L.: Efficient collision search attacks on SHA-0. In: Shoup, V. (ed.) Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3621, pp. 1–16. Springer (2005). https://doi.org/10.1007/11535218_1, https://doi.org/10.1007/11535218_1
25. Zhang, Z., Li, M., Gao, L., Wang, M.: Collision attacks on SHA-256 up to 37 steps with improved trail search. In: Daemen, J., Thomé, E. (eds.) Advances in Cryptology - EUROCRYPT 2026 - 45th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Rome, Italy, May 10-14, 2026, Proceedings, Part VI. pp. 91–120. Lecture Notes in Computer Science, Springer (2026). https://doi.org/10.1007/978-3-032-25333-0_4, https://doi.org/10.1007/978-3-032-25333-0_4
26. Zhang, Z., Li, M., Wang, H., Hou, S., Wang, W., Wang, M.: Improved semi-free-start collision attacks on RIPEMD-160. In: Hanaoka, G., Yang, B. (eds.) Advances in Cryptology - ASIACRYPT 2025 - 31st International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, VIC, Australia, December 8-12, 2025, Proceedings, Part I. Lecture Notes in Computer Science, vol. 16245, pp. 221–252. Springer (2025). https://doi.org/10.1007/978-981-95-5018-0_8, https://doi.org/10.1007/978-981-95-5018-0_8

Supplementary Materials

A Some Tables

Table 5. The differential characteristic for 32-step SHA-256

i	∇A_i	∇E_i	∇W_i
-4	================================	================================	
-3	================================	================================	
-2	================================	================================	
-1	================================	================================	
0	================================	==01=========0============000	================================
1	================================	=0101==1===1=1=====1====1=101	================================
2	===u==========u================u	11un1==0=0=00u==1110=00==100unn	===u==========u================u
3	=======n==n========u=========u==	0u11uu1n01n0u0001=01uu010n=n00uu	============================n===
4	================================	011111110111u01010un01nn01=01n1n	========n===n=n===n====u=n===u=
5	==========n=====================	000=1101010n11n=1=10000101=10100	=====u=un=======n==========n====
6	===u==u======n==n======u=u======	u1n0=unu0=u10nn1n=0n=001uu1=001u	=====n===u=======nnn============
7	================================	1=1=0111u001n0=10=u0===0101=n==1	nnnn===n=====n====n=========u===
8	================================	0=1==1010111100=1=01===11==0==1	u========n===================
9	n=========n====================	n===0n=nuun1==01=1=1==111==110=	================================
10	================================	0==n==u=110u====n====u===u======	================================
11	================================	n==1==1=11u1====1====1===1======	================================
12	================================	1==1==0===10====1====1===1======	================================
13	================================	u========n====================	================================
14	================================	1========1====================	================================
15	================================	1========1====================	================================
16	================================	================================	================================
17	================================	================================	u========u====================
18	================================	================================	================================
19	================================	================================	================================
20	================================	================================	================================
21	================================	================================	================================
22	================================	================================	================================
23	================================	================================	================================
24	================================	================================	================================
25	================================	================================	================================
26	================================	================================	================================
27	================================	================================	================================
28	================================	================================	================================
29	================================	================================	================================
30	================================	================================	================================
31	================================	================================	================================

Table 6. The SFS colliding message pair for 32 steps of SHA-256

cv	f604f1fc eeb3a0ab a17e8834 b3c61523 e04e13b7 ad98efcc 85a314c4 8c19ae04
M'	de9958c9 dea3d356 e8c41e64 6314adfe d0f59a38 b2bcf93e 043c70b6 f15f77e4 6a615447 4c44bb60 32d873ed bf9c1ea0 14f370a0 ac13a010 ff46f88d 1637894b
M	de9958c9 dea3d356 f8c61e65 6314adf6 d0b08a9a b73c792e 007c00b6 005b57ec ea415447 4c44bb60 32d873ed bf9c1ea0 14f370a0 ac13a010 ff46f88d 1637894b
$hash$	5ccc4292 79d66eb4 23405cdc d88e2a1f 27710bb1 b91b9b8a 3def8bca fb9697ce

Table 7. The differential characteristic for 32-step SHA-512

i	∇A_i	∇E_i	∇W_i
-4	==========	==========	
-3	==========	==========	
-2	==========	==========	
-1	==========	==========	
0	==========	01==========0111111====	==========
1	==========	01=011=====111=0===0==0110100+100=00=01+000===1001100000000====	==========
2	=n==========nuu=====	uu=11101001011101110011101000+000101010+11000=001110nuuuuuu00101	=n==========n====
3	==nn=n======u=u======u==u===n=u========uu========n=====n====	uu0unu00110unn0u100u01nuuuuuuuuuu0uu1nuuuuu101nuuuuuu0unn0n10110	==========u=====
4	==========unn==========	u=011nunnnnnnnn0uu100+1unnnnu1uu10unnnn1uuuun00uuun1nnuuu1nnnn=n	====u==uuu==un=n=u=u======n=====nn====u=n======unnnnnnn======u
5	==========u====	=1=11001000u11010n010+11010010=11=0110100011010=0=1100000=nn1100	====nu===u=n==========nnuuuu=u==========n===u===
6	========n===u==========nn====nu=n=====u==un=nn==========	011n0u01nu0001110unnnnnn100nuu10=111u1uuuuuu01=1n0u1010011100100	===n==========n==========unnnn=n==
7	==========	111nuuuuu1uuu11==0101101==01000u110u1n010010nu0=11n1==00==un0==0	===u=====un====u=n==========un===
8	==========	====1=111u1100===0111110===011=00==01011111100==1=0======10====	========n==========n====
9	======n==========n====	===10011001111========0===0===1===001=====11====0======11====	==========
10	==========	========0===u========u===n====0===u==========u======11===1	==========
11	==========	========1===1========0===0=======1==========0======un===0	==========
12	==========	========0===1========1===1=======1==========1======00====	==========
13	==========	========n==========1n====	==========
14	==========	========0==========1====	==========
15	==========	========1==========1====	==========
16	==========	==========	==========
17	==========	==========	========u==========u====
18	==========	==========	==========
19	==========	==========	==========
20	==========	==========	==========
21	==========	==========	==========
22	==========	==========	==========
23	==========	==========	==========
24	==========	==========	==========
25	==========	==========	==========
26	==========	==========	==========
27	==========	==========	==========
28	==========	==========	==========
29	==========	==========	==========
30	==========	==========	==========
31	==========	==========	==========

Table 8. The SFS colliding message pair for 32 steps of SHA-512

cv	72480d1727bbe5ca d64e7c162dcb0bf5 4f7099e73d6f664d 4ce991972499a793 e06cf6ad9942ff05 5d58d9ef0a4e4c81 8838d0f44a2699e0 85794603c065cbb5
M	e1b8a9061a16648f 40dff5e533aef398 c43c8200014433b1 280430032018041c a4252135 7ecabfca ba1a0ab03e0b3bf1 100e1e43bdba28f6 03325e48fc7e760a 834a19345a9e1030 b4ca1cc9ec2584e6 01eec3a5189ad2df d124c0c41bd4e674 3eb8e53b200a8117 5bc2134d6006bf2f bc62f686a000004a be39221a48b401cb
M'	e1b8a9061a16648f 40dff5e533aef398 843c820001443391 280430032018045c ade87125 1f8ac04b b64a0a8f7e0b3af9 000e1e03bdba2902 13531e48fc7e7612 830a19345a9e1020 b4ca1cc9ec2584e6 01eec3a5189ad2df d124c0c41bd4e674 3eb8e53b200a8117 5bc2134d6006bf2f bc62f686a000004a be39221a48b401cb
$hash$	40128a85954b3855 aa46f1ce9d1a4432 3f6801e9835142c2 e608c519c1ee136a 3613f69145debf3c 36388893260abc99 2c3a3a79643fd897 5ec1f48ac8af0f06

Table 9. The SFS colliding message pair for 36 steps of SHA-256

cv	414e392b c207f5e7 56e6b04d 3b8b9fc2 54288441 10492883 8c7b95bb da1be99e
M'	37efdc48 edf7cc76 fd68dc33 5bbf9235 a5ab7e7b 162817f8 6a029e20 9fa012c4 39887bbf f14c73e3 9091f022 69f3cd47 aba6b039 e3e7a8e3 a8615f9e 1c664e4c
M	37efdc48 edf7cc76 fd68dc33 5bbf9235 a5ab7e7b 362817f8 6e429620 bfa012c4 38896b35 f1447fe3 9091f022 69f3cd47 aba6b039 e7a7a0e3 88615f9e 1c664e4c
$hash$	514ef8ec a900d078 478f724c f35a8d1d 1e5ff01c 03299e4c d02070dd 5fd1ab65

Table 10. The SFS colliding message pair for 36 steps of SHA-512

cv	8162f2f3c8e928a0 f2a702535e24fb57 588df0b1dfe519df 728c3772ebe543cb 8aaae45a3a01bc28 e24900d9271b5bd1 0a8e42fc4d33131c 91df55eb45dc517e
M	31343156a2fa85fa 0cf94ecbdac9b8cb 3cc33b7ff887206e 0e7c19da088c0352 fe9e633c3f8cb677 87ce7d4815306d73 47e7409c6a9131cd 165a7d883df24f19 5910c2609effb080 4aff9e930006ff5f 5eeb4ec325a56073 3479d03efc5830d6 d838ca6b356f2d47 1f6b5be779417fce df18af0c8cae3d2d da00be73d5849e2e
M'	31343156a2fa85fa 0cf94ecbdac9b8cb 3cc33b7ff887206e 0e7c19da088c0352 fe9e633c3f8cb677 8fce7d4815306d73 43ef409c6a9131cd 1e5a7d883df24f19 7940e2e19effb080 0adf9f930006ff5f 5eeb4ec325a56073 3479d03efc5830d6 d838ca6b356f2d47 1b635be779417fce d718af0c8cae3d2d da00be73d5849e2e
$hash$	6f8b131dfe2bc008 d45206edaf5d93d3 99437091f745cbd2 4c06e55b6a5e7099 502057a662ac6f24 3586db5050e4d91f 1c698ca99350554e a1cc553f2f844727

Table 11. A partial solution for the 36-step SHA-256 differential characteristic

i	∇A_i	∇E_i	∇W_i
-4	10100010111111010001001001010011	01011010001001001011000001011001	
-3	00000101101001101111110001100011	10000000001111011001000100000010	
-2	10110010010110101111010110100101	10111111000110010001011111110011	
-1	01001000001011010100100000010100	00100001101111000000011001100001	
0	11101010010111001110100110010001	10100101101000011011110101110000	00101110001110100011110011000000
1	11011001101010110000101011011111	10101111001101101110111010000011	11000000000001101000101100011010
2	01100011001110110100110111111110	01000000101010011000101010000101	00101110000011010101100000001110
3	11001111001010011100011000111111	00010010000010001100100010101001011	00000111111000011011010110000101
4	01011001111100000110010000110001	00010011111101010000011011000110	10001111100101010000110011101011
5	0un00011011111001000100101100100	nnn11011100010000011110000011100	10u01100100011001101111011010100
6	100100001n10111u0001n101110n1010	1111u1101n01un001111n110u11n0011	01111u001u1011111011n11001111011
7	10111001111111111010110001100001	10000n0un00n01001010u111011u10nn	10u11000110000010000001111011011
8	00110100011000110001101001100100	10u010000100001n1100010110010000	0011100n1000100u011n1011n011n1n1
9	11001010111111111010011110111101	nnn01011001111000010001011110001	11110001010n1000111uu1111100011
10	1010101001n01000010011100000nu1	11010000n0u=1000uuuuu1111nu0uuu1	10010000100100011111100000100010
11	00101101010101101110101111001100	100000010nu1n01111100uuu000111nu	01101001111100111100110101000111
12	0001n110000101u011110010u10n1000	1001n00101u111u100111100u01n1110	10101011101001101011000000111001
13	1nu110u1n110010nu111100001100001	10nnuuuuuuuuuu111000100n1uuu1001	11100u111n1001111010n00011100011
14	11111010001111001011000101001111	0n1001101u11100n1n10nnn010111000	10n01000011000010101111110011110
15	11n00011011011111101011111000111	10110000010111000001000011101100	00011100011001100100111001001100
16	01000111100000001101011000001100	01110000011101u1111100110000n1011	00110010111000111011011010000010
17	================================	==10===uu====1==n===000===01==0=	10001111000111110100100110011010
18	================================	==1====00====1==0==========1====	10101101001100011100000100111000
19	================================	==n====10=======1===10==========	10000111001111011001111111011101
20	================================	==0================================	11011011010010100110011001100000
21	================================	==1================================	1100011nn0011101u111000100101000
22	================================	================================	11111001101111111111101010101000
23	================================	================================	01u01001110100110010100010100000
24	================================	================================	10010010000000101011110000110110
25	================================	================================	10111011000100000100011111110100
26	================================	================================	10010010010001001111000000111011
27	================================	================================	00111010100101100011010000110000
28	================================	================================	10001011000001100000111001101011
29	================================	================================	00111001111100111010000100110011
30	================================	================================	11000001001101011111111111101010
31	================================	================================	01000010101100111100111010110110
32	================================	================================	00100100110001111110100010100000
33	================================	================================	11010100101100111111000010111010
34	================================	================================	10111101001100111100010101110100
35	================================	================================	11011100000010101011100101001101

M_0	012ce3c3, 1aeed679, 04f4ce3d, e2c4271e, 534951d2, ce83b73b, e5be65b, b0d5650c
	09049add, fdef7671, 4029f826, fd28af73, 9bd98b1d, e0dc7b41, 589d183, 7123d285

Table 12. A partial solution for the 36-step SHA-512 differential characteristic

i	∇A_i	∇E_i	∇W_i
-4	11100010110001001101001000011110001001011011001100100101011011111	10101010010000100101001000011001001010000110110110001011101111100	
-3	10101000001101111100111000001000100101100100110001001111000000011	11100010010111000000111000101010011010001110110001011011010001101	
-2	11011011111011101100001010010111100001010000011011100010000000001	10000001000011010100011101111010011001101101010100001011111001000	
-1	11011111111000110100101000101001110011101010100011100010100111	11101111111101110111111101100000101011010000010010000100000011	
0	10100110001111111100100000011011101100111001000011001001011110	11100010100010001000000000101100010010000111111010111001110000	00000111110001000101000010000101100110110111100000111010010110 10
1	11101110001001001011101011101110010011000101100001111001010100	01110000111001001110111010001111101110000101100000000010100011	11100100111010110110100000000100011100000110011010110011000100 10
2	01011010110111110111000001001011011011010011000001000010110001 1	01110001101111000001100110111111010011111101010011110100011100	01101001111100100011111101100100010101100110110100000101101000 11
3	11010100011101001110101011001000010010011011101001110101110101	00100001000111100011101010011100101010101110110100101110000010	00010000010011101110000001101110010011011010001001111110100011 1
4	10010111001010101111100010001010100100001000110110011001100010 01	01000011110110001000100010100000000000010111100011000010000111 1	01100000001111110011001110011100110100101101000111011011010100
5	0111u0000101100110111001100101010000100011110101000100111001 1100	110un1100010001110001001101110011110001100100011001101010001010 1	0001u1110110111000111111000111001111001001101010111110100110100 1
6	01101n1100001u01100n101u101010000u10010u111110u0111010110100111 10	00001nunnnnnnnuuuuuu1unn000001100100011100111nn1100011010000111 10	10000n100010u000100100000001101001000101101100010011011000011 01
7	00011111111111010111000011101100111000111010100101100101111111 1	010011n10nnuuu1n1010u01001000010 0n011nu0u101uunnnn11n1nu1uunnn 10	1011u1111101010001001100101001000111000001100010011111000100 10
8	00010100011110000110010011111100001100001111101110000101101 0011	1unnnn1001101100100000011000011010110001011111000100000000011 1	01u110010u0n000011u00010u110000u1001111011111111101000010000000
9	00100001111010011101110110000011000000101001110001011011101001 1011	0111u11101000111011111000010011001110111100001110101000111010011	0n00101011n1111110011110u1001001100000000000011011111110101111 1
10	01000100010010001100110100111010111010100101100n110n0n1000u0u11	10111n101001u01101n001u110001001u10011u0unnnnn01n00nu0n000u0nn00	01011110111010110100111010000110010010110100101011000000111001 1
11	01110001010010011000000100101100110101000111101001001111111001	00010111011un111u011000uunnu00nu1nu1n1010011111111100101014n0011	00110100011110011101000001111101111100010011000011010110
12	11110010100001000 1n000u001100111n00010u1111n0u01100111111110010 1	10011010111011110nu10un000000101n1001un11001nn0010100111nuuuu1100	11011000001110001100101001101011001101011011110010110101000111
13	00100nn11110u1000001n10111110010111010011111011011111000001000 0	00100nnnuu0un11n0u10000011100000001101unnnn00u011n1110011u11001	00011n110110n0111010110111110011011111001010000010011010110
14	01110100110111010000110000011010111101001101000000111101100001	100001u01001n1111un00101111111u0nn0100u000011nn01unnn0110001u10	1101n1110001100101011100001100100011001010111000011101001011 01
15	0110n10010011000010111011011010100001101110001000000110111111 1	10nuuuu10nuuu01001101uuuuuuuuuuuuuuuuuuuuuuuu0110111111111100110	11011010000000001011111001110011101010110000100100111100010111 0
16	==	==00111=00111===nuu=10u00000000=0110111101110u01==01110010110100	11110000001110001101111010011000010001110110010000011000010 0
17	==	==0000u==111u===111=n0=0010111101011111011111==1=====0===0===0==	10011011110001000101001011100111000000101111010011110001100100010
18	==	====0=1=====1===111=1=1=========================0==========	01110001001010101100110010111010111111010110010000100110010110
19	==	====n=1=====1=======1==	00001101011100010101100010011000010011010000111101111110100010 1
20	==	====0==	00101011110110010011111001100010010100100001011101100100000111 1
21	==	====1==	011000n00111n0001011u11111110100101011010011000100011101101010 1
22	==	==	01011011001111000011111001100010010000111100010110111010010111 0
23	==	==	1011u0101011010011010111010011000111000101010001101100010101011011
24	==	==	01011110100111101010110110111110111100111111001101100000001001110
25	==	==	11001000100010000001100100110100100010011000100110011101001111
26	==	==	01100110010000111110011001010100110011011111100001010110000001
27	==	==	00010110010111010100101111011100001000001100111111000100101010 1
28	==	==	10110001011110111010010001110110110000000111011010010100100101
29	==	==	10001101100011101010000111010101011000101100111101111000011111010
30	==	==	00000111100010000101111111010111101000001011001011111111001111 01
31	==	==	00010111000010100100010110100110110100001010001010111011110000
32	==	==	00111010110000100110110010110010010011111011010001000011100111 00
33	==	==	01000011110111001011000010110011010010001101010001100001
34	==	==	10101001110000111010000111011011000010110011101010011111110110011 1
35	==	==	01001110111011111000101010110110101000010101100111000010111111
M_0	e2ba0d7069dac1cc, 08e0ae761ea5be63, bb8443442851ea3c, 21a971f11f020357, 74984489329db36f, ad49c90809eeca65, 8ea01a163e5079d2, f75ea5bb22bd999b,		
	6ac6543807df7028, 86134348d9c836a2, d284203f594bac94, 499030e8fd45e992, c439cc1e6e8a138d, f417019f14758a9c, cb1675a8ffc95656, 72bbf6e7ffa3b06f.		

The red-marked bits indicate bits that are satisfied.

Table 13. The SFS colliding message pair for 37 steps of SHA-256

cv	fb70a9c5 0ed13c48 16a246fb e5b1c8d6 2f1f3bbe ddcc317 f1248192 906b35d3
M'	f85a304f 754dbbaf 7181f858 85980da3 0be2a032 0c9593e7 69a00417 7eefd9ff 8d1bf625 7f718fc5 d7c6c040 6019ac52 8ae0d5d2 d54aca04 35f0acc0 bcf8ca22
M	f85a304f 754dbbaf 7181f858 85980da3 0be2a032 0c9593e7 e9a00417 6feff9ff 0d1bf625 6b75c56d d7e69040 6019ac52 8ae0d5d2 d54aca04 24f08cc0 3cf8ca22
$hash$	c63cd26b c3e9b777 82725e7e 5b6d4f71 0fb52f35 3114fa3a dd8c186e c3b2e5a1

Table 14. The SFS colliding message pair for 37 steps of SHA-512

cv	67819f233e3c01af 9d0ec130caee42e6 d736534c37058fbf cf111fe6e5b673e4 1ab625ff45661014 41bc862d76bb1dd5 3a52106a0a1d6bc9 c30746bb9f31c39e
M	20a4664d0ee2a806 5f8c5d2b028e1893 c8b35516ff257481 c45befebdd02d3c1 b52d8a3b4d3a67c3 4b8c01e57a3f9f7c 0396806066988b83 c4bf0e1b7c846769 993fbaf59899649c 8f4ead969c1adb0c bf80f43200a94044 a8f66ee7a6e43470 0358e957c20709f6 dd3e3152223155d4 51960e8064a24e5b fb8bff7d370ed7dc
M'	20a4664d0ee2a806 5f8c5d2b028e1893 c8b35516ff257481 c45befebdd02d3c1 b52d8a3b4d3a67c3 4b8c01e57a3f9f7c 2396806066988b83 d49f0e1b7c846769 b93fbaf59899649c 0e0e2f929c1adb0c bf00f03200a94045 a8f66ee7a6e43470 0358e957c20709f6 dd3e3152223155d4 41b60e8064a24e5b db8bff7d370ed7dc
$hash$	00d282230e77e75a 2d4d1729fea961c0 d5c2a37158ee2135 8f34ad01d2bffbb4 9f77f39828444b73 8a0370dcb0eca162 da5d0ef6eb091c6a 1722f104a4e3df82

Table 15. A partial solution for the 37-step SHA-256 differential characteristic

i	∇A_i	∇E_i	∇W_i
-4	11010101110110100111110101101000	01011110010110110000101001101101	
-3	11101111000001101001110110111100	10100101100000000100101010010001	
-2	00011001000011111001111101001101	01110111100100111011110110011110	
-1	01000101110101110111111001101100	10011101011100110001110100100110	
0	10111010101101010110101111010000	10000011100110100000100110010101	01010000000111100000000100111000
1	00000011101111101110101101100010	00110011101001100011110000011101	10001111000110011000000010001011
2	10101100010010111101110110100101	10100001010100011110001001110001	10100110101000111001110111010010
3	11100011101100001000001001100000	11010101011111110001000101100110	11000100101101000001101100110101
4	00000011101011001010100010111111	00110000110001001000101010011101	00001101010001010100010110010111
5	00001111111110101111011001101001	00111000010111111000111011100010	11010000100100111010111111101011
6	n1001000110000101100110000011001	u10101011110011010100000111100011	u11001010100010001000000011001011
7	011u110n1001uu0111u101n01n111010	00nu101n100u100011u110101n101110	001n010u1000100100u0010101011000
8	10000010100000101100111011010000	11u011000n1100n0u11nnnn1u11u00uu	n0001101000110111111011000100101
9	11011110010001110001100101100101	u10010011111 01n0001000001110010 00	011n1n1101110u011u00n1n1n1u0u101
10	00011001101011100110101111101010	u11n0011000100001111011 10u001011	1101011111u001101n0u000001000000
11	1110n110u00001110000n1001011u10n	0110u0000nnn1n00110u1uu11n100uuu	01100000000110011010110001010010
12	n0010111101011101111011011001110	u011101u0u01u0010uu11nu111u1n10u	10001010111000001101010111010010
13	11u011001101n0n1001111u10n011110	nu10u01101001nn11010101nuu111011	11010101010010101100101000000100
14	1101n0u1000000u01011111001101101	111u11101110101u01010n010111100	001n010n1111000010n0110011000000
15	11001000000000011111111111001110	010nu00u1010u11011nnn1010u11101n	n0111100111110001100101000100010
16	n0101001110111010101101011000010	00101110000000011000010101100110	01111010011001001101001001000000
17	================================	===01001===n1=+===011===0n=====0	00111010010010000001010111000101
18	================================	00===nn====0==u===010===00==0===	01110000000001101000111111101011
19	================================	0====00====1==0==========1======	11010110110000010111101010110111
20	================================	n====10=======1===00============	10011110111011101000110111000001
21	================================	0================================	00100010110110011101100001011010
22	================================	1================================	10110uu1011010n01001000101111001
23	================================	================================	00101101010011001110010101000111
24	================================	================================	n10010010000001010100110101100001
25	================================	================================	00001000101111111010000001011111
26	================================	================================	10000001100011100111011011011100
27	================================	================================	10100100011010111111111001011100
28	================================	================================	01000100110100110101101100011101
29	================================	================================	10110101011010000011000110000001
30	================================	================================	10001010101010000001100000011010
31	================================	================================	01100000000001000001101100001101
32	================================	================================	00011010100111100000110010100111
33	================================	================================	11000101111011111010101010001000
34	================================	================================	11000111111101001011000000000101
35	================================	================================	01101111010010010100001010101100
36	================================	================================	11101000100100100110010101110111

M_0	c0efdd5f, 6a4b663f, df4fe091, 3f074469, adcd1142, 210e8094, c1a58251, 9684d4b6
	4878d5ef, 28530ba4, 989c5ded, f6d41ebc, a3880ee8, 09b7036a, 2e14572e, 7d33177e

Table 16. A partial solution for the 37-step SHA-512 differential characteristic

i	∇A_i	∇E_i	∇W_i
-4	0101110010110011110000111100111010100010110000101011000110011001	0011000100100000111111001010100010001110111001111001010100110	
-3	0001100011101110110010100110011010003010011110000101110010010010	1011000100110101011110111111111111010111100101011000100010100	
-2	0101000101110101001000011000110010000111110111111111000000010011101100001111101000001	1000011000110001010100111111000000001001110110000011110110000000	
-1	1110011110010010011000010011000110100111000000101000010001011011	0101000000110000010001000010001101001001011100101111011010000010	
0	1100010001011100000001100000101100100101011010110101100111111101010	0101110001011001011101111111101011010100101110010011000111001010	0111100101000111100111101000011010011011100100000010011000110100
1	1000001111110101111111011010011100100101111110101110000111010101	0100110111001110011000111110011001100010101011110011111100000000	0101001111100001011111001010010001000111100101110111100110111111
2	0000111010001110010111110001000100001111111010010101001110100111	0110101000001101101011101001100010111001100101100001010010000100	1010100010100100111100100101011010011101010010000111010100
3	0001001101111111111000010010000100011101100100111000101000111001000100111011000011111010001	1101010011000101010010111000101000111001001001110100001111110100	1100010001001011111110000010011011110100110001101111111010111111
4	0100001100101111001001111110011010111111011011111101100000010000100	1111000101100101001100011111011000100111110101111111101111011110	0001100111101001010010010110101010001110010010000100010011110
5	1000100100010100011010000110100000111001101001100010001100010000001	0111010110000010011100110011010010011000011110010011101100110011	1000010001111011111011100110010010100010001011001010010110111010111
6	01u0001110110100010110110010000100011100001111010111001100010	00u0101010000110011100100100001010101001001111101001111000110001	11u011110001100001110100110010010000010001111100110100100001110
7	010u100101n00110n101u011100001u11100u00001uu10110001010001101101101	110unuuuuuunuuuuu0unn11011000111011100110nu011110010101010001100	010u010010n011100100110000001101000100110011111100100110111011011
8	0011110001111110111001101101011100101100111111110100100110110	1011u1nn0101un0001n10100011101u10nu11u000101010nuun0101n1nuu1001	10u1100100111111101111011011101100100110010010010011100
9	0111110100100111010101101111111011011000101001010100110000000n	nu0u1000011111unn001n111100100n000000100111n1010100100110100100nu	n000111n0n001110n01011u110010n10100111000001101011011010001100
10	0011110101101010000001110011101001010110000111000100010100111010100	00u1001100111110001001101100u00010n1110u1111001110010000101001100	10111111n000000011110n000011001000000000101010010100000000100010u
11	0011101110010111110011110011101010100000111n100n0n0001n0n1110	111u00011nu10101n111u001010101u01111u00010u0uuu01n0uu111n0uu1101	1010100011110110010111011100110011010011001000011010001110000
12	11u1101111011110111010100000011000110101110000010010011000000110	nnn110001nu100n110010uu0uu100u0un1n01100000010011100111010u001101	0000001101011000111010010101011111000010000001110000100111110110
13	0011110101100100u100n110000111u01001u0011u0u1100101000010001101	1010000110100100un010n0010000000111101100110u011010u111011u0010nu	1101110100111110001100101001000100010011000101010111010100
14	0111u10010n0001101u011010110001011100100010111001100011001001000	01110uuu10n01u100n1000010010110110010011n0n00101nnnnnn110n0010	010n000110u101100000111010000000110010010000100100111001011011
15	0011010101010110000000001001100101011101000001001000000010011	101110unnnn10nuu0uu1010011000001u01011011001nn100unn111nuuuu1101	11n1101110001011111111110111101010011011100001110110101111011100
16	==n==========n=========n=========n=

Table 17. The differential characteristic for 38-step SHA-256

i	∇A_i	∇E_i	∇W_i
-4	==========	==========	
-3	==========	==========	
-2	==========	==========	
-1	==========	==========	
0	==========	==========	==========
1	==========	==========	==========
2	==========	==========	==========
3	==========	==========	==========
4	==========	==========	==========
5	==========	+++========	==========
6	==========	+++=0====1==01=====0+===0==0====	==========
7	=nu========	uuu=111==0=0101===01+===11001=00	==n========
8	=========n=====n====u======n====	011=u0000n=1nu01=11un=00u10n0=11	=====u===u==========u==========
9	==========	10u11n111u=u01110000011=0un0=1nn	==u========
10	==========	111=101100=1001u==00n===1001=101	=====n=n=======u===n==u=u=u=n=n=
11	==========	uuu01011un=01001O=110101=1111010	==========n======n=u==========
12	==========n	11001110nu1=01nu=0unn10110nu00=n	==========
13	==========	un11100101==n10101011u=11010100u	==========
14	==========n====	0110100111010111=000100nuuuu0110	==========
15	======n=u=======n==========	11nuuuu01111010n0u00100101010u11	=====u===n==========n==========
16	==========	111011111nu=0001001u=n11010u00=u	==u========
17	==u========	nnn110110100unnnn10001===1010100	==========
18	==========	010nn1000001111n00=1101==uuu0100	==========
19	==========	10n00nuun01nuuuuu111111111010=01	==========
20	==========	011110000==111111=======111====	==========
21	==========	unn==0110==110101=1001=====110==	==========
22	==========	111========	==========
23	==========	111==========	=======uu=======u==========
24	==========	==========	==========
25	==========	==========	==n========
26	==========	==========	==========
27	==========	==========	==========
28	==========	==========	==========
29	==========	==========	==========
30	==========	==========	==========
31	==========	==========	==========
32	==========	==========	==========
33	==========	==========	==========
34	==========	==========	==========
35	==========	==========	==========
36	==========	==========	==========
37	==========	==========	==========

Table 18. The differential characteristic for 38-step SHA-512

i	∇A_i	∇E_i	∇W_i
-4	==========	==========	
-3	==========	==========	
-2	==========	==========	
-1	==========	==========	
0	==========	==========	==========
1	==========	==========	==========
2	==========	==========	==========
3	==========	==========	==========
4	==========	==========	==========
5	==========	+==========	==========
6	==========	+++100+++101001=0++==========01==========	==========
7	n==========	n++111+++110011=0++====1===0+0===0==+===00++1==+====+==0+======	u==========0======
8	=n======n=====n===u========u=====n====n=u==========	1unuuuuuunuuuuu=unn======101+001=111+000un++0+==+====+==1+===10+	=u======n==========
9	==========	000unu010un1nnn=u11=0=11001unn0++u10n1=0u0nuuu=0u1000u00un===00+	n==========
10	==========	uu0=00011000110==01=1101011100=++11=00=1=010001001==101000=01nuu	=====u=n======n=====n======u==========
11	==========n==========	n001011101nu01000+01100unnnn101unn01un0n100000000001100111001001	======u==========n==========
12	==========u==n==u====u=n======	0n0101nuu0000nu11+u011110001u1010nu100001u1111uuun10unnn0n11110u	==========
13	n==========	u1010011n0nu10011n0=n1u00unn0un1u0100110=10=n11u0000100n00101100	==========
14	==========u===n========u=====n====u=u==========	011011011=10=1u==1n10=00unnn0001000=10001u011=000111001101=0nuuu	==========
15	==n=====u=======u==========	1=n==u11u01001100110011+100110110un1111n10n00=00n00100uu00000101	=n======n==========
16	==========	1=010nuuu10=1=101=10000+011000n1001001100unu1010un101nuuuu011000	n==========1======
17	u==========	10u0011nu==00=11n01uuuuuuuuuuuuuuuuuuuuuu1100001111010010111011	==========
18	==========	==1==0111=====u=0=n1111111111001110110010u01====01===01011=0====	==========
19	==========	+=n===01u=====0=u=01100111011011111000000==0==========0====1=	==========
20	==========	+=1=====1=====1=1=0==========0==========	==========
21	==========	n=1=====1======1==========	==========
22	==========	0==========	==========
23	==========	1==========	==u=====n======n==========0==========
24	==========	==========	==========
25	==========	==========	n==========
26	==========	==========	==========
27	==========	==========	==========
28	==========	==========	==========
29	==========	==========	==========
30	==========	==========	==========
31	==========	==========	==========
32	==========	==========	==========
33	==========	==========	==========
34	==========	==========	==========
35	==========	==========	==========
36	==========	==========	==========
37	==========	==========	==========

Table 19. The SFS colliding message pair for 38 steps of SHA-256

cv	b390ee83 29821ad3 dab1ca7b fc355e00 13be0f15 7e96a98e e4761863 8746fd83
M'	e5844dd9 b223dfba a8ac6b7f 9aec5e3c 851ac3c8 61aa2353 d9ec17b2 76e622b5 f9b9f475 419e8702 7f881d1a 374a9838 3d3c6845 309ecba9 68bcbcd1 c94debc2
M	e5844dd9 b223dfba a8ac6b7f 9aec5e3c 851ac3c8 61aa2353 d9ec17b2 56e622b5 fdf9fc75 619e8702 7a890fb0 37428c38 3d3c6845 309ecba9 68bcbcd1 cd0de3c2
$hash$	d5751b17 ef21e93a 2a8e4687 edb5d365 74ddddfb 54d027f3 54447619 36148c5f

Table 20. The SFS colliding message pair for 38 steps of SHA-512

cv	1f63b927ddb15896 a648d9cd68960aea 937848593b2f22e2 97303318d12e43dc cd3a00581b3eb8ba 2dae70cd53fd5bd1 4683c946c3afaa39 aacefd5c3aef305e
M	9484c4c9bbae6e65 a53d4de431804786 21c4e41262443dba b173f7f066d9ca6c c52418bb28c8a4f1 55e55f9ea1613da5 ee797b5a86eb0acf 44872c0c8613200a 1bcaff6744bf141d a2a3a37fdda61118 912a9f80b5653c6b b054b951bd5d8717 23157558ba843ee5 9d058ce66b15a490 a3b74697e52fc11e 4fe5ad0e5e7fd19e
M'	9484c4c9bbae6e65 a53d4de431804786 21c4e41262443dba b173f7f066d9ca6c c52418bb28c8a4f1 55e55f9ea1613da5 ee797b5a86eb0acf c4872c0c8613200a 5b4aff6744bf141d 22a3a37fdda61118 94289790b5653c6d b254a951bd5d8713 23157558ba843ee5 9d058ce66b15a490 a3b74697e52fc11e 0f65ad0e5e7fd19e
$hash$	7a49984c02aef6b4 9f0141d8943a99aa c20fa91c87d2c3dc 8c8a8721d51e12e2 b9e519144d15c172 cd4af1ffd7128a55 406ade942fc41965 b0e9ff96fe98db99