Electronic Research Archive

http://www.aimspress.com/journal/era

ERA, 30(6): 2018–2032.
DOI: 10.3934/era.2022102
Received: 04 March 2022
Revised: 25 March 2022
Accepted: 05 April 2022
Published: 13 April 2022

Research article

Improved collision detection of MD5 with additional sufficient conditions

Linan Fang[1], Ting Wu[2], Yongxing Qi[2], Yanzhao Shen[3,*], Peng Zhang[4], Mingmin Lin[1] and Xinfeng Dong[5]

[1] School of Cyberspace, Hangzhou Dianzi University, Hangzhou 310018, China
[2] Hangzhou Innovation Institute, Beihang University, Hangzhou 310020, China
[3] Shandong Institute of Blockchain, Jinan 250102, China
[4] Hangzhou Hikvision Digital Technology Company Limited, Hangzhou 310051, China
[5] Science and Technology on Communication Security Laboratory, Chengdu 610041, China

* **Correspondence:** Email: shenyanzhao@sdibc.cn.

Abstract: One application of counter-cryptanalysis is detecting whether a message block is involved in a collision attack, such as the detection of MD5 and SHA-1. Stevens and Shumow speeded up the detection of SHA-1 by introducing unavoidable conditions in message blocks. They left a challenge: how to determine unavoidable conditions for MD5. Later, Shen et al. found that the unavoidable conditions of MD5 were the sufficient conditions located in the last round of differential paths. In this paper, we made further work. We discover sufficient conditions in the second round that can also be used as unavoidable conditions. With additional sufficient conditions, we subdivide three sets and distinguish seven more classes. As a result, compared with Shen's collision detection algorithm, our improved algorithm reduces the collision detection cost by 8.18%. Finally, we find that they do exist in the differential paths constructed by the automatic tool "HashClash".

Keywords: counter-cryptanalysis; MD5; collision detection; unavoidable condition; sufficient condition

1. Introduction

A hash function compresses messages of arbitrary size into a fixed-length bit array. It is one of the basic components of security applications, for example, digital signatures and message authentication codes. The design of many classical hash functions such as MD5 and SHA-1 is based on the Merkle–Damgård construction [1, 2]. This construction operates on the padded input message using a compression function and updating a fixed-size internal state (also called chaining value).

A secure hash function must fulfill three properties, i.e., pre-image resistance, second pre-image resistance, and collision resistance. Collision resistance means that it is practically impossible to find two different messages with the same hash value. Much research has been carried out on the collision resistance of commonly used hash functions over the past few years. In 2005, Wang et al. proposed an identical-prefix collision attack on MD5 using modular differential [3]. This attack is based on differential cryptanalysis, which is also an essential method for analyzing block ciphers. Linear cryptanalysis, extensively used in attack block ciphers [4], is another powerful method. The key of the attack of Wang et al. is constructing differential paths that are a description of how the differences should propagate through chaining values. The limitation of the identical-prefix collision attack is that the message blocks before the colliding message need to be identical. Later, Stevens et al. overcame the shortcoming [5]. They introduced the chosen-prefix collision attack in that prefix message blocks can be chosen arbitrarily. Such an attack is more dangerous than the identical-prefix collision attack because they created two X.509 certificates with the same signature or even a rogue CA certificate [6]. Another proof of the harmfulness of the collision attack is Flame's malicious certificate. It was forged using the chosen-prefix collision attack on MD5 [7]. In addition, the adaptive chosen-prefix collision attack was used to construct a distinguishing attack on HMAC/NMAC-MD5 with the pseudo-collision [8].

To defend against collision attacks, Stevens introduced counter-cryptanalysis, namely detecting whether a given message block is one of a carefully constructed colliding message pair [7, 9]. The average complexity of Stevens' algorithm to detect the collision of MD5 and SHA-1 is about 224 times and 15 times the hashing of a message block, respectively. Later, Stevens and Shumow greatly reduced the detection complexity of SHA-1 utilizing the unavoidable bit conditions in the message generated from disturbance vectors [10]. The complexity is only 1.96 times a hashing of SHA-1. They left a public problem: how to determine unavoidable conditions for MD5. As a response, Shen et al. analyzed the properties of the Boolean function of the last round of MD5 in detail [11]. They concluded that the last round's sufficient conditions combinations (SCCs) were appropriate as the unavoidable conditions. According to SCCs, the 223 classes are divided into several sets, and 126 classes have adequate SCCs to be identified. As a result, their collision detection algorithm costs only 28.6% runtime of Stevens' algorithm. With the existing SCCs, nine sets cannot be further subdivided, and 79 classes cannot be classified. In particular, the undistinguishable 79 classes significantly affect detection complexity. Thus, a challenging problem is subdividing sets and classifying more classes.

Our contribution. In this paper, we carefully study the properties of the Boolean function of the second round and discover that the sufficient conditions (SCs) in the second round are also suitable to be unavoidable conditions. Note that the closer to the first round of MD5, the more complex the conditions are. There may be no SCs in the opening steps of the second round that can be used for distinguishing classes. Thus, we only consider SCs between steps 31 and 28, termed the additional sufficient conditions (ASCs). After analyzing all nine sets and 79 classes, we find that three sets can be subdivided, and seven classes in the non-distinguishable set can be distinguished, as they have enough SCs when combining ASCs and existing SCCs. We propose a new collision detection algorithm implemented by integrating ASCs into Shen's algorithm. Furthermore, we use "HashClash" to confirm that the ASCs exist in the differential paths of MD5. Eventually, compared to Shen's algorithm, the complexity of our algorithm to detect collision of MD5 is reduced by 8.18%.

The remainder of the paper is organized as follows. In Section 2, we give a short description of

MD5 and summarize the related works. The properties of the Boolean function of the second round and ASCs are discussed in Section 3. In Section 4, we present the verification of ASCs and the complexity of the new detection algorithm. The paper is summarized in Section 5.

2. Preliminaries

Table 1. Comparison of collision detection methods for MD5.

Algorithm	Method	Complexity
Stevens' algorithm	Check 223 classes in sequence	224 times hashing a message block
Shen's algorithm	1) Use SCCs to classify 126 classes into 22 elements 2) Determine the element according to SCCs and check the classes in the element 3) Check the remaining 76 classes	72.2949 times hashing a message block (under our experimental setting)
Our algorithm	1) Use SCCs and ASCs to classify 133 classes into 30 elements 2) Determine the element according to SCCs and ASCs, check the classes in the element 3) Check the remaining 69 classes	66.3834 times hashing a message block

2.1. Description of MD5

MD5 is one of the classical hash functions. The compression function receives a 128-bit intermediate hash value ($IHV_{in} = (a, b, c, d)$) and a 512-bit message block M and outputs a new 128-bit intermediate hash value (IHV_{out}). M is divided into consecutive 32-bit words $(m_0, m_1, \ldots, m_{15})$. The compression function consists of 64 steps which are split into four rounds that contain 16 steps each. For $t = 0, 1, \ldots, 63$ each step is as follows:

$$F_t = f_t(Q_t, Q_{t-1}, Q_{t-2}) \tag{2.1}$$

$$Q_{t+1} = Q_t + RL(F_t + Q_{t-3} + C_t + m_t, R_t) \tag{2.2}$$

where $(Q_0, Q_{-1}, Q_{-2}, Q_{-3}) = (b, c, d, a)$ and $IHV_{out} = (a + Q_{61}, b + Q_{64}, c + Q_{63}, d + Q_{62})$. In each step t, f_t is the corresponding Boolean function, C_t is a fixed constant, m_t is the corresponding 32-bit word, RL denotes left-rotation and R_t is the rotation constant. Q_{t-3} can be calculated using the following

formula in which *RR* denotes right rotation:

$$Q_{t-3} = RR(Q_{t+1} - Q_t, R_t) - F_t - m_t - C_t \tag{2.3}$$

One can find more details of MD5 in [12].

2.2. Related works

This subsection discusses the related works and compares the related methods with our approach. The comparison results can be found in Table 1.

2.2.1. Stevens' collision detection algorithm

In [7, 9], Stevens introduced a collision detection algorithm. The algorithm builds on two observations. One is that there are fewer message differences that can cause collision attacks. The other is that the current differential path used in collision attacks has fixed differences in the working state of some steps. For MD5, there are 223 classes of message difference δM, step t, and working state $\delta WS_t = (Q_{t-3}, Q_{t-2}, Q_t, Q_{t+1})$. One can find more details in Algorithm 1. The average detection complexity is 224 times the complexity of the compression operator of MD5.

Algorithm 1 Stevens' collision detection algorithm

Input: padded and split message blocks $M_0, ..., M_{N-1}$.
Output: `True` if a near-collision or pseudo-collision is detected and `False` otherwise.

Let IHV_0 be the initial chaining values.
for k=0 to N-1 **do**
  $WS_0 = IHV_k$.
  Calculate all working states $WS_1, ..., WS_{64}$ and $IHV_{k+1} = IHV_k + WS_{64}$.
  for each $(\delta M, t, \delta WS_t)$ **do**
    Let $M'_k = M_k + \delta M$ and $WS'_t = WS_t + \delta WS_t$.
    Calculate $WS'_{t-1}, ..., WS'_0$ backward and $WS'_{t+1}, ..., WS'_{64}$ forward.
    Let $IHV'_k = WS'_0$ and $IHV'_{k+1} = IHV'_k + WS'_{64}$.
    if $IHV'_{k+1} = IHV_{k+1}$ **then**
      return `True`. //M_k and M'_k is a near-collision or pseudo-collision block pair
    end if
  end for
end for
return `False`

2.2.2. Shen's collision detection algorithm

In [11], Shen et al. proposed an improved collision detection algorithm of MD5 using unavoidable conditions. They used the properties of MD5 and found that the SCs in the last round can be seen as unavoidable conditions. It is based on the observation that the corresponding SCs always remain the same once the input differences are fixed. Based on the SCCs, the 223 different classes are split into four sets, namely, distinguishable set (DS), individual checked set (ICS), non-distinguishable set (NDS), and discard set. The classes within each element in DS have identical SCCs. The same applies

to the classes within each element in ICS. The SCs at the 31th bit are called main SCs (mSCs). The SCs at the 5th, 9th, 14th, or 20th bit are called auxiliary SCs (aSCs). Elements in NDS do not have enough SCs, and elements in the discard set are not appropriate for building a collision attack. The details of the algorithm are described in Algorithm 2. 126 classes in DS and ICS contribute to reducing the complexity of collision detection by 71.4%. Note that each working state difference is converted to $(\delta Q_{38}, \delta Q_{39}, \delta Q_{40}, \delta Q_{41}) = (0, 0, 0, 0)$ (type I difference) or $(\delta Q_{38}, \delta Q_{39}, \delta Q_{40}, \delta Q_{41}) = (2^{31}, 2^{31}, 2^{31}, 2^{31})$ (type II difference) in [11], where $\delta X = X' - X$ is the modular difference for 32-bit words X and X'. Elements containing multiple message differences are as follows:

1) DS1 using type II difference: $\delta M = 0$, $\delta M = \pm (\delta m_{11} = 2^b)$ $(b \in \{0, \ldots, 31\})$, and $\delta M = \pm (\delta m_4 = 2^b)$ $(b \in \{20, 25, 31\})$.
2) DS2 using type II difference: $\delta M = (\delta m_8 = 2^{31})$ and $\delta M = \pm (\delta m_8 = 2^{31}, m_{11} = 2^{21})$.
3) DS3 using type II difference: $\delta M = (\delta m_5 = 2^{31})$ and $\delta M = (\delta m_5 = 2^{31}, \delta m_{11} = 2^{31})$.
4) DS4 using type II difference: $\delta M = (\delta m_{14} = 2^{31})$ and $\delta M = \pm (\delta m_{11} = 2^{15}, \delta m_4 = \delta m_{14} = 2^{31})$.
5) DS9 using type I difference: $\delta M = \pm (\delta m_2 = 2^8, \delta m_{14} = 2^{31})$, and $\delta M = \pm (\delta m_2 = 2^8, \delta m_{11} = 2^{15}, \delta m_4 = \delta m_{14} = 2^{31})$.
6) DS13 using type I difference: $\delta M = \pm (\delta m_5 = 2^{10})$, $\delta M = \pm (\delta m_5 = 2^{10}, \delta m_{11} = 2^{21})$, and $\delta M = \pm (\delta m_5 = 2^{10}, \delta m_{11} = 2^{31})$.
7) ICS2 using type I difference: $\delta M = (\delta m_{14} = 2^{31})$ and $\delta M = \pm (\delta m_{11} = 2^{15}, \delta m_4 = \delta m_{14} = 2^{31})$.
8) ICS3 using type I difference: $\delta M = (\delta m_5 = 2^{31})$, $\delta M = (\delta m_5 = 2^{31}, \delta m_{11} = 2^{31})$, and $\delta M = (\delta m_5 = 2^{31}, \delta m_8 = 2^{31})$.
9) ICS8 using type II difference: $\delta M = \pm (\delta m_5 = 2^{10})$, $\delta M = \pm (\delta m_5 = 2^{10}, \delta m_{11} = 2^{21})$, and $\delta M = \pm (\delta m_5 = 2^{10}, \delta m_{11} = 2^{31})$.

The classes in NDS are as follows:

1) Using type I difference: $\delta M = \pm (\delta m_{11} = 2^b)$ $(b \in \{0, \ldots, 31\})$, $\delta M = \pm (\delta m_4 = 2^b)$ $(b \in \{20, 25, 31\})$, $\delta M = \pm (\delta m_8 = 2^b)$ $(b \in \{25, 31\})$, and $\delta M = \pm (\delta m_8 = 2^{31}, \delta m_{11} = 2^{21})$.
2) Using type II difference: $\delta M = \pm (\delta m_2 = 2^8, \delta m_{14} = 2^{31})$, $\delta M = \pm (\delta m_6 = 2^8, \delta m_9 = \delta m_{15} = 2^{31})$, and $\delta M = \pm (\delta m_2 = 2^8, \delta m_{11} = 2^{15}, \delta m_4 = \delta m_{14} = 2^{31})$.

Algorithm 2 Shen's collision detection algorithm

Input: padded and split message blocks $M_0, ..., M_{N-1}$.

Output: `True` if a near-collision or pseudo-collision is detected and `False` otherwise.

Let IHV_0 be the initial chaining values.
for k=0 to N-1 **do**
  $(Q_{-3}, Q_0, Q_{-1}, Q_{-2}) = IHV_k$.
  Calculate $Q_1, ..., Q_{64}$ and IHV_{k+1}.
  Obtain $mSCs = Q_{46}[31]||...||Q_{59}[31]$ and $aSCs_b = Q_{46}[b]||...||Q_{59}[b]$ $(b = 5, 9, 14, 20)$.
  if mSCs is one of the SCs of a element E in DS or mSCs and $aSCs_b$ are one of the SCs of a element E in ICS **then**
    for each $(\delta M, \delta WS_{41})$ in E **do**
      Calculate IHV'_{k+1} and IHV_{k+1} using δM and δWS_{41}.
      if $IHV'_{k+1} = IHV_{k+1}$ **then**
        return `True`.
      end if
    end for
  end if
  for each element E in NDS **do**
    for each $(\delta M, \delta WS_{41})$ in E **do**
      Calculate IHV'_{k+1} and IHV_{k+1} using δM and δWS_{41}.
      if $IHV'_{k+1} = IHV_{k+1}$ **then**
        return `True`.
      end if
    end for
  end for
end for
return `False`

3. Improved classification

3.1. Properties of MD5

In this subsection, we describe the properties of the Boolean function of the second round. Let $F(A, B, C) = (C \wedge A) \oplus (\overline{C} \wedge B)$ and $F(A', B', C') = (C' \wedge A') \oplus (\overline{C'} \wedge B')$ where A, B, C, A', B' and C' are 1-bit Boolean variables. Let $\Delta K = K' - K$ $(K \in \{A, B, C\})$ and $\Delta F = F(A', B', C') - F(A, B, C)$. We can get the following property.

Property 1. If $\Delta A \neq 0$, $\Delta B = 0$, and $\Delta C = 0$, then $\Delta F = 0$ if and only if $C = C' = 0$. If $\Delta A = 0$, $\Delta B \neq 0$, and $\Delta C = 0$, then $\Delta F = 0$ if and only if $C = C' = 1$. If $\Delta A = 0$, $\Delta B = 0$, and $\Delta C \neq 0$, then $\Delta F = 0$ if and only if $A = B$. If $\Delta A \neq 0$, $\Delta B \neq 0$, and $\Delta C \neq 0$, then $\Delta F = 0$ if and only if $\Delta A \neq \Delta B$, $\Delta F = 1$ or -1 if and only if $\Delta A = \Delta B$.

Proof. According to the truth table of ΔF, the property is easily derived. Note that $\Delta A = \Delta B$ means that $A = B$ and $A' = B'$; $\Delta A \neq \Delta B$ means that $A \neq B$ and $A' \neq B'$.

When bit length is extended to 32-bit, the properties of the Boolean function $F(Q_t, Q_{t-1}, Q_{t-2}) = (Q_{t-2} \wedge Q_t) \oplus (\overline{Q_{t-2}} \wedge Q_{t-1})$ can be summarized as follows based on property 1. Let $\Delta \mathrm{X} = (X'[i] - X[i])_{i=0}^{N-1}$ be the signed bitwise difference and $X[i]$ indicate to take the i-th bit of X.

Property 2. If $(\delta Q_t, \delta Q_{t-1}, \delta Q_{t-2}) = (2^b, 0, 0)$ and $\delta F_t = 0$, then $Q_{t-2}[b] = 0$.

Proof. $(\delta Q_t, \delta Q_{t-1}, \delta Q_{t-2}) = (2^b, 0, 0)$ means that $\Delta Q_t[b] \neq 0$, $\Delta Q_{t-1}[b] = 0$ and $\Delta Q_{t-2}[b] = 0$. $\delta F_t = 0$ means that $\Delta F_t[i] = 0$ $(0 \leq i \leq 31)$. According to property 1, $Q_{t-2}[b] = 0$.

Property 3. If $(\delta Q_t, \delta Q_{t-1}, \delta Q_{t-2}) = (0, 2^b, 0)$ and $\delta F_t = 0$, then $Q_{t-2}[b] = 1$.

Proof. $(\delta Q_t, \delta Q_{t-1}, \delta Q_{t-2}) = (0, 2^b, 0)$ means that $\Delta Q_t[b] = 0$, $\Delta Q_{t-1}[b] \neq 0$ and $\Delta Q_{t-2}[b] = 0$. $\delta F_t = 0$ means that $\Delta F_t[i] = 0$ $(0 \leq i \leq 31)$. According to property 1, $Q_{t-2}[b] = 1$.

Property 4. If $(\delta Q_t, \delta Q_{t-1}, \delta Q_{t-2}) = (0, 0, 2^b)$ and $\delta F_t = 0$, then $Q_t[b] = Q_{t-1}[b]$.

Proof. $(\delta Q_t, \delta Q_{t-1}, \delta Q_{t-2}) = (0, 0, 2^b)$ means that $\Delta Q_t[b] = 0$, $\Delta Q_{t-1}[b] = 0$ and $\Delta Q_{t-2}[b] \neq 0$. $\delta F_t = 0$ means that $\Delta F_t[i] = 0$ $(0 \leq i \leq 31)$. According to property 1, $Q_t[b] = Q_{t-1}[b]$.

Property 5. If $(\delta Q_t, \delta Q_{t-1}, \delta Q_{t-2}) = (2^b, 2^b, 2^b)$ and $\delta F_t = 0$, then $\Delta Q_t[b] \neq \Delta Q_{t-1}[b]$.

Proof. $(\delta Q_t, \delta Q_{t-1}, \delta Q_{t-2}) = (2^b, 2^b, 2^b)$ means that $\Delta Q_t[b] \neq 0$, $\Delta Q_{t-1}[b] \neq 0$ and $\Delta Q_{t-2}[b] \neq 0$. $\delta F_t = 0$ means that $\Delta F_t[i] = 0$ $(0 \leq i \leq 31)$. According to property 1, $\Delta Q_t[b] \neq \Delta Q_{t-1}[b]$.

Property 6. If $(\delta Q_t, \delta Q_{t-1}, \delta Q_{t-2}) = (2^b, 2^b, 2^b)$ and $\delta F_t = \pm\, 2^b$, then $\Delta Q_t[b] = \Delta Q_{t-1}[b]$.

Proof. $(\delta Q_t, \delta Q_{t-1}, \delta Q_{t-2}) = (2^b, 2^b, 2^b)$ means that $\Delta Q_t[b] \neq 0$, $\Delta Q_{t-1}[b] \neq 0$ and $\Delta Q_{t-2}[b] \neq 0$. $\delta F_t = \pm\, 2^b$ means that $\Delta F_t[b] \neq 0$. According to property 1, $\Delta Q_t[b] = \Delta Q_{t-1}[b]$.

3.2. Additional sufficient conditions

The conditions in the differential path must be satisfied for a collision attack. Thus, the success probability of a collision attack depends on the number of conditions. It is easy to fulfill almost all conditions in the first 25 steps with message modification techniques. Conditions in the remaining steps are randomly satisfied. We only choose SCs between steps 31 and 28 as ASCs. Because conditions in the second round are more complex than conditions in the last round, we choose the part with relatively few simple conditions. In this subsection, we describe how to deduce ASCs.

Due to the fixed chaining value difference, we start from step 40 using Eq (2.3) to calculate the Q_{t-3} reversely with linearizing step 40 to step 32, and we deduce the conditions from step 31 to step 28. In Eq (2.3), if $\delta Q_{t+1} - \delta Q_t \neq 0$ then $RR(\delta Q_{t+1} - \delta Q_t, R_t)$ will rotate the difference to another bit position, which may increase the number of conditions. In the following, we refer to $\delta Q_{t+1} - \delta Q_t \neq 0$ as the subtraction difference. Support $\delta Q_t = \pm\, 2^b$ and δQ_k $(k \in \{\ldots, t-1\} \cup \{t+1, \ldots\})$ is the chaining values adjacent to Q_t. There are roughly two ways to deal with the subtraction difference:

1) If there is no difference at bit position b in δQ_k, then a good choice is to eliminate the difference $\pm\, 2^b$ at steps $t+2$, $t+1$, and t according to properties 2–4, respectively. If not eliminated, the number of conditions will increase owing to the introduced difference at bit position b. Note that the subtraction difference occurs twice in either case.

2) If there are consecutive differences at bit position b in δQ_k, then we can eliminate the subtraction difference according to properties 5 and 6, namely, keeping the difference at bit position b. Thus, no conditions will be introduced because of the subtraction difference except for the initial and last subtraction difference.

According to the two ways, we deduce the conditions of a high probability differential path with properties 2 to 6. Note that the high bit position conditions may be eliminated because of carrying expansion.

Depending on ASCs, we find that DS9, ICS2, and ICS3 can be subdivided. DS1, DS2, DS3, DS4, DS13, and ICS8 cannot be subdivided. The reasons are as follows. The conditions of elements in DS1 are too complicated. After removing the repeated SCs, the number of remaining conditions of elements in DS2, DS3, and ICS8 is less than two. Besides, DS4 and DS13 have less than three SCs.

The elements in NDS that have enough SCs to be distinguished are as follows: $\delta M = \pm (\delta m_8 = 2^{25})$ including two classes using type I difference, $\delta M = (\delta m_8 = 2^{31})$ including one class using type I difference, $\delta M = \pm(\delta m_6 = 2^8, \delta m_9 = \delta m_{15} = 2^{31})$ including two classes using type II difference, and $\delta M = \pm (\delta m_2 = 2^8, \delta m_{14} = 2^{31})$ including two classes using type II difference. All of them may have SCs located at the 31th, 27th, 22th, 17th, 11th, or 8th bits. The ASCs at the 31th bit are called main ASCs. The ASCs at other bit locations are called auxiliary ASCs. The other elements in NDS do not have enough SCs. The conditions of $\delta M = \pm (\delta m_{11} = 2^b)$ $(b \in \{0, \ldots, 31\})$, $\delta M = \pm (\delta m_4 = 2^b)$ $(b \in \{20, 25, 31\})$, and $\delta M = \pm (\delta m_8 = 2^{31}, \delta m_{11} = 2^{21})$ $(b \in \{0, \ldots, 31\})$ using type I difference are too complex, and the number of SCs is less than four. $\delta M = \pm (\delta m_2 = 2^8, \delta m_{11} = 2^{15}, \delta m_4 = \delta m_{14} = 2^{31})$ using type II difference has just four SCs.

3.2.1. The subdivided set

We add the classes in DS9, ICS2, and ICS3 to the subdivided set, and the details are as follows.

1) DS9 element has four classes and uses type I difference. It is divided into two subsets:
   (a) DS9_1. For the message differences $\delta M = \pm (\delta m_2 = 2^8, \delta m14 = 2^{31})$, chaining value differences are $\delta Q_t = 2^{31}$ $(31 \le t \le 32)$, $\delta Q_t = 0$ $(t \in \{30, 28\})$, $\delta Q_{29} = \pm 2^{27}$, $\delta Q_{27} = \pm 2^{17}$, and unavoidable sufficient conditions are $Q_{30}[27] = Q_{31}[27]$, $Q_{27}[8] = Q_{28}[8]$, $Q_{28}[27] = 1$. Note that at step 29, $RR(\delta Q_{30} - \delta Q_{29}, 9) = \pm 2^{18}$ and δQ_{27} can be eliminated or retained; $\delta m_2 = \pm 2^8$ is introduced. Therefore, $\delta Q_{26} = \pm 2^{18} \pm 2^{17} \pm 2^8$ or $\pm 2^{18} \pm 2^8$. δQ_{25} has $\pm 2^{22}$ because of $\delta Q_{28} - \delta Q_{27} \neq 0$.
   (b) DS9_2. For the message differences $\delta M = \pm (\delta m_2 = 2^8, \delta m_{11} = 2^{15}, \delta m_4 = \delta m_{14} = 2^{31})$, chaining value differences are $\delta Q_t = 2^{31}$ $(27 \le t \le 35)$, $\delta Q_{26} = 2^{31} \pm 2^8$, $\delta Q_{25} = \pm 2^{31}$, and unavoidable sufficient conditions are $Q_t[31] = Q_{t+1}[31]$ $(27 \le t \le 30)$, $Q_{27}[8] = Q_{28}[8]$.
2) ICS2 element has three classes and uses type I difference. It is divided into two subsets:
   (a) ICS2_1. For the message difference $\delta M = (\delta m_{14} = 2^{31})$, chaining value differences are $\delta Q_t = 2^{31}$ $(31 \le t \le 32)$, $\delta Q_t = 0$ $(t \in \{30, 28\})$, $\delta Q_{29} = \pm 2^{27}$, $\delta Q_{27} = \pm 2^{17}$, and unavoidable sufficient conditions are $Q_{30}[27] = Q_{31}[27]$, $Q_{28}[27] = 1$. Note that the calculation of δQ_{26} and δQ_{25} is similar to the calculation in DS9_1 except for no $\delta m_2 = \pm 2^8$.
   (b) ICS2_2. For the message difference $\delta M = \pm (\delta m_{11} = 2^{15}, \delta m_4 = \delta m_{14} = 2^{31})$, chaining value differences are $\delta Q_t = 2^{31}$ $(25 \le t \le 34)$, and unavoidable sufficient conditions are $Q_t[31] = Q_{t+1}[31]$ $(27 \le t \le 30)$.

3) ICS3 element has three classes and uses type I difference. It is divided into three subsets:
   (a) ICS3_1. For the message difference $\delta M = (\delta m_5 = 2^{31})$, chaining value differences are $\delta Q_t = 0$ ($t \in \{30, 31\}$), $\delta Q_t = 2^{31}$ ($27 \le t \le 29$), $\delta Q_{26} = 2^{31} \pm 2^{22}$, $\delta Q_{25} = 2^{31}$, and unavoidable sufficient conditions are $Q_{30}[31] \neq Q_{31}[31]$, $Q_t[31] = Q_{t+1}[31]$ ($27 \le t \le 28$), $Q_{27}[22] = Q_{28}[22]$.
   (b) ICS3_2. For the message difference $\delta M = (\delta m_5 = 2^{31}, \delta m_{11} = 2^{31})$, chaining value differences are $\delta Q_t = 2^{31}$ ($26 \le t \le 27,\ 29 \le t \le 31$), $\delta Q_{28} = 2^{31} \pm 2^{11}$, $\delta Q_{25} = 2^{31} \pm 2^6$, and unavoidable sufficient conditions are $Q_t[31] = Q_{t+1}[31]$ ($27 \le t \le 30$), $Q_{29}[11] = Q_{30}[11]$, $Q_{27}[11] = 1$, $Q_{26}[11] = 0$.
   (c) ICS3_3. For the message difference $\delta M = (\delta m_5 = 2^{31}, \delta m_8 = 2^{31})$, chaining value differences are $\delta Q_{30} = 2^{31}$, $\delta Q_{27} = \pm 2^{17}$, $\delta Q_{26} = \pm 2^{22}$, $\delta Q_t = 0$ ($t \in \{25, 28, 29, 31\}$), and unavoidable sufficient conditions are $Q_{29}[31] = 1$, $Q_{28}[31] = 0$, $Q_{28}[17] = Q_{29}[17]$, $Q_{26}[17] = 1$.

3.2.2. The new individual checked set

We add seven classes to the new individual checked set (NICS). The SCs of each element consist of the SCCs in the last 16 steps and ASCs to ensure enough SCs. The details of the new elements are as follows:

1) NICS1 element has two classes and uses type I difference. Message difference is $\delta M = \pm (\delta m_8 = 2^{25})$. Chaining value differences are $\delta Q_t = 2^{31}$ ($57 \le t \le 59$), $\delta Q_t = \pm 2^{25}$ ($29 \le t \le 30$). Unavoidable sufficient conditions are $Q_{57}[31] \neq Q_{59}[31]$, $Q_{56}[31] = 1$, $Q_{55}[31] = 0$, $Q_{28}[11] = Q_{29}[11]$, $Q_{26}[11] = 1$. Note that δQ_{27} has $\pm 2^{11}$ owing to $\delta Q_{31} - \delta Q_{30} \neq 0$. Because at step 31, we get $\delta Q_{28} = \pm 2^{25}$ or 0, the differences of δQ_t ($25 \le t \le 28$) at 25th bit are uncertain.
2) NICS2 element has one class and uses type I difference. Message difference is $\delta M = (\delta m_8 = 2^{31})$. Chaining value differences are $\delta Q_t = 2^{31}$ ($25 \le t \le 26,\ 28 \le t \le 30$), $\delta Q_t = \pm 2^5$ ($57 \le t \le 59$), $\delta Q_{27} = 2^{31} \pm 2^{17}$. Unavoidable sufficient conditions are $Q_{57}[5] \neq Q_{59}[5]$, $Q_{56}[5] = 1$, $Q_{55}[5] = 0$, $Q_t[31] = Q_{t+1}[31]$ ($27 \le t \le 29$), $Q_{28}[17] = Q_{29}[17]$, $Q_{26}[17] = 1$.
3) NICS3 element has two classes and uses type II difference. Message difference is $\delta M = \pm (\delta m_6 = 2^8, \delta m_9 = \delta m_{15} = 2^{31})$. Chaining value differences are $\delta Q_{59} = \pm 2^{23} \pm 2^9$, $\delta Q_{58} = \pm 2^9$, $\delta Q_t = 2^{31}$ ($25 \le t \le 43$). Unavoidable sufficient conditions are $Q_{57}[9] = 1$, $Q_{56}[9] = 0$, $Q_t[31] = Q_{t+1}[31]$ ($27 \le t \le 30$).
4) NICS4 element has two classes and uses type II difference. Message difference is $\delta M = \pm (\delta m_2 = 2^8, \delta m_{14} = 2^{31})$. Chaining value differences are $\delta Q_t = 2^{31}$ ($30 \le t \le 47$), $\delta Q_{29} = 2^{31} \pm 2^{27}$, $\delta Q_{27} = \pm 2^{17}$, $\delta Q_{26} = \pm 2^8$, $\delta Q_{25} = \pm 2^{22}$. Unavoidable sufficient conditions are $Q_{49}[31] = 0$, $Q_{48}[31] = 1$, $Q_{30}[27] = Q_{31}[27]$, $Q_{28}[27] = 1$, $Q_{27}[8] = Q_{28}[8]$. The difference on 31th bit of δQ_t ($25 \le t \le 28$) are uncertain. Because at step 29, $\delta Q_{27} = \pm 2^{17}$ can be eliminated or retained, the differences of δQ_{26} at 17th bit are undetermined.

3.3. Our new algorithm and implementation

In this subsection, we discuss how ASCs can be integrated into Shen's algorithm. Our new algorithm is described in Algorithm 3. Details are as follows. In [11], all possible bit values are listed in a table using a hexadecimal representation; the key bits determine the rest of the bits; the bit mask indicates those positions where the SCs are. We use the same method to represent ASCs. Because

there are only 6 bits, we use binary to represent them. The most significant bit is $\delta Q_{26}[b]$, and the least significant bit is $\delta Q_{31}[b]$ ($b \in \{31, 27, 22, 17, 11, 8\}$). For example, ASCs of ICS_2 are $Q_t[31] = Q_{t+1}[31]$ ($27 \leq t \leq 30$). They are respected by two binary values, 0b000000 and 0b011111. The bit mask is 0b011111. We choose $Q_{30}[31]$ and $Q_{31}[31]$ as key bits. Obviously, $Q_{30}[31] = Q_{31}[31] = 0$ indicates 0b000000; $Q_{30}[31] = Q_{31}[31] = 1$ indicates 0b011111. Two bits are used as they determine up to four different values.

Algorithm 3 Algorithm integrated with ASC

Input: padded and split message blocks $M_0, ..., M_{N-1}$.

Output: `True` if a near-collision or pseudo-collision is detected and `False` otherwise.

Let IHV_0 be the initial chaining values.
for k=0 to N-1 **do**
  $(Q_{-3}, Q_0, Q_{-1}, Q_{-2}) = IHV_k$.
  Calculate $Q_1, ..., Q_{64}$ and IHV_{k+1}.
  Obtain $mSCs = Q_{46}[31]||...||Q_{59}[31]$, $aSCs_b = Q_{46}[b]||...||Q_{59}[b]$ ($b = 5, 9, 14, 20$) and $ASCs_b = Q_{26}[b]||...||Q_{31}[b]$ ($b = 31, 27, 22, 17, 11, 8$).
  if mSCs is one of the SCs of a element E in DS or mSCs and $aSCs_b$ are one of the SCs of a element E in ICS or $mSCs$, $aSCs_b$ and $ASCs_b$ is one of the SCs of a element E in NICS **then**
    if E belongs to the Subdivided Set **then**
      according to $ASCs_b$, choose δM and δWS_{41} and calculate IHV'_{k+1} and IHV_{k+1}.
      if $IHV'_{k+1} = IHV_{k+1}$ **then**
        return `True`.
      end if
    else
      for each $(\delta M, \delta WS_{41})$ in E **do**
        Calculate IHV'_{k+1} and IHV_{k+1} using δM and δWS_{41}.
        if $IHV'_{k+1} = IHV_{k+1}$ **then**
          return `True`.
        end if
      end for
    end if
  end if
  for each element E in NDS **do**
    for each $(\delta M, \delta WS_{41})$ in E **do**
      Calculate IHV'_{k+1} and IHV_{k+1} using δM and δWS_{41} in E.
      if $IHV'_{k+1} = IHV_{k+1}$ **then**
        return `True`.
      end if
    end for
  end for
end for
return `False`

For the elements in the subdivided set, conditions are specified using key bits according to the table of SCCs in [11]. Then, ASCs are used to determine which message difference these conditions belong to. Therefore, it is no longer necessary to calculate the IHV_{out} of all messages difference, in turn, to detect a collision. ASCs of the subdivided set are listed in Table 2. The column where the value of SCs is located is determined by the key bits. For instance, the key bits are 11, which means column 3. Note that the condition $Q_{27}[8] = Q_{28}[8]$ exists both in DS9_1 and DS9_2, so it is not used. DS9 and ICS2 conditions are identical in Table 2, so they share a 2×5 array in implementation. ICS3 conditions are saved in a 6×5 array. For the key bits, ICS3_1* and ICS3_2† are $Q_{29}[b]$ and $Q_{30}[b]$; ICS3_1† and ICS3_3† are $Q_{27}[b]$ and $Q_{28}[b]$; ICS3_3* are $Q_{28}[b]$ and $Q_{29}[b]$; all others are $Q_{30}[b]$ and $Q_{31}[b]$.

Table 2. ASCs of the subdivided set.

Element	C[i][0]	C[i][1]	C[i][2]	C[i][3]	Bit mask
DS9_1	0b001000	-1	-1	0b001011	0b001011
DS9_2	0b000000	-1	-1	0b011111	0b011111
ICS2_1	0b001000	-1	-1	0b001011	0b001011
ICS2_2	0b000000	-1	-1	0b011111	0b011111
ICS3_1*	0b000001	0b000010	0b011101	0b011110	0b011111
ICS3_1†	0b000000	-1	-1	0b011000	0b011000
ICS3_2*	0b000000	-1	-1	0b011111	0b011111
ICS3_2†	0b010000	-1	-1	0b010110	0b110110
ICS3_3*	-1	0b000100	-1	-1	0b001100
ICS3_3†	0b100000	-1	-1	0b111000	0b111000

* is the main ASCs. † is the auxiliary ASCs. -1 denotes that there is no value.

We add NICS to the end of ICS in the original algorithm. Namely, after checking ICS, the algorithm will process the elements in NICS. SCs in NICS are listed in Table 3. Note that the values of NICS1‡ and NICS1† are identical to NICS2‡ and NICS2†, respectively. They are merged in the implementation. For the key bits, NICS1†, NICS2*, NICS2† and NICS3* are $Q_{28}[b]$ and $Q_{29}[b]$; NICS4* are $Q_{29}[b]$ and $Q_{30}[b]$; NICS4† are $Q_{27}[b]$ and $Q_{28}[b]$; NICS1‡ and NICS2‡ are $Q_{57}[b]$ and $Q_{59}[b]$.

Table 3. SCCs and ASCs of the new individual checked set.

Element	C[i][0]	C[i][1]	C[i][2]	C[i][3]	Bit mask
NICS1‡	-1	0x00009	0x0000c	-1	0x0001d
NICS1†	0b100000	-1	-1	0b101100	0b101100
NICS2‡	-1	0x00009	0x0000c	-1	0x0001d
NICS2*	0b000000	-1	-1	0b011110	0b011110
NICS2†	0b100000	-1	-1	0b101100	0b101100
NICS3‡	0x00004	-1	-1	-1	0x00004
NICS3*	0b000000	-1	-1	0b011111	0b011111
NICS4‡	0x20000	-1	-1	-1	0x20000
NICS4*	0b001000	-1	-1	0b001011	0b001011
NICS4†	0b000000	-1	-1	0b011000	0b011000

‡ is the SCCs of the last 16 steps.

4. Experiments

4.1. Verification of ASCs

We use "HashClash" developed by Stevens to verify whether ASCs exist in the differential path. "HashClash" using the MIT License is a tool to construct differential paths for MD5 and SHA-1, also including the chosen-prefix collision attack on MD5. It consists of three parts, i.e., forward extension, backward extension, and connection. One can find the source code of "HashClash" in [13].

To achieve the maximum complexity of conditions as much as possible, we set the parameter as follows:

1) $Autobalance = 1,000,000$. The parameter limits the maximum number of paths that can be saved per step;
2) $Estimate = 4$. The program will evaluate the maximum number of conditions under $4\times1,000,000$ available paths.
3) $Maxweight = 32$ and $Maxsdrs = 2000$. $Maxweight$ controls the carrying extension length; $Maxsdrs$ limits the total number of ΔQ. Such a setting means no limitation to carrying extension of δQ.

Other parameters are the default values. The backward program of MD5 is used to construct differential paths starting from step 40 to step 16. After observing the paths with the minimum and maximum conditions, we found that all ASCs existed. This result suggests that the conditions found are sufficient. One can find the results and our new algorithm code in https://github.com/Finsenty54/Improved-Collision-Detection-Of-MD5.

4.2. Complexity

The algorithm of Shen et al. takes 78.44 times the runtime of hashing a message block to detect collisions in theory. In this paper, as seven classes can be distinguished which make a major contribution to improving the detection efficiency, the total complexity of the new collision algorithm is about 71.44 times of computing the MD5 hash of a message block. Compared with the original algorithm, the complexity is reduced by 8.92%.

Table 4. Comparison of the original and our new algorithm.

Experiments	Generate random message [$\times 10^6/ms$]	Standard MD5 hash [$\times 10^6/ms$]	Collision detection [$\times 10^8/ms$]	Average check number [multiples of one hashing operation]
Original test1	3.27323	5.11324	1.35930	72.0957
Original test2	3.27849	5.10581	1.35466	72.3395
Original test3	3.26173	5.09832	1.36322	72.4496
New test1	3.22541	5.07214	1.26397	66.6971
New test2	3.25677	5.11430	1.27354	66.4313
New test3	3.21775	5.07999	1.26165	66.0214

The C++ code of the collision detection algorithm with ASCs was compiled by g++ 10.2.1 under -O3 optimization and was run on AMD Ryzen 5 2600 at 3.40 GHz under Parrot OS 4.11. We conducted three experiments on the original code of Shen et al. and our code, respectively. The total number of messages generated is 2^{24}. The results are shown in Table 4. The average checks number of the original algorithm is 72.2949 in contrast to 66.3834 of our new algorithm. The experiments show that it reduces the runtime by 8.18% with ASCs.

5. Conclusions

In this paper, we derive the sufficient conditions between steps 31 and 28 of MD5 based on the properties of the Boolean function of the second round and describe the reverse deduction ways. Combining the additional sufficient conditions, we improve Shen's detection algorithm by subdividing three sets and distinguishing seven more classes. After adding the subdivided set and NICS to their algorithm, we get our new detection algorithm. The cost of collision detection of MD5 is reduced by 8.18%. This study further supports the idea of using unavoidable conditions to accelerate collision detection in previous research. The experimental results of Shen's algorithm differ from those claimed in [11]. This inconsistency may be due to the various experimental setting.

Acknowledgments

This work was supported by the Zhejiang Provincial Natural Science Foundation of China [Grant No. LQ20F020019] and the Technology on Communication Security Laboratory [Grant No. 6142103190105].

Conflict of interest

All authors declare no conflicts of interest in this paper.

References

1. I. B. Damgård, A design principle for hash functions, in *Advances in Cryptology — CRYPTO' 89 Proceedings* (ed. G. Brassard), Springer, New York, NY, (1990), 416–427. https://doi.org/10.1007/0-387-34805-0_39
2. R. C. Merkle, One way hash functions and DES, in *Advances in Cryptology — CRYPTO' 89 Proceedings* (ed. G. Brassard), Springer, New York, NY, (1990), 428–446. https://doi.org/10.1007/0-387-34805-0_40
3. X. Wang, H. Yu, How to break MD5 and other hash functions, in *Advances in Cryptology – EUROCRYPT 2005* (ed. R. Cramer), Springer, Berlin, Heidelberg, (2005), 19–35. https://doi.org/10.1007/11426639_2
4. A. D. Dwivedi, S. Dhar, G. Srivastava, R. Singh, Cryptanalysis of round-reduced fantomas, robin and iSCREAM, *Cryptography*, **3** (2019), 4. https://doi.org/10.3390/cryptography3010004
5. M. Stevens, A. Lenstra, B. de Weger, Chosen-prefix collisions for MD5 and colliding X.509 certificates for different identities, in *Advances in Cryptology - EUROCRYPT 2007* (ed. M. Naor), Springer, Berlin, Heidelberg, (2007), 1–22. https://doi.org/10.1007/978-3-540-72540-4_1
6. M. Stevens, A. Sotirov, J. Appelbaum, A. Lenstra, D. Molnar, D. A. Osvik, et al., Short chosen-prefix collisions for MD5 and the creation of a rogue CA certificate, in *Advances in Cryptology - CRYPTO 2009* (ed. S. Halevi), Springer, Berlin, Heidelberg, (2009), 55–69. https://doi.org/10.1007/978-3-642-03356-8_4
7. M. Stevens, Counter-cryptanalysis, in *Advances in Cryptology – CRYPTO 2013* (eds. R. Canetti and J. A. Garay), Springer, Berlin, Heidelberg, (2013), 129–146. https://doi.org/10.1007/978-3-642-40041-4_8
8. X. Wang, H. Yu, W. Wang, H. Zhang, T. Zhan, Cryptanalysis on HMAC/NMAC-MD5 and MD5-MAC, in *Advances in Cryptology - EUROCRYPT 2009* (ed. A. Joux), Springer, Berlin, Heidelberg, (2009), 121–133. https://doi.org/10.1007/978-3-642-01001-9_7
9. M. Stevens, *Attacks on hash functions and applications*, Ph.D thesis, Leiden University in Leiden, 2012. https://doi.org/10.6100/ir749233
10. M. Stevens, D. Shumow, Speeding up detection of SHA-1 collision attacks using unavoidable attack conditions, in *26th USENIX Security Symposium (USENIX Security 17)*, USENIX Association, Vancouver, BC, (2017), 881–897. https://doi.org/10.5555/3241189.3241259

11. Y. Shen, T. Wu, G. Wang, X. Dong, H. Qian, Improved collision detection of MD5 using sufficient condition combination, *Comput. J.*, (2021). https://doi.org/10.1093/comjnl/bxab109
12. R. L. Rivest, The MD5 message-digest algorithm, *RFC*, **1321** (1992), 1–21. https://doi.org/10.17487/RFC1321
13. M. Stevens, Project HashClash - MD5 & SHA-1 cryptanalytic toolbox [Internet]. Available from: https://github.com/cr-marcstevens/hashclash.

AIMS Press