Finding collisions for MD4 hash algorithm using hybrid algorithm

Marko Carić

caric.marko@gmail.com

May 28, 2014

Abstract

The modification of message that meets the sufficient conditions for collision is found in the last step of differential attack proposed by Wang et all. (2005) on MD4 hash algorithm. Here we show how this attack phase, finding a collision starting from the list of sufficient conditions for the collision, can be implemented using a combination of two algorithms - evolutionary algorithm and hill climbing. Hybridization of evolutionary algorithm and hill climbing is a well-known technique for improving solutions, but it isn't applied to this domain (at least by information that author has collected).

Keywords— Evolutionary algorithm, hill climbing, hash algorithm, MD4, collision, differential attack

1 Introduction

A cryptographic hash algorithm for arbitrary message M computes fixed-size bit string *hash value* $h(M)$ such that any change of M significantly must change $h(M)$. For example, the hash algorithm MD4 [5] for an arbitrary message computes its hash value of 128 bits in length. The request that a good hash algorithm should meet is that for a given message M it is practically impossible to find another message M' such that $h(M) = h(M')$. Often, stricter requirement is set it should be virtually impossible to find a pair of messages (M, M') (collision) such that $h(M) = h(M')$. In the case of differential attack on a hash algorithm, we follow the evolution of two similar messages that pass through the algorithm. Differential attack on a hash algorithm can be divided into four phases:

1. Selection of message difference,
2. Finding the differential path,
3. Finding sufficient conditions for collision,
4. Finding messages that satisfy sufficient conditions.

The first two stages can be automated to some extent [8, 9]. One way of realizing the last phase is to gradually modify randomly selected message. The modification of the message in different rounds of the MD4 algorithm is different in terms of difficulty. Modification to satisfy the conditions of the first round is the easiest; it requires changing one message word (*basic modification*). Modification to satisfy the conditions of the second round is more complicated; as it must not disarrange the already fulfilled conditions of the first round, it is necessary to change more than one word of message (*advanced modification*). Typically, a brute force is used to find modification satisfying the conditions of the third round. That is why the tendency is to include the most conditions for the modification of message in the first round [3]. One way to realize message modification is achieved using SAT solvers [6]. Based od author's knowledge, there is no other effective way for solving this step.

This paper proposes a possibility of the realization of the last phase of the attack independently either by using evolutionary algorithms (EA) or by hill climbing, or by their combination. Alternating EA and hill climbing can improve the efficiency of EA while overcoming the lack of robustness of hill climbing.

2 Description of MD4

Let $B = \{0, 1\}$. MD4 algorithm for arbitrary message M calculates the hash value $h(M) \in B^{128}$. An iterative compression function $f : B^{128} \times B^{512} \rightarrow B^{128}$ is defined using functions $F, G, H : B^{32} \times B^{32} \times B^{32} \rightarrow B^{32}$:

$$F(X, Y, Z) = XY \vee \bar{X}Z = (X \wedge Y) \vee (\neg X \wedge Z)$$

$$G(X, Y, Z) = XY \vee XZ \vee YZ = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$$

$$H(X, Y, Z) = X \oplus Y \oplus Z$$

where $\wedge, \vee, \neg, \oplus$ are bit operations AND, OR, NEG, XOR applied 32 times to the corresponding bits of the arguments. Let $X + Y$, $X - Y$ denote the sum and difference X and Y modulo 2^{32}, and let $X \lll s$ be the result of the cyclical shift of the contents of X for s bits to the left. Let $t_i \in B^{32}$, $i = -3, -2, ...48$, $T = (t_{-3}, t_{-2}, t_{-1}, t_0) \in B^{128}$ and $M = (M_0, M_1, ...M_{15}) \in B^{128}$. Then $f(T, M) = (t_{-3} + t_{45}, t_{-2} + t_{46}, t_{-1} + t_{47}, t_0 + t_{48})$ where

$$t_i = (t_{i-4} + f_i(t_{i-3}, t_{i-2}, t_{i-1}) + M_{p_i} + c_i) \lll s_i, \quad i = 1, 2, ..., 48, \tag{1}$$

and values f_i, c_i, p_i, s_i, $i = 1, 2, ..., 48$, are shown in the following tables:

i	f_i	c_i	s_i															
1...16	F	$0x00000000$	3	7	11	19	3	7	11	19	3	7	11	19	3	7	11	19
17...32	G	$0x5a827999$	3	5	9	13	3	5	9	13	3	5	9	13	3	5	9	13
33...48	H	$0x6ed9eba1$	3	9	11	15	3	9	11	15	3	9	11	15	3	9	11	15

i	p_i															
1...16	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
17...32	0	4	8	12	1	5	9	13	2	6	10	14	3	7	11	15
33...48	0	8	4	12	2	10	6	14	1	9	5	13	3	11	7	15

Calculation of t_i, $i = 16(j-1) + k$, $k = 1, 2, ..., 16$ (1) makes the jth round of MD4, $j = 1, 2, 3$.

Before calculating hash value, the message M is first padded to the multiple of 512 by one, $447 - |M| \mod 512$ zeroes (where $a \mod b \in \{0, 1, ..., b-1\}$ denotes the remainder of a division of b by a) and 64 bits to represent a binary number $|M|$. Let expanded message be $M_0, M_1, ..., M_{n-1}$, $M_i \in B^{512}$, $i = 1, 2, ..., n-1$. The hash value of the message M is $h(M) = H_n$, where

$$H_0 = (67452301, 10325476, efcdab89, 98badcfe), \tag{2}$$

(hex), and $H_i = f(H_{i-1}, M_{i-1}), i = 1, 2, ..., n$.

3 Differential path and set of sufficient conditions

To find collisions for MD4 algorithm it is enough to find collisions for the compression function f, i.e. an arbitrary pair $(M, M') \in B^{512}$, $M = (M_0, M_1, ..., M_{15})$, $M' = (M'_0, M'_1, ..., M'_{15})$, such that $f(H_0, M) = f(H_0, M')$, where H_0 is a constant (2). This equation is equivalent to $(t_{45}, t_{46}, t_{47}, t_{48}) = (t'_{45}, t'_{46}, t'_{47}, t'_{48})$, where, according to (1)

$$\begin{aligned} t_i &= (t_{i-4} + f_i(t_{i-3}, t_{i-2}, t_{i-1}) + M_{p_i} + c_i) \lll s_i, \\ t'_i &= (t'_{i-4} + f_i(t'_{i-3}, t'_{i-2}, t'_{i-1}) + M'_{p_i} + c_i) \lll s_i, i = 1, 2, ..., 48, \end{aligned} \tag{3}$$

The *message difference* corresponding to collision (M, M') is a series of differences (modulo 2^{32}) $\Delta M = M' - M = (\Delta M_0, \Delta M_1, ..., \Delta M_{15}) = (M'_0 - M_0, M'_1 - M_1, ..., M'_{15} - M_{15})$. Let $t_i[j]$ and $t_i[-j]$ be the values obtained from t_i by changing the j-th bit; thereby, $t_i[j]$ means that the value of the j-th bit is changed from zero to one, while $t_i[-j]$ means that value of the jth bit is changed from one to zero. More generally, let $t_i[\pm j_1, \pm j_2, ..., \pm j_l]$ be obtained by changing the values of bits in indicated indexes. The *differential path* is a series of expressions of the form

$$t_i[\pm j_1, \pm j_2, ..., \pm j_l], l = l(i), i = 1, ..., 48 \tag{4}$$

where $l = 0$ for $i = 45, 46, 47, 48$. In order for a pair (M, M') to be a collision, a set of *sufficient conditions* for a pair (M, M') is determined by the differential (4) and conditions (3). This can be obtained starting from the properties of functions F, G i H. A message M that satisfies a set of sufficient conditions is called a *weak message.* If M is a weak message and $M' = M + \Delta M$, then the pair (M, M') is collision. In this paper we use two message differences: $\Delta m_4 = 2^2$, $\Delta m_i = 0, i \neq 4$ [1, 11] and $\Delta m_1 = 2^{31}$, $\Delta m_2 = 2^{31} - 2^{28}$, $\Delta m_{16} = -2^{16}$, $\Delta m_i = 0$, $i \neq 1, 2, 16$ [4, 9]. The first message difference corresponds to a set of 62 conditions [11], that is obtained starting from the differential path that consists of two branches:

$$t_5[6] \xrightarrow{3} t_9[-9, 10] \xrightarrow{3} t_{13}[12] \xrightarrow{3} t_{17}[15] \xrightarrow{5} t_{18}[20] \xrightarrow{5} t_{22}[25] \xrightarrow{5} t_{26}[30] \xrightarrow{5} t_{30}[3]$$

and

$$t_5[6] \xrightarrow{3} t_9[-9] \xrightarrow{11} t_{11}[-20] \xrightarrow{11} t_{15}[-31] \xrightarrow{9} t_{19}[-8] \xrightarrow{9} t_{23}[17, -18] \xrightarrow{9} t_{27}[-26] \xrightarrow{9} t_{31}[-3]$$

This differential path and corresponding 62 sufficient conditions for message M are shown in Table 3.

The difference $t_{19}[-8]$ cancels the difference $t_{19}[8]$ caused by the influence of ΔM_4 in the second round. The difference $t_{23}[-18]$ cancels difference that occurs as a continuation of the changes $t_{17}[15]$ in the first round. Similarly, if sufficient conditions are satisfied in steps:

$$t_{32} \leftarrow ((t_{28} + G(t_{31}, t_{30}, t_{29}) + M_{15} + 5A827999) \lll 13)$$

$$t_{33} \leftarrow ((t_{29} + H(t_{32}, t_{31}, t_{30}) + M_0 + 6ED9EBA1) \lll 3)$$

$$t_{34} \leftarrow ((t_{30} + H(t_{33}, t_{32}, t_{31}) + M_8 + 6ED9EBA1) \lll 9)$$

$$t_{35} \leftarrow ((t_{31} + H(t_{34}, t_{33}, t_{32}) + M_4 + 6ED9EBA1) \lll 11)$$

then the differences $t_{30}[3]$ and $t_{31}[-3]$ cancel each other out in terms t_{32},t_{33} and t_{34}, while difference $t_{31}[-3]$ cancels the last changed word M_4, and the pair $(M, M + \Delta M)$ is collision. Second message difference corresponds to the second set of 123 (in the original paper [4] the conditions listed 122, and the work [7] shows that the two conditions are omitted; it turns out that one of the conditions expressed by the other two, so the number of sufficient conditions is 123), e.g. 146 sufficient conditions [9].

4 Algorithm for finding collisions

To search for collisions by evolutionary algorithm, or by hill climbing, it is necessary to choose a fitness function, whose optimization leads to the message M for which $(M, M + \Delta M)$ is collision. For this purpose, fitness function $c(M)$ is defined as maximal k such that the first k sufficient conditions are satisfied for the sequence in (1). The rest of this section presents an evolutionary algorithm, hill climbing algorithm and a hybrid algorithm, which is actually their composition.

4.1 Evolutionary algorithm

A population which is input to the evolutionary algorithm is made of N randomly selected messages $p[1], p[2], ..., p[N]$. Individuals are considered to be messages that need to be changed to meet the sufficient conditions for the collision. Individual fitness function is defined in terms of set of sufficient conditions that an individual meets. Individual crossover is defined as a replacement of the parts of messages, and the individual mutation means changing a few of their bits. Since the main purpose of the evolutionary algorithm is successive message modification in a way that the conditions for a collision are successively fulfilled, this paper uses its modified version where the process of selection involves comparing the existing individuals with new ones and the survival of the fittest. This approach is the result of experiments in which a standard genetic algorithm that allows the possibility that two weak individuals can produce a good one, gave worse results. This can be explained by a choice of fitness function for which all conditions must be successively met, making it unlikely that the two low fitness individuals may produce a good one. To the lack of possibility for inapt individuals to survive, leads to uniformity of the population. As the individuals become very similar, enhanced mutation is applied in each iteration. In this way, the evolutionary algorithm actually simulates message modification, which tends to incline towards the message that meets all relevant conditions. This process can be optionally continued using hill climbing algorithm. A general evolutionary algorithm can be described by the following code:

```
Algorithm EA($p, N, M$)
Input:
  $p$ - initial population, a series of $N$ messages
  $M$- number of generations in the genetic algorithm
Output:
  $p$ - the final population after generation $M$
begin
    for $i \leftarrow 1$ to $M$ do
        for $j \leftarrow 1$ to $N/4$ do
            if $p[2j-1] = p[2j]$ then mutation($p[2j]$)
            crossover $p[2j-1], p[2j]$ creates $p[N+2j-1], p[N+2j]$
            mutation ($p[N+2j-1], p[N+2j]$)
        sort $p$ by values $c(p[j]))$ and keep the first $N$ messages
        for $j \leftarrow 2$ to $N$ do
            while ($p[j]$ is equal $p[k]$ for some $k = 1, 2, ..., j-1$) do
              mutation ($p[j]$)
    return $p$
end
```

EA uses elimination selection without duplicates. In each iteration, we crossover messages $[2i-1]$ and $p[2i]$ with random crossing point resulting in their children, the messages $p[N+2i-1]$ i $p[N+2i]$, $i = 1, 2, ..., N/4$. The crossover is done in a standard way:

$$(\alpha, \beta), (\gamma, \delta) \rightarrow (\alpha, \delta), (\gamma, \beta), 0 \leq |\alpha| = |\gamma| \leq 512$$

After each iteration, individuals are sorted according to fitness function value. The individuals with lower fitness are rejected in a way the population size remains unchanged. The fittest messages do not fall out of the population. Mutation of individuals is carried out in three cases: to avoid the crossover of the same message, in order to further distinguish parents from the children and after changing population to throw out possible duplicate messages. Mutation is performed in the following way: choose a random number $0 \leq n \leq 127$ and xor message bits in positions $4n$, $4n+1$, $4n+2$, $4n+3$ with random 4-bit block. In that way at least one of the bits in the specified positions will be inverted with probability 15/16.

4.2 Correction algorithm

Let $d(x, y)$ denote the Hamming distance between vectors x and y. For arbitrary $y \in B^{512}$ let $S_t(y) = \{x | d(x, y) \leq t\}$ denote the Hamming ball of radius t with center y. If the fittest message from EA does not satisfy all sufficient conditions, we continue with correction algorithm (CA) — modification of the "Steepest-ascent hill climbing" algorithm [10]. The fittest messages obtained from the evolutionary algorithm are inserted into the priority queue Q. Correction algorithm in each iteration removes from Q the message y of highest priority, computes $c(z)$ for all $z \in S_t(y)$ and inserts into Q all z for which $c(z) \geq c(y)$. The algorithm is described by the following code:

```
**Algorithm CA**$(Q, MAX, OP)$
**Input:**
  $Q$          - array of messages
  $MAX$        - maximal possible message fitness
  $OP$         - limit for the number of tested messages achieving maximum fitness
**Output:**
  $collisions$ - array of weak messages
**begin**
    $collisions \leftarrow \emptyset$
    $counter \leftarrow 0$
    $y \leftarrow$ pullHighestPriorityElement$(Q)$
    $achievedFitness \leftarrow y.fitness$
    **while**$(y.fitness < MAX$ **and** $counter < OP)$
        **for all** $p \in S_3(y)$ **do**
            **if** $p$ is new **and** $p.fitness \geq achievedFitness$ **then**
                $achievedFitness \leftarrow p.fitness$
                $Q \leftarrow p$
        **if** $achievedFitness = y.fitness$ **then**
            increment $counter$
        **else**
            $counter \leftarrow 0$
        $y \leftarrow$ pullHighestPriorityElement$(Q)$
    $collisions \leftarrow \{p | p \in Q, p.fitness = MAX\}$
    **return** $collisions$
**end**
```

If the number of subsequent messages y with the same fitness less than MAX exceeds OP, the execution stops.

4.3 Hybrid algorithm

Hybrid algorithm is the composition of EA and CA.

```
Algorithm HA($N, M, MAX, OP, V$)
Input: $N$ - initial size of the population in EA
       $M$ - number of generations in EA
       $MAX$ - maximal possible message fitness
       $OP$ - limit for the number of tested messages which achieve maximum fitness
       $V$ - initial size of the queue for CA
Output: $collisions$ - array of weak messages
begin
  $collisions \leftarrow \emptyset$
  $p \leftarrow$ set of $N$ random messages
  $newPopulation \leftarrow$ EA($p, N, M$)
  $collisions \leftarrow$ all weak messages from $newPopulation$
  if $collisions \neq \emptyset$ then
      return $collisions$
  else
      $Q \leftarrow$ first $V$ messages from $newPopulation$
      return CA($Q, MAX, OP$)
end
```

5 Results

The population size $N = 800$ and the number of generations $M = 1000$ were chosen for the evolutionary algorithm. For the OP parameter used in correction algorithm value $OP = 7$ was selected. The results of 10 tests of the first, second and the third set of sufficient conditions (with successively 62, 123 and 144 conditions) are shown in Table 1.

In the table, the number written in italic font indicates the case when all the conditions are met, but the collision was not found. Since EA cannot always find a collision when second set of 123 conditions is met, it can be concluded that this set of sufficient conditions derived from the differential path [4] is not complete. In the case of the third set of 146 conditions, the paper [9] gives an incomplete list of 144 of them, and their fulfillment by EA does not always produce the expected collision, since the omitted conditions are randomly met. Generally, if m conditions are omitted from a set of sufficient conditions, then to obtain a collision, EA needs to create 2^m weak

Table 1: Test results

test	62 conditions		123 conditions		144 conditions	
number	EA	HA	EA	HA	EA	HA
1	56	62	*123*	0	*144*	0
2	54	62	*123*	0	138	144
3	52	62	123	0	130	144
4	55	62	116	123	144	0
5	54	62	104	123	144	0
6	51	62	115	115	143	*144*
7	54	62	105	105	140	144
8	52	62	116	116	*144*	0
9	51	56	99	99	144	0
10	59	59	115	115	133	*144*

messages on average. Specifically, for a population of 800 messages, after 1000 iterations, in the best case, the whole newly obtained population (of the last generation) satisfies all the conditions; then for $m \leq 9$ EA continues to produce collisions. The results show that EA successfully produced collision with an incomplete set of sufficient conditions, which means that EA can also be used as a criterion to check the completeness of a set of sufficient conditions. The result of the experiment is quite satisfactory.

Tests were carried out on a computer with a processor AMD Phenom II X4 945 3.01 GHz and 4 GB of working memory. Evolutionary algorithm runs about ten minutes for all sets of conditions, while CA examines one message about 20 minutes. Although CA is designed to serve as a possible continuation of the EA, it is also independent of EA and can independently produce collisions. However, on average, it takes more time and produces fewer collisions than EA. From an initial population of randomly generated 800 messages, the fittest of them on average meets the first 9 conditions (probability of satisfying these conditions is $1/2^9$), after which a message that satisfies all sufficient collisions is created by successive corrections.

As an illustration of the results obtained, the examples of collisions detected for each of the three used sets of sufficient conditions are given. Messages M and M' are different in the third bit of the fourth word in the case of 62 conditions, or in bits of the first, second and twelfth word in the case of 123 and 146 conditions (Table 2). The above collisions were obtained in experiments numbered by 6, 3, 4 in Table 1.

Experiments with a fitness function equal to the total number of satisfied conditions did not give good results, except for the first, the smallest set of sufficient conditions.

6 Comparison with the other approaches

Message modification problem, which has been solved using SAT solvers [6] and which also gives collision in less than 10 minutes, is a somewhat faster solution as it has been obtained in a computer with a low performance. However, the heuristic approach discussed in this paper produces not one but multiple collisions. In the best case, if the evolutionary algorithm succeeds in producing collisions on its own, each message in the population may produce a collision.

7 Conclusion

After obtaining sufficient conditions for MD4 collision, message modification can be very demanding. This work explains a new method for replacing standard message modification phase in MD4 algorithm using evolutionary algorithm and hill climbing. Further research involves the application on this procedure to other hash algorithms belonging to the MD4 family.

References

[1] H.B. Yu, G.L. Wang, G.Y. Zhang, X.Y. Wang: The Second-Preimage Attack on MD4. CANS 2005, LNCS 3810, pp. 1-12.

[2] X.Y. Wang, H.B. Yu: How to Break MD5 and Other Hash Functions. Eurocrypt'05, LNCS 3494, pp. 19-35.

[3] Y. Sasaki, L. Wang, K. Ohta, N. Kunihiro: New Message Difference for MD4. In A. Biryukov, ed. FSE 2007, Proceedings, volume 4593 of LNCS, pp. 329-348. Springer, 2007.

[4] X.Y. Wang, X.J. Lai, D.G. Feng, H. Chen, X.Y. Yu: Cryptanalysis for Hash Functions MD4 and RIPEMD. Eurocrypt'05, May 2005.

[5] R.L. Rivest: The MD4 Message Digest Algorithm. Advances in Cryptology, Crypto'90, Springer 1991, pp. 303-311.

[6] I. Mironov, L. Zhang: Applications of SAT Solvers to Cryptanalysis of Hash Functions. SAT 2006, pp. 102-115.

[7] Y. Naito, Y. Sasaki, N. Kunihiro, K. Ohta: Improved Collision Attack on MD4 with Probability Almost 1. In D. Won and S. Kim, eds. ICISC, volume 3935 LNCS, pp. 129-145. Springer, 2005.

[8] P.A. Fouque, G. Leurent, P. Nguyen: Automatic Search of Differential Path in MD4. ECRYPT Hash Worshop - Cryptology ePrint Archive, Report 2007/206 (2007) http://eprint.iacr.org/.

[9] M. Schläffer, E. Oswald: Searching for Differential Paths in MD4. Robshaw, FSE 2006, pp. 242-261.

[10] M. Mitchell: An introduction to genetic algorithms. MIT Press, Cambridge, Massachusetts (1999).

[11] M. Carić: Finding collisions in cryptographic hash functions, Master Thesis, Faculty of Mathematics, Belgrade, 2010 (in Serbian).

[12] M. Carić, M. Živković: Finding collisions for MD4 hash algorithm using metaheuristics algorithms, YUINFO 2013, Kopaonik, pp. 628-632 (in Serbian).

Table 2: Examples of collisions for the first, second or third set of sufficient conditions

M	53fdda09	dbd460d2	6b0d1c7e	d41233e2
	0e973a6**3**	ee35f949	3d28ca69	6d101738
	1f760241	173b3175	07531617	4c867a98
	797525e2	dd0b1b98	cec0df99	4f45d906
M'	53fdda09	dbd460d2	6b0d1c7e	d41233e2
	0e973a6**7**	ee35f949	3d28ca69	6d101738
	1f760241	173b3175	07531617	4c867a98
	797525e2	dd0b1b98	cec0df99	4f45d906
Hash	cd9369f2	5c77a88c	349b7fd3	b820249c
M	e85075c4	**6**e37e13b	**7**29fb981	ffefea77
	8ae24c2b	cec4f21e	62a4c566	b681a50a
	89e04800	e9bfdf35	5d38b88a	9a847d14
	9af**d**08ad	bfdc2bf2	e6b416e2	d7edc8b5
M'	e85075c4	**e**e37e13b	**e**29fb981	ffefea77
	8ae24c2b	cec4f21e	62a4c566	b681a50a
	89e04800	e9bfdf35	5d38b88a	9a847d14
	9af**c**08ad	bfdc2bf2	e6b416e2	d7edc8b5
Hash	85a9276a	331f0b93	7661e284	92531baa
M	7d4182c2	**3**64f032b	**f**369358a	32568a9e
	1417ea70	357ba164	770ab75f	bb95772e
	f78c5797	b651d1fa	234a288d	3660f5c0
	004**0**cb07	a9ded82a	458d2286	29378008
M'	7d4182c2	**b**64f032b	**6**369358a	32568a9e
	1417ea70	357ba164	770ab75f	bb95772e
	f78c5797	b651d1fa	234a288d	3660f5c0
	004**f**cb07	a9ded82a	458d2286	29378008
Hash	eb8fa12e	cf298b09	bcec958d	4cefae4b

Table 3: Differential path that corresponds to the message difference $\Delta m_4 = 2^2, \Delta m_i = 0, i \neq 4$.

i	t_i	$M_{p(i)}$	s_i	ΔM_i	$t'_i - t_i$	t'_i	Sufficient conditions
1	t_1	M_0	3			t_1	
2	t_2	M_1	7			t_2	
3	t_3	M_2	11			t_3	
4	t_4	M_3	19			t_4	
5	t_5	M_4	3	2^2	2^5	$t_5[6]$	$t_{5,6} = 0$
6	t_6	M_5	7			t_6	$t_{4,6} = t_{3,6}$
7	t_7	M_6	11			t_7	$t_{6,6} = 0$
8	t_8	M_7	19			t_8	$t_{7,6} = 1$
9	t_9	M_8	3		2^8	$t_9[-9, 10]$	$t_{9,9} = 1$, $t_{9,10} = 0$
10	t_{10}	M_9	7			t_{10}	$t_{8,9} = t_{7,9}$, $t_{8,10} = t_{7,10}$
11	t_{11}	M_{10}	11		-2^{19}	$t_{11}[-20]$	$t_{11,20} = 1$, $t_{10,9} = 1$, $t_{10,10} = 0$
12	t_{12}	M_{11}	19			t_{12}	$t_{11,9} = 1$, $t_{11,10} = 1$, $t_{10,20} = t_{9,20}$
13	t_{13}	M_{12}	3		2^{11}	$t_{13}[12]$	$t_{13,12} = 0$, $t_{12,20} = 0$
14	t_{14}	M_{13}	7			t_{14}	$t_{12,12} = t_{11,12}$, $t_{13,20} = 1$
15	t_{15}	M_{14}	11		-2^{30}	$t_{15}[-31]$	$t_{15,31} = 1$, $t_{14,12} = 0$
16	t_{16}	M_{15}	19			t_{16}	$t_{14,31} = t_{13,31}$, $t_{15,12} = 1$
17	t_{17}	M_0	3		2^{14}	$t_{17}[15]$	$t_{17,15} = 0$, $t_{16,31} = t_{14,31}$

Table 3: (Continued.)

i	t_i	$M_{p(i)}$	s_i	ΔM_i	$t'_i - t_i$	t'_i	Sufficient conditions
18	t_{18}	M_4	5	2^2	$2^7 + 2^{19}$	$t_{18}[8, 20]$	$t_{18,8} = 0$, $t_{16,15} = t_{15,15} + 1$, $t_{18,20} = 0$, $t_{17,31} = t_{16,31}$
19	t_{19}	M_8	9		-2^7	$t_{19}[-8]$	$t_{19,8} = 1$, $t_{17,8} = t_{16,8}$, $t_{18,15} = t_{16,15}$, $t_{17,20} = t_{16,20}$
20	t_{20}	M_{12}	13			t_{20}	$t_{19,15} = t_{18,15}$, $t_{19,20} = t_{17,20}$
21	t_{21}	M_1	3		2^{17}	$t_{21}[18]$	$t_{21,18} = 0$, $t_{20,20} = t_{19,20}$
22	t_{22}	M_5	5		2^{24}	$t_{22}[25]$	$t_{22,25} = 0$, $t_{20,18} = t_{19,18}$, $t_{21,8} = t_{20,8} + 1$
23	t_{23}	M_9	9		-2^{16}	$t_{23}[17, -18]$	$t_{23,17} = 0$, $t_{23,18} = 1$, $t_{22,18} = t_{20,18}$, $t_{21,25} = t_{20,25}$
24	t_{24}	M_{13}	13			t_{24}	$t_{22,17} = t_{21,17}$, $t_{23,25} = t_{21,25}$
25	t_{25}	M_2	3			t_{25}	$t_{24,17} = t_{22,17}$, $t_{24,25} = t_{23,25}$, $t_{24,18} = t_{22,18} + 1$
26	t_{26}	M_6	5		2^{29}	$t_{26}[30]$	$t_{25,17} = t_{24,17}$, $t_{25,18} = t_{24,18}$, $t_{26,30} = 0$
27	t_{27}	M_{10}	9		-2^{25}	$t_{27}[-26]$	$t_{27,26} = 1$, $t_{25,30} = t_{24,30}$
28	t_{28}	M_{14}	13			t_{28}	$t_{26,26} = t_{25,26}$, $t_{27,30} = t_{25,30}$
29	t_{29}	M_3	3			t_{29}	$t_{28,26} = t_{26,26}$, $t_{28,30} = t_{27,30}$
30	t_{30}	M_7	5		2^2	$t_{30}[3]$	$t_{30,3} = 0$, $t_{29,26} = t_{28,26}$
31	t_{31}	M_{11}	9		-2^2	$t_{31}[-3]$	$t_{31,3} = 1$, $t_{29,3} = t_{28,3}$
32	t_{32}	M_{15}	13			t_{32}	
33	t_{33}	M_0	3			t_{33}	
34	t_{34}	M_8	9			t_{34}	$t_{33,3} = t_{32,3}$
35	t_{35}	M_4	11	2^2		t_{35}	