An Attack on the Last Two Rounds of MD4

Bert den Boer
Philips Crypto B.V.
P.O. Box 218
5600 MD Eindhoven
The Netherlands

Antoon Bosselaers
ESAT Laboratory, K.U. Leuven
Kard. Mercierlaan 94
B-3001 Heverlee
Belgium
bosselaers@esat.kuleuven.ac.be

15 October 1991

Abstract

In [Rive90] the MD4 message digest algorithm was introduced taking an input message of arbitrary length and producing an output 128-bit message digest. It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given prespecified target message. In this paper it is shown that if the three round MD4 algorithm is stripped of its first round, it is possible to find for a given (initial) input value two different messages hashing to the same output. A computer program implementing this attack takes about 1 millisecond on a 16 Mhz IBM PS/2 to find such a collision.

1 Introduction

The MD4 Message Digest Algorithm, by Ronald L. Rivest and RSA Data Security, Inc., is intended for file hashing: it accepts arbitrarily large inputs and produces an output of 128 bits. It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given prespecified target message digest. The MD4 algorithm is designed to be quite fast on 32-bit machines.

An interesting topic of investigation is how these claims relate to the number of rounds of MD4, and whether they hold for a weaker version of the algorithm, stripped of its first or last round. The latter was considered by Ralph Merkle [Merk90], who showed that skipping the last round jeopardizes the strength of the system: for 99.99% of all initial values it is possible to find two message differing in only 3 bits, which are hashed to the same output value. We derived an algorithm for it and wrote a computer program implementing the attack. The program takes less than a millisecond on a 16 Mhz IBM PS/2 to find such a collision. Next, we investigated whether by skipping the first round one would also be able to produce a collision of messages. It will be shown that it is indeed possible for MD4 stripped of its first round to find for a given (initial) input value two different messages hashing to the same output.

In Section 2 a short description of MD4 is given. In Section 3 the attack on the last

two rounds is explained, and an example collision for MD4 skipping the first round is given. The possibility to extend this attack to the full three round algorithm is briefly discussed.

2 Short description of MD4

Below only a general description of the MD4 message digest algorithm is given. For a detailed description the reader is referred to [Rive90]. In this description a *byte* is defined as an 8-bit quantity and a *word* as a 32-bit quantity. A sequence of bits can be interpreted as a sequence of bytes, where each consecutive group of 8 bits is interpreted as a byte with the *high*-order bit of each byte listed first. Similarly, a sequence of bytes can be interpreted as a sequence of words, where each consecutive group of 4 bytes is interpreted as a word with the *low*-order byte given first. The input and output of MD4 is considered to be a sequence of bytes, not words, but the internal operations of the algorithm are word oriented.

The MD4 recipe for a b-bit message consists of the following steps:

1. **Append padding bits and the message length**: a single "1" bit, $l-1$ zero bits ($1 \leq l < 512$) and the 64-bit representation of $b \bmod 2^{64}$ are appended to the message such that the length $b+l+64$ of the extended message is a multiple of 512. The 64 bits containing the message length are appended as two 32-bit words, *low*-order word first in accordance with the previous conventions. The (new) message is now represented as a sequence $M[0], M[1], \ldots, M[n-1]$ of n words, where n is a multiple of 16 (because $32 \cdot n = b + l + 64 \equiv 0 \bmod 512$).

2. **Initialize a 4-word buffer** (A, B, C, D)**:**

   $A = \texttt{67452301}; B = \texttt{EFCDAB89}; C = \texttt{98BADCFE}; D = \texttt{10325476}.$
   (32-bit constants in hexadecimal notation, high-order digits first)

3. **Process the message in 16-word blocks:**

```
for i = 0 to (n/16) − 1 do
   begin
      for j = 0 to 15 do
         X[j] = M[16i + j];
      (AA, BB, CC, DD) = (A, B, C, D);
      Round 1;
      Round 2;
      Round 3;
```

$A = A + AA;$
$B = B + BB;$
$C = C + CC;$
$D = D + DD;$
`end;`

4. **The output is the 4-word buffer** (A, B, C, D)**.**

Figure 2 shows the outline of step 3 of MD4, without the feedforward.

3 Description of the attack on the last two rounds

Each of the three rounds of MD4 consists of 16 elementary operations on the 4-word buffer (A, B, C, D). Let (A_i, B_i, C_i, D_i) denote the value of (A, B, C, D) after i elementary operations. In this section only the last two rounds of MD4 are considered. The elementary operations are therefore numbered relative to the beginning of the second round. It will be shown that it is possible to find for a given input two different 16-word message blocks hashing to the same output. The underlying observation is that the 8 message words $X[1]$, $X[5]$, $X[9]$, $X[13]$, $X[2]$, $X[6]$, $X[10]$ and $X[14]$ used in the elementary operations 5 till 12 are the same as those used in the elementary operations 21 till 28. In other words the *same* 8 message words are used in the middle 8 elementary operations of the second and third round. A similar observation applies for the 8 message words $X[0]$, $X[4]$, $X[8]$, $X[12]$, $X[3]$, $X[7]$, $X[11]$ and $X[15]$ used in the first 4 (elementary operations 1 till 4 and 17 till 20) and last 4 (elementary operations 13 till 16 and 29 till 32) elementary operations of the second and third round. The latter 8 message words $X[0]$, $X[4]$, $X[8]$, $X[12]$, $X[3]$, $X[7]$, $X[11]$ and $X[15]$ are given the same value in the two 16-word message blocks. Consequently (A_4, B_4, C_4, D_4) has the same value for both messages. If $(A_{12}, B_{12}, C_{12}, D_{12})$ is equal for both messages, then $(A_{20}, B_{20}, C_{20}, D_{20})$ will be equal too, and if $(A_{28}, B_{28}, C_{28}, D_{28})$ has the same value for both messages, both messages are hashed to the same output value. The two message blocks only differ in the remaining 8 words $X[1]$, $X[5]$, $X[9]$, $X[13]$, $X[2]$, $X[6]$, $X[10]$ and $X[14]$ used in the middle 8 elementary operations of the last two rounds. The two alternatives for these message words are precisely chosen in such a way that the 4-word buffer (A, B, C, D) has two alternatives after 8 and 24 elementary operations (this is halfway the second and third round), but the same value for both messages after 12 and 28 elementary operations. Hence we have two different messages and a single input value, which are hashed to the same output value. This situation is illustrated in Figure 1, where every dot represents a different value of the buffer (A, B, C, D).

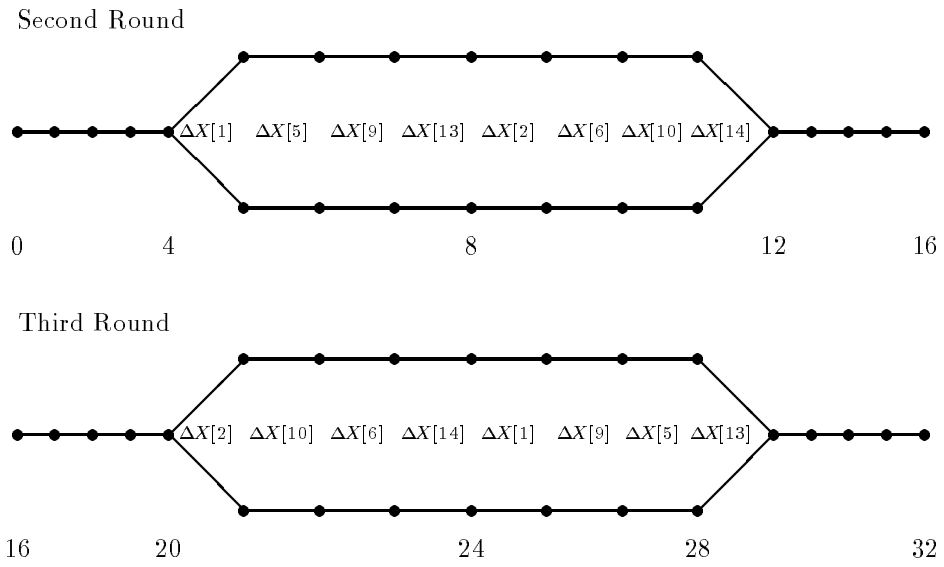


Figure 1: Outline of the attack on the last two rounds

The problem we are confronted with is the following. We have to solve 32 equations (twice the middle 8 elementary operations of each round, one for each alternative of the input) with unknowns:

- the value of (A_4, B_4, C_4, D_4)
- two alternatives of (A_8, B_8, C_8, D_8)
- the value of $(A_{12}, B_{12}, C_{12}, D_{12})$
- the value of $(A_{20}, B_{20}, C_{20}, D_{20})$
- two alternatives of $(A_{24}, B_{24}, C_{24}, D_{24})$
- the value of $(A_{28}, B_{28}, C_{28}, D_{28})$
- two alternatives for the message words $X[2]$, $X[6]$, $X[10]$, $X[14]$, $X[1]$, $X[5]$, $X[9]$ and $X[13]$

Altogether there are 48 unknown words. Once we found a solution for this problem, we are left with 16 equations (the first and last 4 elementary operations of each round) with 16 unknowns:

- the final value of (A, B, C, D)
- the value of $(A_{16}, B_{16}, C_{16}, D_{16})$
- the message words $X[0]$, $X[4]$, $X[8]$, $X[12]$, $X[3]$, $X[7]$, $X[11]$ and $X[15]$

An initial value of the word (A, B, C, D) can be met with precisely one choice of $X[0]$, $X[4]$, $X[8]$ and $X[12]$, which on their turn determine the value of $(A_{16}, B_{16}, C_{16}, D_{16})$. Precisely one choice of $X[3]$, $X[7]$, $X[11]$ and $X[15]$ couples the value of $(A_{12}, B_{12}, C_{12}, D_{12})$ with the value of $(A_{16}, B_{16}, C_{16}, D_{16})$. Finally this same choice determines the final value of (A, B, C, D).

In what follows we will explain how to find one set of the above mentioned 48 unknown words. In this solution a crucial role is played by the hexadecimal word 55555555, which will be called N. The bits on the odd positions of N are equal to zero, and on the even positions equal to one (this defines what is meant by 'odd' and 'even' positions). The number N has the following interesting property: any rotation in any direction over an odd number of bits yields $2N$ (or $\overline{N}$). It is precisely this property of N which is used to find a solution for the problem we are left with. We will first introduce some notation:

$\overline{A}$	denotes the bit wise complement of A.
$A\&B$	denotes the bit wise AND of the words A and B.
$A \ll s$	denotes the shifting of A to the left by s bit positions.
$A \lll s$	denotes the circularly shifting (rotating) of A to the left by s bit positions.
$A \ggg s$	same as above, but to the right.
A_0^N	denotes a word with following two alternatives: the bits of A_0^N on the even positions are all equal to one for the first alternative (denoted by A^N), and all equal to zero for the second alternative (denoted by A_0), i.e. $A_0^N\&N$ equals either N or 0. The bits on the odd positions of A_0^N can be chosen freely, but are equal for both alternatives.
A^N, A_0	are the two alternatives of the word A_0^N.
A_N^0	denotes similarly a word with following two alternatives: the bits of A_N^0 on the even positions are all equal to zero for the first alternative (denoted by A^0), and all equal to one for the second alternative (denoted by A_N), i.e. $A_N^0\&N$ equals either 0 or N. The bits on the odd positions of A_N^0 can be chosen freely, but are equal for both alternatives.
A^0, A_N	are the two alternatives of the word A_N^0.

The solution consists of the following 5 steps:

step 1 Choose the value of (A, B, C, D) equal to

- $(A_N^0, B_N^0, C_N^0, D_N^0)$ after 8 elementary operations,
- $(A_0^N, B_0^N, C_0^N, D_0^N)$ after 24 elementary operations.

The 128 ($=2 \cdot 4 \cdot 16$) bits on the odd positions of one alternative of these words can be chosen freely. The bits on the odd positions of the other alternative are of course the same. Note that the even positions after 8 and 24 elementary operations are the complement of each other. This results in 32 equations: the middle 8 elementary equations of the last two rounds, each with 2 alternatives.

$$
\begin{array}{rcllllllll}
A_N^0 & = & (A_4 & + & g(B_4, C_4, D_4) & + & X^i[1] & + & E2) \lll gs1 & (1) \\
D_N^0 & = & (D_4 & + & g(A_N^0, B_4, C_4) & + & X^i[5] & + & E2) \lll gs2 & (2) \\
C_N^0 & = & (C_4 & + & g(D_N^0, A_N^0, B_4) & + & X^i[9] & + & E2) \lll gs3 & (3) \\
B_N^0 & = & (B_4 & + & g(C_N^0, D_N^0, A_N^0) & + & X^i[13] & + & E2) \lll gs4 & (4) \\
 & & & & & & & & & \\
A_{12} & = & (A_N^0 & + & g(B_N^0, C_N^0, D_N^0) & + & X^i[2] & + & E2) \lll gs1 & (5) \\
D_{12} & = & (D_N^0 & + & g(A_{12}, B_N^0, C_N^0) & + & X^i[6] & + & E2) \lll gs2 & (6) \\
C_{12} & = & (C_N^0 & + & g(D_{12}, A_{12}, B_N^0) & + & X^i[10] & + & E2) \lll gs3 & (7) \\
B_{12} & = & (B_N^0 & + & g(C_{12}, D_{12}, A_{12}) & + & X^i[14] & + & E2) \lll gs4 & (8) \\
 & & & & & & & & & \\
A_0^N & = & (A_{20} & + & h(B_{20}, C_{20}, D_{20}) & + & X^i[2] & + & E3) \lll hs1 & (9) \\
D_0^N & = & (D_{20} & + & h(A_0^N, B_{20}, C_{20}) & + & X^i[10] & + & E3) \lll hs2 & (10) \\
C_0^N & = & (C_{20} & + & h(D_0^N, A_0^N, B_{20}) & + & X^i[6] & + & E3) \lll hs3 & (11) \\
B_0^N & = & (B_{20} & + & h(C_0^N, D_0^N, A_0^N) & + & X^i[14] & + & E3) \lll hs4 & (12) \\
 & & & & & & & & & \\
A_{28} & = & (A_0^N & + & h(B_0^N, C_0^N, D_0^N) & + & X^i[1] & + & E3) \lll hs1 & (13) \\
D_{28} & = & (D_0^N & + & h(A_{28}, B_0^N, C_0^N) & + & X^i[9] & + & E3) \lll hs2 & (14) \\
C_{28} & = & (C_0^N & + & h(D_{28}, A_{28}, B_0^N) & + & X^i[5] & + & E3) \lll hs3 & (15) \\
B_{28} & = & (B_0^N & + & h(C_{28}, D_{28}, A_{28}) & + & X^i[13] & + & E3) \lll hs4 & (16)
\end{array}
$$

In each block of four equations there is now one equation with only 2 unknowns: (4), (5), (12) and (13). Going up or down within each block (up in the first and third, down in the other two) one encounters equations with more and more unknowns. The route to follow is quite obvious now: solve the equations with only 2 unknowns by making a free choice for one of them, and work your way up (or down) the other equations, solving them for the remaining unknowns.

step 2 Choose $X^1[1]$, $X^1[2]$, $X^1[5]$, $X^1[10]$, $X^1[13]$ and $X^1[14]$. This gives, in addition to the 128 bits of step 1, another 192 bits of freedom. Equations (4), (5), (12) and (13) yield B_4, A_{12}, B_{20} and A_{28} respectively, as well as the alternative message words $X^2[13]$, $X^2[2]$, $X^2[14]$ and $X^2[1]$. The choices for $X^1[5]$ and

$X^1[10]$ are not used until step 5. The two alternatives of $X[6]$ and $X[9]$ are fixed by the choices we already made, as will become clear in the following three steps.

step 3 If a majority (g) or an exor (h) function has two fixed inputs and one input with an alternative then the bits on the even positions of the fixed inputs are taken equal. This means that the bits on the even positions of the pairs (B_4, C_4), (A_{12}, D_{12}), (B_{20}, C_{20}) and (A_{28}, D_{28}) are per pair taken equal.

The two alternatives for the equations (7) and (10) contain the unknowns C_{12}, D_{20} and two alternatives for $X[10]$, in addition to the unknowns D_{12} and C_{20}. By reversely rotating these four equations by the right amount and subtracting them from each other, the first four unknowns drop out leaving one equation in the unknowns D_{12} and C_{20}. We do the same with the first alternative of the equations (6) and (11). The unknown alternative $X^1[6]$ drops out, resulting in another equation in the unknowns D_{12} and C_{20}. If we denote the two alternatives of equation (i) by (ia) and (ib), we obtain the following set of 2 equations:

$$((7\text{a}) \ggg gs3) - ((7\text{b}) \ggg gs3) - (((10\text{a}) \ggg hs2) - ((10\text{b}) \ggg hs2))$$

$$((6\text{a}) \ggg gs2) - ((11\text{a}) \ggg hs3)$$

or

$$h(A^N, B_{20}, C_{20}) - h(A_0, B_{20}, C_{20}) + g(D_{12}, A_{12}, B^0) - g(D_{12}, A_{12}, B_N) = N$$

$$(D_{12} \ggg gs2) + C_{20} = (C^N \ggg hs3) + D^0 + g(A_{12}, B^0, C^0) - h(D^N, A^N, B_{20}) + E2 - E3$$

where only the lefthand sides contain the unknowns D_{12} and C_{20}. The choice mentioned in the beginning of this step gives a solution for this set of 2 equations. For, the first equation holds if the bits on the even positions of C_{20} are taken equal to those of B_{20}, and the even bits of D_{12} equal to those of A_{12}. The remaining 16 bits of D_{12} and C_{20} can then easily be solved from the second equation, as explained in the next step. An analogous set of 2 equations in the unknowns C_4 and D_{28} can be derived in the same way from equations (2) and (15), and (3) and (14). This set can be easily solved under the same conditions.

step 4 Since $gs2$ is odd, the left term of the second equation in the previous step can be written as the sum of two variables DC and CD:

$$(D_{12} \ggg gs2) + C_{20} = DC + CD$$

where

$$\begin{aligned} DC &= (D_{12} \ggg gs2)\,\&\,\overline{N} + C_{20}\,\&\,N \\ CD &= (D_{12} \ggg gs2)\,\&\,N + C_{20}\,\&\,\overline{N} \end{aligned}$$

DC contains on the odd bit positions the known bits of D_{12} and on the even bit positions the known bits of C_{20}, and CD contains, respectively on the even and odd bit positions, the yet unknown bits of D_{12} and C_{20}, which can now easily be found. An analogous method, based on the oddness of $hs2$, yields the remaining unknown bits of C_4 and D_{28}.

step 5 This step is no more than an exercise with blanks.

- (6) or (11) yields two alternatives of $X[6]$
- (7) and (8) yield C_{12} and B_{12}, respectively
- (10) and (9) yield D_{20} and A_{20}, respectively
- (3) or (14) yields two alternatives of $X[9]$
- (15) and (16) yield C_{28} and B_{28}, respectively
- (2) and (1) yield D_4 and A_4, respectively

The only condition imposed by the attack on the constants of equations (1) to (16) is that all the values of the rotation constants must be odd.

One can (easily) proof that the difference between two messages produced in this way is constant and equal to

$$0 \quad -2N \quad 2N \quad 0 \quad 0 \quad -2N \quad 2N \quad 0 \quad 0 \quad -N \quad N \quad 0 \quad 0 \quad -N \quad N \quad 0$$

where every number stands for a 32-bit word. A program has been written implementing this attack. With this program it takes about 1 millisecond on a 16 Mhz IBM PS/2 to find such a collision. As an example, the following pair of 512-bit messages (written as 16 32-bit words, in hexadecimal notation) produces a collision for MD4, as described in Section 2 and using only the last two rounds in the third step of the recipe:

Message 1:
`72A3B049 213AE143 D954E8C9 50BD4CB5 25A3A0B3 C79B12BE 029B6AE9 091A6156`
`75B5516B DA420FD6 0A6854EB 758F514D 9EA01345 0F796EAC DB54B645 4089373B`

Message 2:
`72A3B049 CBE58BED 2EAA3E1F 50BD4CB5 25A3A0B3 7245BD68 57F0C03F 091A6156`
`75B5516B 2F97652B B512FF96 758F514D 9EA01345 64CEC401 85FF60F0 4089373B`

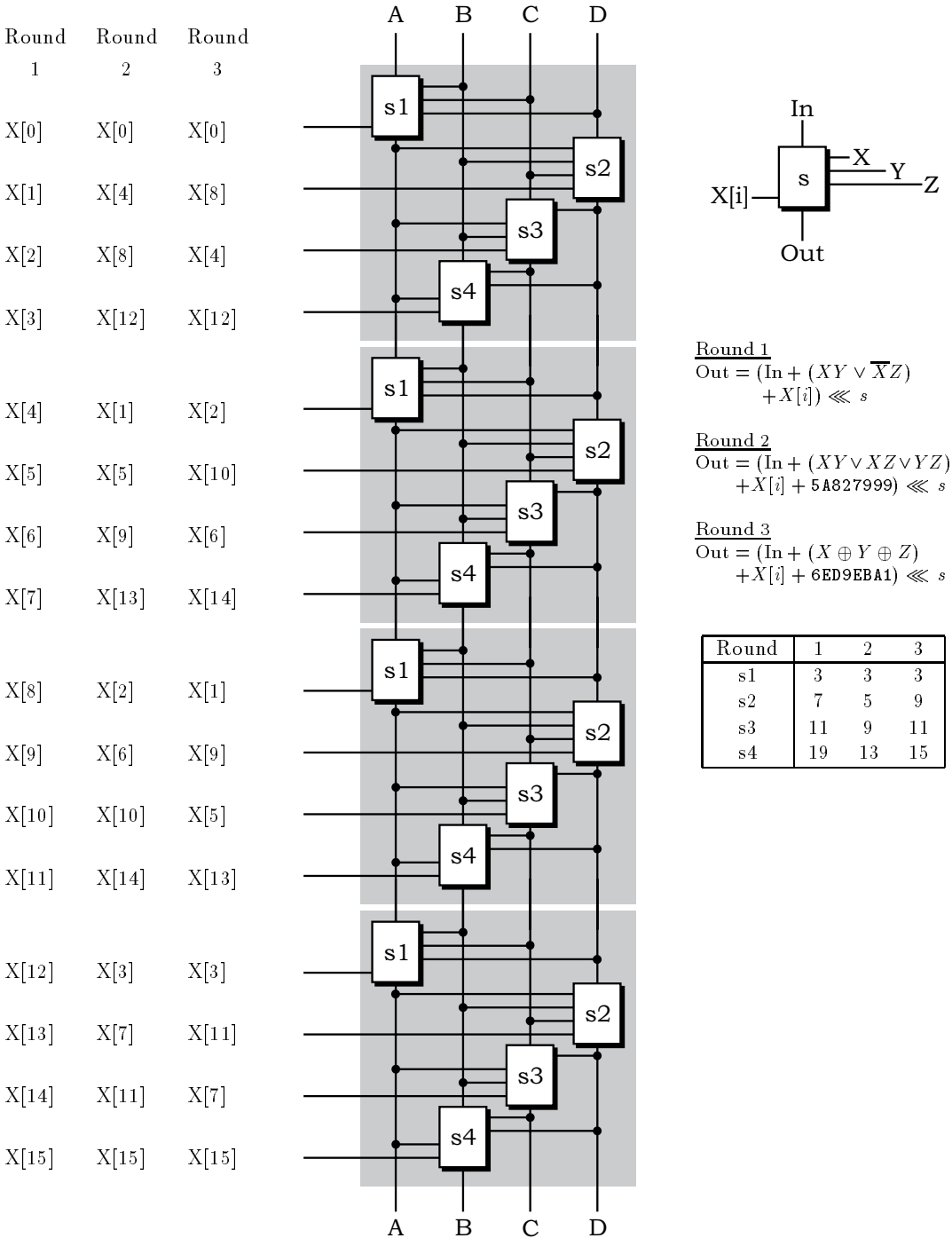
Common Message Digest:
`E259F484 488AE67F B01E240C 497301A3`

To produce such a pair of messages one has altogether 320 bits of freedom. It is conceivable that this freedom can be used to produce identical intermediate values and different 512-bit message blocks, producing a collision for the entire algorithm. In addition one could also use the extra freedom of $X[0]$, $X[4]$, $X[8]$ and $X[12]$ (or of $X[3]$, $X[7]$, $X[11]$ and $X[15]$). However, this does not result in an attack on the complete three round algorithm, since in that case the initial value is not dealt with.

References

[Rive90] R. L. Rivest, "The MD4 Message Digest Algorithm", Abstracts Crypto '90, pp. 281-291.

[Merk90] R. Merkle, personal communication.



Round	1	2	3
s1	3	3	3
s2	7	5	9
s3	11	9	11
s4	19	13	15

Figure 2: Outline of MD4 (without feedforward)