Multi-block Collisions in Hash functions based on 3C and 3C+ Enhancements of the Merkle-Damgård Construction*

Daniel Joščák and Jiří Tůma
joscd1am@karlin.mff.cuni.cz and tuma@karlin.mff.cuni.cz

Charles University in Prague,
Faculty of Mathematics and Physics, Department of Algebra,
Sokolovská 83, 186 75 Prague 8, Czech Republic.

Abstract. At the ACISP 2006 conference Praveen Gauravaram et al [2] proposed 3C and 3C+ constructions as enhancements of the Merkle-Damgård construction of cryptographic hash functions. They conjectured these constructions improved multi-block collision resistance of the hash functions. In this paper we show that the recently found collision attack on MD5 can be easily extended to the 3C and 3C+ constructions based on the MD5 compression function. In fact we show that if an algorithm satisfying some mild assumptions can find multi-block collisions for the Merkle-Damgård construction then it can be easily modified to find multi-block collisions for the 3C and 3C+ constructions based on the same compression function.
Keywords: hash functions, multi-block collision attack, 3C and 3C+ constructions.

1 Introduction

Research in the design and analysis of cryptographic hash functions has been very active since Wang at al [7] published their first collision search algorithm for the MD5 hash function. Collision search algorithms for other hash functions have been discovered, in particular for SHA-0, see [1], [8]. An algorithm for finding collisions in SHA-1 that is significantly more efficient than the generic birthday attack was announced in [9].

In the light of these attacks Gauravaram et al [2] have proposed a slight modification to the Merkle-Damgård construction for an improved protection against many known attacks on MD based hash functions. Their idea is to add additional registers that would collect xors of all chaining variables. After the message is processed the content of additional registers is padded to provide one more message block and the

* Research was supported by the Institutional grant MSM0021620839

extra block is used as an input for the last calculation of the compression function. Thus the original MD construction remains and the extra security is supposed to be provided by the additional registers, see Figure 1.

Since the 3C construction contains the original MD contruction, any n-block internal collision for the 3C construction must be in fact an n-block collision of the MD construction based on the same compression function. However, because of the extra use of the compression function at the end of the 3C construction one cannot claim that an n-block collision for the 3C construction must be also an n-block collision of the MD construction. To find an n-block collision (where $n \geq 2$) for the 3C construction that is not an n-block collision for the MD construction based on the same compression function would require to find a collision in the compression function with different IV's and possibly different input blocks.

In this paper we show that the 3C construction does not increase significantly resistance against multi-block collisions. In fact, under very mild assumptions we prove that if there is an algorithm that finds n-block collisions for the MD construction based on a compression function, then one can easily find $(2n)$-block collisions for the 3C construction based on the same compression function and $(2n + 1)$-block collisions for its modification called 3C+. Our theorem can be applied in particular to the MD5 compression function.

We also observe that the 2-block collisions for the SHA-0 hash function published in [8] are in fact also 2-block collisions for the 3C construction based on the same collision function.

The paper is organized as follows. In section 2 we discuss the 3C and 3C+ design principles, in section 3 we point out a few important properties of the recent 2-block collision attacks on MD5. In Section 4 we prove two simple general theorems how multi-collision attacks on the MD construction can be extended to multi-collision attacks on the 3C and 3C+ constructions. We conclude the paper in section 5. In the appendix we present concrete examples of colliding messages for the 3C and 3C+ constructions based on the compression function of MD5.

2 Description of 3C and 3C+

The 3C construction is a generic construction designed as an enhancement to the Merkle-Damgård construction with the idea to increase its resistance against multi-block collision attacks. One of the main proper-

ties of the 3C construction is that it is as efficient as the standard hash functions when it is instantiated with the compression functions of any of these hash functions.

The 3C construction accumulates every chaining state of the MD construction by xoring it to the register already containing xor of all previous chaining states.

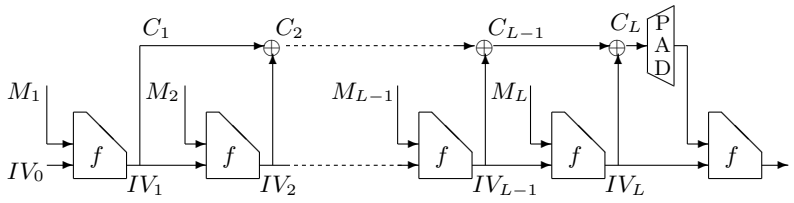

Fig. 1. 3C construction of hash function

Thus if IV_i is the chaining variable obtained as the result of i-th iteration of the compression function (IV_0 is the initialization vector), then the value of the additional accumulation registers (denoted by C_i) after the i-th iteration of the compression function is $C_1 = \mathrm{IV}_1$ and

$$C_i = C_{i-1} \oplus \mathrm{IV}_i = \mathrm{IV}_1 \oplus \mathrm{IV}_2 \oplus \cdots \oplus \mathrm{IV}_i$$

for $i = 2, \ldots, L$, where L is the number of blocks of the message. The authors also suggest in their paper [2] that different variants can be obtained for 3C by replacing the xor function in the accumulation chain by other non-linear functions.

The 3C+ construction is a different modification of the 3C construction in which yet another chain D_i, $i = 1, \ldots, L$ of additional registers accumulating the values of chaining variables is added. This time $D_1 = \mathrm{IV}_0$ and

$$D_i = D_{i-1} \oplus \mathrm{IV}_i = \mathrm{IV}_0 \oplus \mathrm{IV}_2 \oplus \cdots \oplus \mathrm{IV}_i$$

for $i = 2, \ldots, L$. Thus

$$D_i = C_i \oplus \mathrm{IV}_1 \oplus \mathrm{IV}_0$$

for every $i = 2, \ldots, L$.

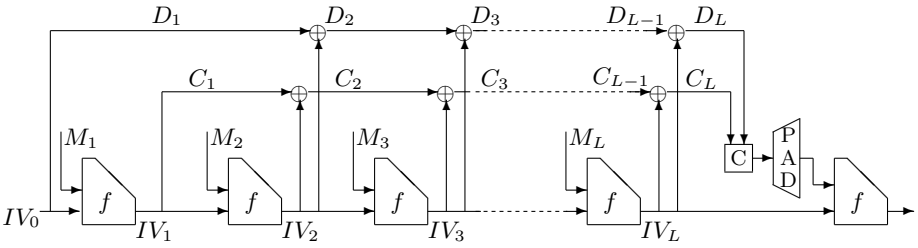

Fig. 2. 3C+ construction of hash function

3 Recent multi-block collision attacks

The hash function MD5 uses four 32-bit registers to keep the value of each chaining variable IV_i. We denote them by $\mathrm{IV}_{i,0}, \mathrm{IV}_{i,1}, \mathrm{IV}_{i,2}, \mathrm{IV}_{i,3}$. Thus

$$\mathrm{IV}_i = (\mathrm{IV}_{i,0}||\mathrm{IV}_{i,1}||\mathrm{IV}_{i,2}||\mathrm{IV}_{i,3}).$$

Wang et al presented in [7] an algorithm for finding 2-block collisions in the hash function MD5. Their algorithm works for an arbitrary initialization vector IV_0. If $(M_1||M_2)$ and $(M_1'||M_2')$ are two colliding messages found by their algorithm then the modular differences of the chaining variables after processing the first blocks M_1 and M_1' are

$$\begin{aligned}
\Delta_{i,0} &= \mathrm{IV}'_{1,0} - \mathrm{IV}_{1,0} = 2^{31} \\
\Delta_{i,1} &= \mathrm{IV}'_{1,1} - \mathrm{IV}_{1,1} = 2^{31} + 2^{25} \\
\Delta_{i,2} &= \mathrm{IV}'_{1,2} - \mathrm{IV}_{1,2} = 2^{31} + 2^{25} \\
\Delta_{i,3} &= \mathrm{IV}'_{1,3} - \mathrm{IV}_{1,3} = 2^{31} + 2^{25},
\end{aligned} \tag{1}$$

where

$$\begin{aligned}
\mathrm{IV}_1 &= f(\mathrm{IV}_0, M_1) \\
\mathrm{IV}'_1 &= f(\mathrm{IV}_0, M'_1)
\end{aligned}$$

and f is the compression function used in MD5.

Wang et al in [7] also presented a set of so-called sufficient conditions for registers in computation of $f(\mathrm{IV}, M_1)$ to produce the first blocks of a pair of colliding messages. These conditions in fact were not sufficient and various authors, e.g. [5] [6] offered their sets of sufficient conditions. For our purposes only the conditions for IV_1 are important and these

conditions were the same for all authors. In fact, we need only four of the sufficient conditions for IV_1 and these four conditions are described in the Table 1.

$IV_{1,0}$	....
$IV_{1,1}$	.....0. ....
$IV_{1,2}$	.....01. ....
$IV_{1,3}$	.....0. ....

Table 1. Prescribed conditions for IV_1

The exact value of $\mathrm{IV}_1 \oplus \mathrm{IV}'_1$ then follows from given modular differences (1) and prescribed conditions for IV_1 in the Table 1. Thus $\mathrm{IV}_1 \oplus \mathrm{IV}'_1$ is a constant independent of the initialization vector IV_0 and the first blocks M_1 and M'_1 of the colliding messages $(M_1||M_2)$ and $(M'_1||M'_2)$.

$\delta_{1,0} = \mathrm{IV}_{1,0} \oplus \mathrm{IV}'_{1,0}$	10000000 00000000 00000000 00000000
$\delta_{1,1} = \mathrm{IV}_{1,1} \oplus \mathrm{IV}'_{1,1}$	10000010 00000000 00000000 00000000
$\delta_{1,2} = \mathrm{IV}_{1,2} \oplus \mathrm{IV}'_{1,2}$	10000110 00000000 00000000 00000000
$\delta_{1,3} = \mathrm{IV}_{1,3} \oplus \mathrm{IV}'_{1,3}$	10000010 00000000 00000000 00000000

Table 2. Prescribed δ for IV_1

The collision finding algorithm for SHA-0 by Wang et al [8] also finds 2-block colliding messages but the structure of the messages in this attack is different than in the case of MD5. The first blocks of the colliding messages $(M_1||M_2)$ and $(M_1||M'_2)$ are the same and serve to obtain the chaining variable IV_1 satisfying the conditions sufficient for finding the second blocks M_2 and M'_2. The algorithm again works for an arbitrary IV_0.

In another paper [9] Wang et al propose an algorithm for finding 2-block collisions in SHA-1 that is faster than the generic birthday attack. Although no real collisions in SHA-1 have been found so far, the form of colliding messages of the proposed attack is in fact the same as in the case of MD5. It means that the algorithm should work for any IV_0 and $\mathrm{IV}_1 \oplus \mathrm{IV}'_1$ should be a constant independent of IV and M_1 and M'_1.

4 Multi-block collision attacks on 3C and 3C+

The idea of the attack on the 3C construction when the compression function is the same as in MD5 is very simple. First, we find 2-block colliding messages $(M_1||M_2)$ and $(M_1'||M_2')$ in MD5 using the attack by Wang et al [7]. Then we take the chaining variable $\mathrm{IV}_2 = \mathrm{IV}_2'$ as the initialization vector for the second run of the Wang et al algorithm. In this way we obtain another pair of messages $(M_3||M_4)$ and $(M_3'||M_4')$. The 4-block messages $(M_1||M_2||M_3||M_4)$ and $(M_1'||M_2'||M_3'||M_4')$ then form a collision for the 3C construction based on the MD5 compression function. The scheme of the attack and the distribution of differences are shown on Figure 3.

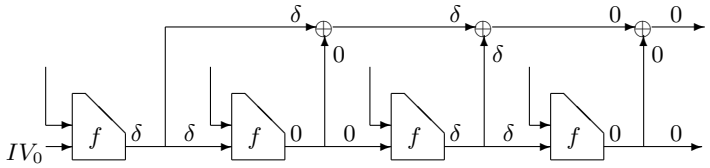

Fig. 3. 4-block internal collision attack on 3C without the final processing

A formal verification of the idea is contained in the following theorem.

Theorem 1 *Let H be an MD hash function based on a compression function f. Suppose that for some $n \geq 2$ there exists an algorithm finding n-block collisions for H that works for any initialization vector IV_0 and has the property that $\mathrm{IV}_i \oplus \mathrm{IV}_i'$ for $i = 1, \ldots, n$ is a constant independent of IV_0 and the actual colliding messages (but can be dependent on i). Then there exists an algorithm that finds $(2n)$-block collisions for the 3C construction based on the same compression function f.*

The running time of the algorithm for finding collisions in the 3C construction is twice the running time of the algorithm for finding collisions in the MD construction using the same compression function.

Proof. Let $(M_1||M_2||\cdots||M_n)$ and $(M_1'||M_2'||\cdots||M_n')$ be two colliding messages obtained by the first run of the algorithm finding collisions in H. Thus $\mathrm{IV}_n = \mathrm{IV}_n'$. We use this value as the initialization vector for the second run of the collision search algorithm for H. We obtain another pair of colliding messages $(M_{n+1}||M_{n+2}||\cdots||M_{n+n})$ and

$(M'_{n+1}||M'_{n+2}||\cdots||M'_{n+n})$. We denote the chaining variables in the second run of the algorithm by IV_{n+i} and IV'_{n+i} for $i = 1, \ldots, n$.

By our assumption on the collision search algorithm for H we can write

$$\mathrm{IV}_i \oplus \mathrm{IV}'_i = \mathrm{IV}_{n+i} \oplus \mathrm{IV}'_{n+i}$$

for every $i = 1, \ldots, n$. Thus we obtain

$$C_{2n} = \bigoplus_{i=1}^{2n} \mathrm{IV}_i$$

and

$$C'_{2n} = \bigoplus_{i=1}^{2n} \mathrm{IV}'_i.$$

Hence

$$\begin{aligned} C_{2n} \oplus C'_{2n} &= \bigoplus_{i=1}^{2n} \mathrm{IV}_i \oplus \bigoplus_{i=1}^{2n} \mathrm{IV}'_i = \bigoplus_{i=1}^{2n} (\mathrm{IV}_i \oplus \mathrm{IV}'_i) \\ &= \bigoplus_{i=1}^{n} (\mathrm{IV}_i \oplus \mathrm{IV}'_i) \oplus (\mathrm{IV}_{n+i} \oplus \mathrm{IV}'_{n+i}) \\ &= 0. \end{aligned}$$

Since $\mathrm{IV}_{2n} = \mathrm{IV}'_{2n}$, the messages $(M_1||\cdots||M_n||M_{n+1}||\cdots||M_{2n})$ and $(M'_1||\cdots||M'_n||M'_{n+1}||\cdots||M'_{2n})$ form a collision for the 3C construction based on f. □

In Section 3 we explained that the Wang et al [7] collision search algorithm for MD5 satisfied the assumptions of Theorem 1 for $n = 2$. Thus there exists an algorithm finding 4-block collisions in the 3C construction based on the MD5 compression function. The fastest implementation of the Wang et al algorithm known in the moment of writing the paper is by Klima [4] and finds collisions in MD5 in about 30 seconds in average. Thus at this moment collisions in the 3C construction based on the MD5 compression function can be found within a minute.

As for the 3C construction based on the SHA-0 compression function there is no need for running the algorithm twice to obtain a collision. Since the collision search algorithm for SHA-0 finds colliding messages of the form $(M_1||M_2)$ and $(M_1||M'_2)$, we get $\mathrm{IV}_1 = \mathrm{IV}'_1$ and $\mathrm{IV}_2 = \mathrm{IV}'_2$, thus $C_2 = C'_2$. Hence the SHA-0 collisions found by the algorithm are simultaneously collisions for the 3C construction based on the SHA-0 compression function.

Since the theoretical algorithm for finding collisions in SHA-1 proposed by Wang et al in [9] also satisfies the assumption of Theorem 1 running the algorithm twice should again produce a 4-block collision in the 3C construction based on the SHA-1 compression function.

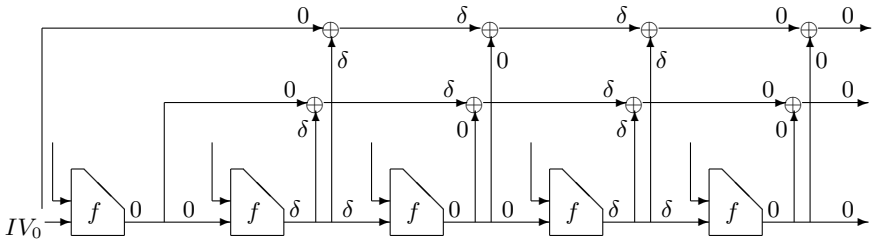

Fig. 4. 5-block collision attack on 3C+ without the final processing

The Figure 4 shows how a 5-block collision for the 3C+ construction based on the $MD5$ compression function can be found. The only difference with the collision search algorithm for the 3C construction is that we start with an arbitrary message block M_1, calculate the value of the compression function for the block with given IV_0 to obtain a new initialization vector IV_1 and then we run the collision search algorithm for the 3C construction with the initialization vector IV_1. We obtain messages $(M_1||M_2||M_3||M_4||M_5)$ and $(M_1||M_2'||M_3'||M_4'||M_5')$ such that $C_5 = C_5'$ and $\mathrm{IV}_5 = \mathrm{IV}_5'$. Since $D_5 = C_5 \oplus \mathrm{IV}_1 \oplus \mathrm{IV}_0$ and $D_5' = C_5' \oplus \mathrm{IV}_1 \oplus \mathrm{IV}_0$, we obtain also $D_5 = D_5'$.

From this observation one obtains the following theorem.

Theorem 2 *Suppose there exists an algorithm finding k-block collisions in the 3C construction based on a compression function f. Then there exists an algorithm for finding $(k+1)$-block collisions in the 3C+ construction based on the same compression function f.*

The running time of the algorithm for the 3C+ construction is equal the running time of the algorithm for the 3C+ construction plus the running time of the one calculation of the compression function.

5 Conclusion

In this paper, we have shown how to find collisions for 3C and 3C+ constructions based on a compression function f provided a collision search

algorithm for the MD construction based on f is known. We also present concrete collisions for the 3C and 3C+ constructions based on the MD5 compression function.

Acknowledgment

The authors thank to Praveen Gauravaram for sending them the preliminary version of [2] and [3] which motivated this paper.

References

1. Eli Biham, Rafi Chen, Antonie Joux, Patrick Carribault, Christophe Lemuet, and William Jalby. *Collisions of SHA-0 and reduced SHA-1.* In Ronald Cramer, editor, Advances in Cryptology – EUROCRYPT 2005, volume 3494 of Lecture Notes in Computer Science, pages 36–57. Springer, 2005.
2. Praveen Gauravaram, William Millan, Ed Dawson, and Kapali Viswanathan. *Constructing Secure Hash Functions by Enhancing Merkle-Damgård Construction.* In Lynn Batten, Reihaneh Safavi-Naini, editors, Information Security and Privacy, volume 4058 of Lecture Notes in Computer Science, pages 407–420. Springer, 2006.
3. Praveen Gauravaram, William Millan, Ed Dawson, and Kapali Viswanathan. *Constructing Secure Hash Functions by Enhancing Merkle-Damgård Construction (Extended Version).* Information Security Institute (ISI), Queensland University of Technology (QUT), number QUT-ISI-TR-2006-013, http://www.isi.qut.edu.au/research/publications/technical/qut-isi-tr-2006-013.pdf, July 2006.
4. Vlastimil Klima. *Tunnels in Hash Functions: MD5 Collisions Within a Minute*, Cryptology ePrint Archive: Report 105/2006, http://eprint.iacr.org/2006/105.
5. Jie Liang, Xuejia Lai *Improved collision attack on hash function MD5*, Cryptology ePrint Archive: Report 425/2005, http://eprint.iacr.org/2005/425.
6. Jun Yajima, Takeshi Shimoyama. *Wangs sufficient conditions of MD5 are not sufficient*, Cryptology ePrint Archive: Report 263/2005, http://eprint.iacr.org/2005/263.
7. Xiaoyun Wang, Dengguo Feng, Xuejia Lai, and Hongbo Yu. *Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD.* Cryptology ePrint Archive, Report 2004/199, 2004. http://eprint.iacr.org/2004/199.
8. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. *Effcient collision search attacks on SHA-0.* In Victor Shoup, editor, Advances in Cryptology - CRYPTO – 05, volume 3621 of Lecture Notes in Computer Science, pages 1–16. Springer, 2005, 14–18 August 2005.
9. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. *Finding collisions in the full SHA-1.* In Victor Shoup, editor, Advances in Cryptology - CRYPTO – 05, volume 3621 of Lecture Notes in Computer Science, pages 17–36. Springer, 2005, 14–18 August 2005.
10. Xiaoyun Wang and Hongbo Yu. *How to Break MD5 and Other Hash Functions.* In Ronald Cramer, editor, Advances in Cryptology – EUROCRYPT 2005, volume 3494 of Lecture Notes in Computer Science, pages 19–35. Springer, 2005.

A Examples of Collisions

IV	0x67452301	0x10325476	0x98badcfe	0xefcdab89
M_1	0x4e1a8245	0x5fe0e55d	0xfe3faa53	0x0d8546b3
	0x18ccad34	0xac0bae59	0xd59d3352	0x4805693e
	0x06342cd5	0x81b41206	0x83c2bea3	0x8fd22557
	0xc41a4cd6	0x9e9a4fe1	0x818ae34d	0x1a97e731
N_1	0x4e1a8245	0x5fe0e55d	0xfe3faa53	0x0d8546b3
	0x98ccad34	0xac0bae59	0xd59d3352	0x4805693e
	0x06342cd5	0x81b41206	0x83c2bea3	0x8fd2a557
	0xc41a4cd6	0x9e9a4fe1	0x18ae34d	0x1a97e731
IV_1	0xadebbbec	0xc85d058e	0xa2672e58	0xb91d144b
IV'_1	0x2debbbec	0x4a5d058e	0x24672e58	0x3b1d144b
$\mathrm{IV}_1 \oplus \mathrm{IV}'_1$	0x80000000	0x82000000	0x86000000	0x82000000
M_2	0x06faa233	0x1c84a4bf	0xf38ee5f1	0x08deb9af
	0x467ad36b	0x4c900712	0xd6a37d26	0x11f6de56
	0x8577e045	0x299991d5	0x5940588e	0x3fd25887
	0x301fc8fa	0x77dc0e81	0xe8c1a1a7	0x13d51d82
N_2	0x06faa233	0x1c84a4bf	0xf38ee5f1	0x08deb9af
	0xc67ad36b	0x4c900712	0xd6a37d26	0x11f6de56
	0x8577e045	0x299991d5	0x5940588e	0x3fd1d887
	0x301fc8fa	0x77dc0e81	0x68c1a1a7	0x13d51d82
$\mathrm{IV}_2 = \mathrm{IV}'_2$	0xa918ce8d	0xb7ea0df6	0x69bdb806	0x713af4de
M_3	0xcd71fe0c	0x58d0f463	0xa9399e1d	0x7db79e98
	0x3622a432	0x736cb277	0x011cb460	0x6a04e9b4
	0x06332d55	0x23f47e02	0x799ab597	0xd3ba5325
	0xb9e866e6	0xde6b9cd3	0xde6cebbb	0x0b4c3783
N_3	0xcd71fe0c	0x58d0f463	0xa9399e1d	0x7db79e98
	0xb622a432	0x736cb277	0x011cb460	0x6a04e9b4
	0x06332d55	0x23f47e02	0x799ab597	0xd3bad325
	0xb9e866e6	0xde6b9cd3	0x5e6cebbb	0x0b4c3783
IV_3	0x2b30549a	0x089c590a	0x52710661	0x6932f794
IV'_3	0xab30549a	0x8a9c590a	0xd4710661	0xeb32f794
$\mathrm{IV}_3 \oplus \mathrm{IV}'_3$	0x80000000	0x82000000	0x86000000	0x82000000
M_4	0x96ded638	0x4c1be33a	0xd46e6a5f	0xdbc8da73
	0x473af92b	0x4d0da98e	0x56dd6d3e	0xd19e7bd1
	0x53f857cd	0x4c25f191	0x918be4da	0xc09e206c
	0x320b28d4	0xcc6c0e7a	0x68515c76	0x57840834
N_4	0x96ded638	0x4c1be33a	0xd46e6a5f	0xdbc8da73
	0xc73af92b	0x4d0da98e	0x56dd6d3e	0xd19e7bd1
	0x53f857cd	0x4c25f191	0x918be4da	0xc09da06c
	0x320b28d4	0xcc6c0e7a	0xe8515c76	0x57840834
$\mathrm{IV}_4 = \mathrm{IV}'_4$	0x6a1a021a	0xc81fe980	0x88e1db5b	0x512e7c88

Table 3. Collision in 3C invoked with MD5 compression function

IV	0x67452301	0x10325476	0x98badcfe	0xefcdab89
M_1	0x0634add5	0x4074c002	0x7baaf717	0x0f522d75
	0xbf6ac0ec	0xa4885903	0x7349e78b	0x2aad1b45
	0x281dfb7e	0x173e6c0c	0xab79fc54	0x39453670
	0x44fb372b	0x4d5259c8	0xf7ad2d48	0xd1254b51
N_1	0x0634add5	0x4074c002	0x7baaf717	0x0f522d75
	0xbf6ac0ec	0xa4885903	0x7349e78b	0x2aad1b45
	0x281dfb7e	0x173e6c0c	0xab79fc54	0x39453670
	0x44fb372b	0x4d5259c8	0xf7ad2d48	0xd1254b51
$\mathrm{IV}_1 = \mathrm{IV}'_1$	0xd3f4b63c	0x595f4645	0xa890d3d0	0x9cc907db
M_2	0xa72fc176	0x64b7a050	0xe266ae7a	0x1b21009e
	0xfac1ee4c	0x9e588e8e	0x076d346d	0x805529b7
	0x0633ad55	0x02342602	0x83b4ba0b	0x56d1d924
	0x82d9651a	0xba9c8de6	0xebbbe37e	0xb78c63d5
N_2	0xa72fc176	0x64b7a050	0xe266ae7a	0x1b21009e
	0x7ac1ee4c	0x9e588e8e	0x076d346d	0x805529b7
	0x0633ad55	0x02342602	0x83b4ba0b	0x56d25924
	0x82d9651a	0xba9c8de6	0x6bbbe37e	0xb78c63d5
IV_2	0xead1c69e	0xd19e34c2	0xca2e528e	0xb1790589
IV'_2	0x6ad1c69e	0x539e34c2	0x4c2e528e	0x33790589
$\mathrm{IV}_2 \oplus \mathrm{IV}'_2$	0x80000000	0x82000000	0x86000000	0x82000000
M_3	0x6dbb34a0	0x9c1b815b	0x7ceb8ffd	0x1502296c
	0x467d585b	0x4d0d8038	0xc6db2d16	0x00d11ad5
	0xd2b2eeed	0x4a04145b	0x2f79d4aa	0x00be08a0
	0xf2e830f3	0x10bc0a85	0xe9019cb8	0x4fd512a2
N_3	0x6dbb34a0	0x9c1b815b	0x7ceb8ffd	0x1502296c
	0xc67d585b	0x4d0d8038	0xc6db2d16	0x00d11ad5
	0xd2b2eeed	0x4a04145b	0x2f79d4aa	0x00bd88a0
	0xf2e830f3	0x10bc0a85	0x69019cb8	0x4fd512a2
$\mathrm{IV}_3 = \mathrm{IV}'_3$	0x46321911	0x9d317bd2	0xfde6d50e	0xeb2170d8
M_4	0x122cdc12	0x5f60de22	0xedac78fd	0xf506f854
	0x2b85436b	0x3c980908	0xda4c144d	0x03344bbe
	0x0634ad55	0x0113f402	0x80aab777	0x13888f67
	0xadea26f7	0x623cc142	0x1192759e	0x0e74317c
N_4	0x122cdc12	0x5f60de22	0xedac78fd	0xf506f854
	0xab85436b	0x3c980908	0xda4c144d	0x03344bbe
	0x0634ad55	0x0113f402	0x80aab777	0x13890f67
	0xadea26f7	0x623cc142	0x9192759e	0x0e74317c
IV_4	0x754b85c2	0x45386ef2	0x3adad7b7	0x61523316
IV'_4	0xf54b85c2	0xc7386ef2	0xbcdad7b7	0xe3523316
$\mathrm{IV}_4 \oplus \mathrm{IV}'_4$	0x80000000	0x82000000	0x86000000	0x82000000
M_5	0x65171431	0x2615affc	0x2a2519e7	0xe2e99ce8
	0x44bcf42b	0x4c4def0e	0x47aadd22	0x127d7d56
	0x62bf776d	0x6cc9d58b	0x597058d6	0x602a5867
	0x3e2bc8ce	0xb3ec1267	0x68716155	0x17a50429
N_5	0x65171431	0x2615affc	0x2a2519e7	0xe2e99ce8
	0xc4bcf42b	0x4c4def0e	0x47aadd22	0x127d7d56
	0x62bf776d	0x6cc9d58b	0x597058d6	0x6029d867
	0x3e2bc8ce	0xb3ec1267	0xe8716155	0x17a50429
$\mathrm{IV}_5 = \mathrm{IV}'_5$	0x1453b7b0	0x803e8aee	0xfd85765e	0x176ca5d9

Table 4. Collision in 3C+ invoked with MD5 compression function