2005－ＣＳＥＣ－30 (50)
2005／7／22

MD4に対するコリジョンアタックの改良

内藤 祐介† 佐々木 悠† 國廣 昇† 太田 和夫†

† 電気通信大学 〒182-8585 東京都調布市調布ヶ丘 1-5-1

あらまし EUROCRYPT2005 で Wang らによって，MD4 への攻撃手法が提案された．Wang らの論文中ではこの攻撃は確率 2^{-6}～2^{-2}，計算量が 2^8 回以下の MD4 の演算でコリジョンメッセージを作ることができると主張している．しかし，この論文中では幾つか見落とされていた問題点があった．
そこで本稿では次の 3 点を示す．

(1) Wang らの手法には攻撃が成功するための条件の記述漏れがある．その記述漏れ部分を考慮に入れ正確な成功確率評価を得た．

(2) Wang らの手順の記述漏れ部分を修正し，成功確率を評価した．

(3) 3 ラウンド目の sufficient condition を修正できる “手戻りを削減したメッセージ変更法” を提案した．

(1) によって，Wang らの手法では確率およそ $2^{-5.61}$ で攻撃に成功することが分かり，(2) によって確率 2^{-2} で攻撃に成功する．さらに (2) と (3) を組み合わせると成功確率がほぼ 1 になる．また計算量は平均 3 回以下の MD4 の演算になる．改良方式は Wang らの手法よりおよそ 85 倍高速であること机上検討により確認した．

キーワード MD4，コリジョンアタック，メッセージ変更，sufficient condition

Improved Collision Attack on MD4 with Probability Almost 1

Yusuke NAITO†, Yu SASAKI†, Noboru KUNIHIRO†, and Kazuo OHTA†

† The University of Electro-Communications, Chofugaoka 1-5-1, Chofu-shi, Tokyo, 182-8585 Japan

Abstract In EUROCRYPT2005, a collision attack on MD4 was proposed by Wang et. al.. Wang et. al. claimed that collision messeages are found with probability 2^{-6} to 2^{-2}, and the complexity is less than 2^8 MD4 hash operations. However, there were tyops and oversights in the method of Wang et. al.. In this paper,

(1) We will evaluate the exact success probability again,

(2) We will point out the typos and oversights in Wang's method, and

(3) We will a new messege modification in third round of MD4.

From (1), we proved that the method of Wang et. al. can find collision messages with success probability $2^{5.61}$. From (2), we are able to find collision messages with the probability 2^{-2}. Also by combining the results of (2) and (3), our improved method is able to find collision messages with the probability almost 1. This complexity is less than 3 repetitions of MD4 hash operations. We confirmed that the improved method is about 85 times as fast as the method of Wang et. al..

Key words MD4, collision attack, message modification, sufficient condition

1. はじめに

MD4 は Rivest によって提案されたハッシュ関数である [8]. MD4 が提案された後，MD4 を基にした MD5, RIPEMD や SHA-family など様々なハッシュ関数が提案されてきた．MD4 は，加算，排他的論理和，シフト演算と内部変数によって構成されていて，高速に処理を行うことができるハッシュ関数である．

EUROCRYPT2005 で Wang らが MD4 への効率的な攻撃手法を提案した [10]．この手法は差分攻撃の一種である．これまで提案されてきた差分攻撃 [3], [5] は排他的論理和を差分に用いた攻撃法であるが，Wang らの手法は減算を差分に用いた差分攻撃である．減算を用いる最大の利点は，詳しくビットをみることが可能になるのでキャリーの影響も正確に扱える点である．

この手法では，入力の差分が決められており，出力の差分が無くなれば攻撃成功になる．文献 [10] では，出力差分を 0 にするために内部変数に条件を設けることで入力差分を打ち消している．以降では，この入力差分を打ち消すために内部変数に課した条件を "sufficient condition" と記すことにする．MD4 に入力されたメッセージに対して，内部変数が全て sufficient condition を満たしていれば出力差分は無くなり，確率 1 でコリジョンメッセージを作ることができる．しかしランダムに選んだメッセージが sufficient condition を満たす確率は非常に低いため工夫が必要である．

文献 [10] では sufficient condition を満たすメッセージを作るためのメッセージ変更法を提案している．この手法を組み合わせて用いることにより sufficient condition を満たす確率を大幅に向上でき，確率 2^{-6}〜2^{-2} で攻撃に成功すると主張している．

本稿では，文献 [10] の有効性を追試によって検証した．その結果，文献 [10] に攻撃成功の条件の記述漏れが 3 つあることが分かった．

（1） 単独でメッセージ変更手法を行うと sufficient condition を満たすが，他の sufficient condition に対してのメッセージ変更と同時に行うと sufficient condition を満たさない場合があること (4.1.1 章)

（2） Wang らのメッセージ変更手法では成功しない変更法があること (4.1.2 章)

（3） Wang らの sufficient condition で欠けている条件があることである (4.2 章)

よって，我々の第 1 の成果として，この 3 つの条件の記述漏れを考慮に入れて Wang らの手法の正確な確率評価を行い，この手法の成功確率はおよそ $2^{-5.61}$ になることを示す (4.3.1 章)．第 2 の成果として，この条件の記述漏れ部分を修正して，確率 2^{-2} で攻撃に成功することを示す (4.3.2 章)．第 3 の成果として，3 ラウンド目の sufficient condition を修正するメッセージ変更法を提案する (5 章)．これらの改良を組み合わせることで，成功確率をほぼ 1 にすることができる．また計算量は 3 回以下の MD4 の演算で攻撃に成功する．改良方式は Wang らの手法よりもおよそ 85 倍高速であり，これまで提案されている MD4 への攻撃の中で最も効率的な手法であることを机上検討によって確認した．

なお，EUROCRYPT2005 では WangMD4 へのコリジョンアタックと同じ手法を用いた MD5 へのコリジョンアタックを発表した [11]．MD5 は MD4 と同じような構造をしているので，我々の手法が MD5 へ適用できることが期待できる．この詳しい内容は別途報告予定である．

2. MD4 の概要

MD4 は任意長のメッセージから 128 ビットのランダムな値を生成するハッシュ関数である．メッセージ M が MD4 に入力されたとき以下の手順でハッシュ値を計算する．

（1） メッセージ M を 512 ビットごとのメッセージ $M_1, M_2, ..., M_n$ に分ける．

$$M = (M_1, M_2, ..., M_n),\ |M_i| = 512$$

（2） h_i を MD4 の i ブロック目の出力値とし，M_i と h_{i-1} を圧縮関数に入力して入手する．この計算を全ての M_i について行う．ここで初期値 h_0 は以下の通りである．

$$h_0 = (0x67452301, 0xefcdab89, 0x98badcfe, 0x10325476)$$

（3） 最後のブロックの出力 h_n が MD4 の出力値になる．

圧縮関数

圧縮関数に入力したメッセージ M_j を 16 個の 32 ビットのメッセージ $m_0, ..., m_{15}$ に分ける．圧縮関数は 48 ステップあり，ステップ 1 からステップ 16 までを 1 ラウンド目，ステップ 17 からステップ 32 までを 2 ラウンド目，そしてステップ 33 からステップ 48 までを 3 ラウンド目と呼ぶ．それぞれのステップで内部変数 a, b, c, d を $a_1, d_1, c_1, b_1, a_2, ...$ という順番でステップごとに計算する．このとき圧縮関数の出力値 h_i は $h_i = (a_0 + a_{16}, b_0 + b_{16}, c_0 + c_{16}, d_0 + d_{16})$ となる．

次に内部変数の計算法を説明する．ϕ 関数を以下のように定義する．

$$\phi(x, y, z, w, m, s, t) = ((x + f(y, z, w) + m + t) \bmod 2^{32}) \lll s$$

m は各ステップで入力するメッセージであり，ステップごとにどのメッセージを入力するかは予め定められている．s は各ステップで定まっているシフト数である．

t の値は 1 ラウンド目を $t = 0$, 2 ラウンド目を

$t = 0x5a827999$，3 ラウンド目を $t = 0x10325476$ とする．次に，関数 f は 1 ラウンド目は $f(y,z,w) = F(y,z,w)$，2 ラウンド目は $f(y,z,w) = G(y,z,w)$，3 ラウンド目は $f(y,z,w) = H(y,z,w)$ と定め，それぞれの関数を以下で定義する．

$$F(y,z,w) = (y \wedge z) \vee (\neg y \wedge w)$$

$$G(y,z,w) = (y \wedge z) \vee (y \wedge w) \vee (z \wedge w)$$

$$H(y,z,w) = y \oplus z \oplus w$$

このとき内部変数 a_i, b_i, c_i, d_i を以下のように計算する．

$$a_i = \phi(a_{i-1}, b_{i-1}, c_{i-1}, d_{i-1}, m, s, t)$$

$$d_i = \phi(d_{i-1}, a_i, b_{i-1}, c_{i-1}, m, s, t)$$

$$c_i = \phi(c_{i-1}, d_i, a_i, b_{i-1}, m, s, t)$$

$$b_i = \phi(b_{i-1}, c_i, d_i, a_i, m, s, t)$$

3. Wang らの MD4 に対する攻撃

Wang らの手法は，減算を差分に用いた差分攻撃である．入力差分 ΔM を出力までに打ち消し，出力差分を 0 にすることでコリジョンメッセージを作る．通常，入力差分が出力時に打ち消される確率は非常に低い．文献 [10] では，ある入力差分を打ち消すことができるような内部変数の条件を利用する．

例えば，m_1 の差分として $\Delta m_1 = 1_{32}$ を考える．ここで，1_i は先頭から i ビット目が 1 でそれ以外のビットが 0 である 32 ビットの 2 進数である．この差分によって 2 ステップ目で計算される d_1 の 7 ビット目に $0 \rightarrow 1$ という差分が表れる．差分入力後の d_1 の値を $d_1[7]$ と記述する．また 7 ビット目が 1 から 0 に変化する場合 $d_1[-7]$ と記述する．d_1 は以下のように a_2 の計算に使われるので a_2 に差分が表れる可能性がある．

$$a_2 = (a_1 + F(b_1, c_1, d_1) + m_4) \ggg 3$$

この差分を出力で打ち消したいので，関数 F の仕様に注目する．$F(b_1, c_1, d_1) = (b_1 \wedge c_1) \vee (\neg b_1 \wedge d_1)$ なので $b_{1,7} = 1$ であれば $F(b_1, c_1, d_1) = F(b_1, c_1, d_1[7])$ となって，d_1 の変更を除去できる．すなわち $b_{1,7} = 1$ を内部変数の条件に設けることで，差分を打ち消すことができる．そこで b_1 の 7 ビット目を $b_{1,7}$ と記述した．

このような操作を各差分ごとに行うことにより，入力差分を打ち消すことができる．しかし，ランダムに選んだメッセージが内部変数に課せられた条件を成立させる確率は非常に低いので，次に述べるメッセージ変更によって内部変数に課せられた条件を満たす確率を上げることを試みる．

3.1 Wang らの攻撃の手法

Wang らの攻撃はメッセージ差分として以下の値を用いる．

$$\Delta M = M^* - M' = (\Delta m_0, \Delta m_1, ..., \Delta m_{15})$$

$$\Delta m_1 = 1_{32}, \Delta m_2 = 1_{32} - 1_{28}, \Delta m_{12} = -1_{17}$$

$$\Delta m_i = 0, 0 \leqq i \leqq 15, i \neq 1, 2, 12$$

sufficient condition は 122 個ある．sufficient condition はメッセージ変更を行うことによりほとんど満たすことができる．しかし，1 ラウンド目と 2 ラウンド目にメッセージ変更を行うが，3 ラウンド目のメッセージ変更は行わないので，sufficient condition を満たせない場合がでてくる．この場合には，メッセージを取り直し，再びメッセージ変更を行う．この取り直すメッセージは 1 ラウンド目のステップ 1 からステップ 14 までのメッセージ変更の手順を省くために m_{14} と m_{15} とする．コリジョンメッセージ (M', M^*) の生成法は以下の通りである．

（1） 512 ビットのランダムなメッセージ M を生成

（2） sufficient condition を満たすようにメッセージ M を M' に変更

（3） 全ての sufficient condition を満たせば，変更されたメッセージ M' をコリジョンメッセージとして出力し，どこか 1 つでも sufficient condition を満たさなければ M' の m_{14} と m_{15} をそれぞれランダムな 32 ビットのメッセージに変更し，ステップ 2 に戻る

（4） $M^* = M' + \Delta M$ を計算

（5） M' と M^* がコリジョンメッセージになるので出力

3.1.1 メッセージ変更方法

ステップ 2 の手順は以下の通り．メッセージ変更は sufficient condition を満たすために行うものであり，single-step modification と multi-step modification の 2 つある．single-step modification は 1 ラウンド目で行うメッセージ変更である．この変更は他の条件に干渉することはないので容易に行うことができる．multi-step modification は 2 ラウンド目で行うメッセージ変更である．2 ラウンド目のメッセージ変更は 1 ラウンド目にも影響を与えてしまうので，この影響を 1 ラウンド目に与えないように修正する必要が生じる．また，2 ラウンド目のメッセージ変更で新たに extra condition を課すこともある．

Single-Step Modification

single-step modification は 1 ラウンド目の sufficient condition を満たすために用いるメッセージ変更法である．例えば，3 ステップ目の sufficient condition($c_{1,7} = 1, c_{1,8} = 1, c_{1,11} = 0, c_{1,26} = d_{1,26}$) を満たす方法を考える．$c_1$ の計算は $c_1 = (c_0 + F(d_1, a_1, b_0) + m_2) \lll 11$ である．c_1 の計

算では，直接的に値を変更できるのは m_2 だけである．よって，以下のように m_2 を変更して 3 ステップ目の sufficient condition を満たすようにする．

$$c_1^{new} \leftarrow c_1 \oplus (c_{1,7} \oplus 1_7) \oplus (c_{1,8} \oplus 1_8) \oplus c_{1,11} \oplus (c_{1,26} \oplus d_{1,26})$$
$$m_2 \leftarrow (c_1^{new} \ggg 11) - c_0 - F(d_1, a_1, b_0)$$

Multi-Step Modification

multi-step modification は 2 ラウンド目の sufficient condition を満たすために用いるメッセージ変更法である．例えば，17 ステップ目の sufficient condition の $a_{5,19} = c_{4,19}$ を満たす方法を考える．このとき，a_5 の計算は $a_5 = a_4 + G(b_4, c_4, d_4) + m_0 + 0x5a827999$ である．a_5 の計算で直接変更できる値は m_0 のみである．よって，$a_{5,19} = c_{4,19}$ の条件を満たしていない場合は，この条件を満たすために以下のように m_0 を変更する．

$$m_0 \leftarrow m_0 \pm 1_{16}$$

この変更で $a_{5,19} = c_{4,19}$ を満たすことができるが，m_0 は 1 ステップ目の計算でも使用されるので，m_0 の変更の影響が以下のように 1 ステップ目に生じてしまう．

$$m_0 \leftarrow m_0 \pm 1_{16} \Rightarrow a_1^{new} = a_1[\pm 19]$$

この影響によって sufficient condition を満たしている内部変数まで変更されてしまう．そのため，m_0 の変更の影響が他の満たされている sufficient condition に加わらないように次のようなメッセージの変更を行う．

$$m_0 \leftarrow m_0 \pm 1_{16} \Rightarrow a_1^{new} = a_1[\pm 19]$$
$$m_1 \leftarrow (d_1 \ggg 7) - d_0 - F(a_1^{new}, b_0, c_0)$$
$$m_2 \leftarrow (c_1 \ggg 11) - c_0 - F(d_1, a_1^{new}, b_0)$$
$$m_3 \leftarrow (b_1 \ggg 19) - b_0 - F(c_1, d_1, a_1^{new})$$
$$m_4 \leftarrow (a_2 \ggg 3) - a_1^{new} - F(b_1, c_1, d_1)$$

この変更によって，m_0 の変更の 1 ラウンド目への影響を打ち消すことができる．本稿ではこの変更方法を “multi-step modification(1)” と呼ぶことにする．

しかし，multi-step modification(1) では変更できないような sufficient condition が 2 ラウンド目には存在する．例えば，ステップ 19 の sufficient condition の $c_{5,27} = d_{5,27}$ を考えてみる．このとき multi-step modification(1) を適用すると，以下のようにメッセージ変更が 1 ラウンド目に影響する．

$$m_8 \leftarrow m_8 \pm 1_{18} \Rightarrow a_3^{new} = a_3[\pm 21]$$

$a_{3,21}$ に関する sufficient condition があるので，この変更を行ってしまうと $a_{3,21}$ の sufficient condition を崩してしまい，コリジョンメッセージを作ることはできなくなる．よってこの場合には，m_8 の変更が $a_{3,21}$ に影響しないように，表 1 ようなメッセージ変更を行う．ここで extra condition は 2 ラウンド目の変更の影響が 1 ラウンド目の sufficient condition に及ぶのを防ぐために，2 ラウンド目で変更を行うかに関わらず予め 1 ラウンド目実行時に新たに追加した条件である．

表 1 “$c_{5,27}$” の修正

step	shift	Modify m_i	Chaining value after message modification	Extra Conditions
6	7	$m_5 \leftarrow m_5 + 1_{11}$	$d_2[18], a_2, b_1, c_1$	$d_{2,18} = 0$
7	11		$c_2, d_2[18], a_2, b_1$	$a_{2,18} = b_{1,18}$
8	19		$b_2, c_2, d_2[18], a_2$	$c_{2,18} = 0$
9	3	$m_8 \leftarrow m_8 - 1_{18}$	$a_3, b_2, c_2, d_2[18]$	$b_{2,18} = 0$
10	7	$m_9 \leftarrow m_9 - 1_{18}$	d_3, a_3, b_2, c_2	

この変更や設定を行った場合 a_3 の計算で用いる関数 F の 18 ビット目が以下のように変化する．

$$F(b_{2,18}, c_{2,18}, d_{2,18})_{18} = 0 \rightarrow 1$$

この $F(b_{2,18}, c_{2,18}, d_{2,18})$ の変化により m_8 の変更の a_3 への影響を打ち消すことができる．よってこの方法を用いると $c_{5,27} = d_{5,27}$ を満たすことができ，かつ $a_{3,21}$ に影響を与えないようにすることができる．本稿ではこの変更方法を “multi-step modification(2)” と呼ぶことにする．

4. Wang らの手法の正確な確率評価と改善

Wang らは文献 [10] で確率 $2^{-6} \sim 2^{-2}$ で MD4 のコリジョンメッセージを作ることができると主張しているが，彼らの確率評価は不十分である．そこで，我々は Wang らの手法を細かく解析し，正確な確率評価を行った．

4.1 Wang らの手法における攻撃成功の条件の見落としとその改善法

Wang らは様々なメッセージ変更方法を提案したが，文献 [10] の手法を詳しく解析した結果 Wang らの修正では sufficient condition を満たせない場合があることが分かった．以下 sufficient condition を満たさない理由を説明し，その改善方法を示す．

4.1.1 排他的な変更

文献 [10] で提案されている手法を用いた場合，ある 2 つの条件を満たすためにメッセージ変更を行うと，メッセージ変更に失敗する場合がある．sufficient condition “$d_{5,19} = a_{5,19}$” と “$c_{5,26} = d_{5,26}$” の修正，“$c_{5,32} = d_{5,32}$” と

"$c_{6,32} = d_{6,32} + 1$" の修正である．そこで，我々はこの修正が失敗する理由を説明し，この改善方法を提案する．

(1) 問題点

sufficient condition"$d_{5,19} = a_{5,19}$" と "$c_{5,26} = d_{5,26}$" の修正について説明する．$d_{5,19}$ の修正に関して，Wang らは multi-step modification(1) を使用する．この場合，$m_4 \leftarrow m_4 \pm 1_{14}$ という変更が行われ，その結果 $a_{2,17}$ の値が変化する．次に $c_{5,26}$ の修正は Wang らの手法では multi-step modification(2) を使用する．この場合，extra condition として $a_{2,17} = b_{1,17}$ を設定する必要がある．しかし，両方の変更を行った場合 $d_{5,19}$ の修正により $a_{2,17}$ の値が変化し，extra condition として設定されていた $a_{2,17} = b_{1,17}$ が崩れてしまう．この extra condition は変更の影響を打ち消すためすなわち c_2 が変更されないように設定していたが，この問題により c_2 が満たしてほしい値とは別の値に変化してしまう場合がある．

(2) 修正に成功する確率

問題が発生する確率は $d_{5,19}$ と $c_{5,26}$ の両方を修正した場合なので $\frac{1}{4}$ である．しかし，問題が発生しても $d_{6,26}$ の修正が行われた場合 $d_{5,19}$ と $c_{5,26}$ の修正に成功する．以下にその理由を説明する．

$d_{5,19}$ と $c_{5,26}$ 両方を修正し，$d_{6,26}$ の修正が行われた場合 $c_{5,26}$ の修正に成功する．extra condition $a_{2,17} = b_{1,17}$ が崩れることにより c_2 にその影響が出てしまうことが問題であるが，$d_{6,29}$ 修正中で m_6 が以下のように変化するので影響を打ち消すことができる．

$$m_6 \leftarrow (c_2 \ggg 11) - c_1 - f(d'_2, a_2, b_1)$$

この変更は c_2 の値が変更しないように行われるので，この変更によって extra condition $a_{2,17} = b_{1,17}$ が崩れた影響を打ち消すことができる．よって，$d_{5,19}$ と $c_{5,26}$ 修正時の両方の修正が起きた場合でも $d_{6,29}$ の修正が起きれば $c_{5,26}$ の修正に成功する．よって，$c_{5,26}$ の修正に成功する確率は以下の通りである．

$$1 - \frac{1}{4} + \frac{1}{8} = \frac{7}{8}$$

同様に，$c_{5,32}$ と $c_{6,32}$ の修正を行った場合同じような問題が生じる．このとき $c_{5,32}$ と $c_{6,32}$ の修正が成功する確率は $\frac{3}{4}$ である．

(3) 改善策

$d_{5,19}$ と $c_{5,26}$ の修正の改善

Wang らの手法では上述のように題が発生するので，我々は $d_{5,19}$ の修正を multi-step modification(2) で行う方法を提案する．詳細は表 2 に示す．これにより，他の条件に影響を及ぼさずに修正ができるので，$d_{5,19}$ の修正と $c_{5,26}$ の修正が成功する確率は 1 になる．

表 2 "$d_{5,19}$" の修正

step	shift	Modify m_i	Chaining value after message modification	Extra Conditions
2	7	$m_1 \leftarrow m_1 + 1_7$	$d_1[14], a_1, b_0, c_0$	$d_{1,14} = 0$
3	11		$c_1, d_1[14], a_1, b_0$	$a_{1,14} = b_{0,14}$
4	19		$b_1, c_1, d_1[14], a_1$	$c_{1,14} = 0$
5	3	$m_4 \leftarrow m_4 - 1_{14}$	$a_2, b_1, c_1, d_1[14]$	$b_{1,14} = 0$
6	7	$m_5 \leftarrow m_5 - 1_{14}$	d_2, a_2, b_1, c_1	

$c_{5,32}$ と $c_{6,32}$ の修正の改善

Wang らの手法では上述のように問題が発生するので，我々は表 3 のように $c_{5,32}$ の修正法を変えることで $c_{5,32}$ と $c_{6,32}$ の両方の修正が行われた場合でも問題が生じないようにする．

表 3 "$c_{5,32}$" の修正

step	Modify m_i	Chaining value after message modification	Extra Conditions
15	$m_{14} \leftarrow m_{14} + 1_{11}$	$c_4[22], d_4, a_4, b_3$	$c_{4,22} = 0$
16		$b_4, c_4[22], d_4, a_4$	$d_{4,22} = a_{4,22}$
17		$a_5, b_4, c_4[22], d_4$	$b_{4,22} = d_{4,22}$
18		$d_5, a_5, b_4, c_4[22]$	$a_{5,22} = b_{4,22}$
19		$c_5^{new} = c_5 + 1_{31}, d_5, a_5, b_4$	$c_{5,31} = 1$

$c_{5,32}$ を変更する場合，19 ステップ目で使われるメッセージを変更する修正法やキャリーを用いない変更法が考えられるが，これらの変更法は他の条件に影響を与えてしまうため，表 3 の桁上がりを使った修正法を用いることにした．この変更によって，他の設定に影響を与えずにかつ $c_{5,32}$ を修正することができる．よって，$c_{5,32}$ と $c_{6,32}$ の修正は確率 1 で成功する．

4.1.2 sufficient condition を崩す変更

(1) 問題点

文献 [10] の $c_{5,29}$ の修正は multi-message modification(2) を用い，この変更によって $d_{2,20}$ が 0 から 1 に変更される．しかし，sufficient condition $d_{2,20} = a_{2,20}$ がすでに設定されているので，このメッセージ変更を行った場合，$d_{2,20} = a_{2,20}$ が崩れて，コリジョンメッセージを作ることができなくなる．

(2) 改善策

c_5 は定義より以下のように計算される．

$$c_5 = (c_4 + G(d_5, a_5, b_4) + m_8 + 0x5a827999) \lll 9$$

このとき $c_{5,29} = 1$ を満たすように m_8 を変更すると，他の条件に影響を与えてしまうので，$c_{4,20}$ を変更することに

表 4 "$c_{5,29}$"の修正

step	Modify m_i	Chaining value after message modification	Extra Conditions
15	$m_{14} \leftarrow m_{14} + 1_9$	$c_4[20], d_4, a_4, b_3$	$c_{4,20} = 0$
16		$b_4, c_4[20], d_4, a_4$	$d_{4,20} = a_{4,20}$
17		$a_5, b_4, c_4[20], d_4$	$b_{4,20} = d_{4,20}$
18		$d_5, a_5, b_4, c_4[20]$	$a_{5,20} = b_{4,20}$
19		$c_5^{new} = c_5 + 1_{29}, d_5, a_5, b_4$	

より sufficient condition$c_{5,29} = 1$ を満たすようにする．c_4 は

$$c_4 = (c_3 + F(d_4, a_4, b_3) + m_{14}) \lll 11$$

と計算されるので，$m_{14,9}$ を変更することにより $c_{4,20}$ を変更することができる．よって表 4 のようなメッセージ変更を行うことにより，他の条件に影響を与えずに $c_{5,29}$ を修正することができ，このメッセージ変更は確率 1 で成功する．

4.2 sufficient condition の欠如

入力差分から sufficient condition を求めたところ，文献 [10] の表 5 の sufficient condition には $a_{6,30} = 0$ と $b_{4,32} = c_{4,32}$ の sufficient condition が欠けていることが分かった．以下ではこの sufficient condition の導出法を説明する．

(1) sufficient condition"$a_{6,30} = 0$" の導出

表 5 "$a_{6,30} = 0$" の導出

Step	Shift	Δm_i	The i-th output for M'
21	3	2^{31}	$a_6[-29, 30, -32]$

まず $a_{6,30} = 0$ について説明する．表 5 は文献 [10] の表 5 の一部である．この表の中に $a_6[30]$ という内部状態があるが，これは $a_{6,30}$ が差分により 0 から 1 に変化することを意味するので，sufficient condition に $a_{6,30} = 0$ が必要である．

$a_{6,30} = 0$ は他の a_6 の修正と同様の方法で修正を行う．

(2) sufficient condition"$b_{4,32} = c_{4,32}$" の導出

表 6 "$b_{4,32} = c_{4,32}$" の導出

Step	Shift	Δm_i	The i-th output for M'
14	7		$d_4[-27, -29, 30]$
15	11		c_4
16	19		$b_4[19]$
17	3		$a_5[-26, 27, -29, -32]$
18	5		d_5

次に $b_{4,32} = c_{4,32}$ について説明する．表 6 は文献 [10] の表 5 の一部である．内部変数に $a_5[-32]$ という差分があるが，d_5 には差分が存在していない．よって a_5 の差分 $a_5[-32]$ を d_5 で打ち消す必要がある．この差分 $a_5[-32]$ は次のように打ち消す．以下，その方法を論証する．d_5 は以下のように計算する．

$$d_5 \leftarrow ((d_4 + G(a_5, b_4, c_4) + m_4 + 0x5a827999) \mod 2^{32}) \lll 5.$$

関数 G は以下の通りである．

$$G(y, z, w) = (y \wedge z) \vee (y \wedge w) \vee (z \wedge w).$$

関数 G は a_5 使っているので，関数 G で使われる a_5 以外の値を都合よく設定することで差分 $a_5[-32]$ を打ち消すことにする．よって，関数 G は多数決関数なので，sufficient condition に $b_{4,32} = c_{4,32}$ を設定すれば差分 $a_5[-32]$ を打ち消すことができる．

4.3 Wang らの手法の再評価と改善結果

4.3.1 Wang らの手法の正確な確率評価

文献 [10] に含まれる問題点を考慮に入れて確率評価を行うと以下のようになる．

- $b_{4,32} = c_{4,32}$ と $a_{6,30} = 0$ の sufficient condition の欠如を考慮に入れた成功確率 (4.2 章): $\left(\frac{1}{2}\right)^2$
- $c_{5,29}$ の修正の誤りを考慮に入れた成功確率 (4.2 章): $\frac{1}{2}$
- $d_{5,19}$ と $c_{5,26}$，$c_{5,32}$ と $c_{6,32}$ の修正の問題点を考慮に入れた成功確率 (4.1.2 章) : $\frac{7}{8} \times \frac{3}{4}$
- 3 ラウンド目の sufficient condition$b_{9,32} = 1$ と $a_{10,32} = 1$ を満たす確率 (4.1.1 章): $\left(\frac{1}{2}\right)^2$

以上より文献 [10] の正確な成功確率は以下の通りである．

$$Pr[success] = \frac{3}{4} \times \frac{7}{8} \times \frac{1}{2} \times \left(\frac{1}{2}\right)^2 \times \left(\frac{1}{2}\right)^2$$
$$= 2^{-5.61}$$

4.3.2 改善結果

我々は Wang らの手法の条件の見落としを以下のように改善した．

- sufficinet condition に $b_{4,32} = c_{4,32}$ と $a_{6,30} = 0$ を追加

表 7　手戻りを削減したメッセージ変更法

step	Modify m_j	Chaining value after message modification	Extra Conditions in 1^{st} round
12	$m_{11} \leftarrow m_{11} \pm 1_i$	$b_3[\pm(i+19)], c_3, d_3, a_3$	
13		$a_4, b_3[\pm(i+19)], c_3, d_3$	$c_{3,i+19} = d_{3,i+19}$
14		$d_4, a_4, b_3[\pm(i+19)], c_3$	$a_{4,i+19} = 0$
15		$c_4, d_4, a_4, b_3[\pm(i+19)]$	$d_{4,19} = 0$
16	$m_{15} \leftarrow (b_4 \ggg 19) - b_3[\pm(i+19)] - F(c_4, d_4, a_4)$	b_4, c_4, d_4, a_4	

- $d_{5,19} = a_{5,19}$ と $c_{5,26} = d_{5,26}$ に関して同時にメッセージ変更できるように $d_{5,19}$ のメッセージ変更方法を改善
- $c_{5,32} = d_{5,32}$ と $c_{6,32} = d_{6,32} + 1$ に関して同時にメッセージ変更できるような $c_{5,32}$ のメッセージ変更方法を提案

この結果，2 ラウンド目の sufficinet condition の修正時のメッセージ変更は全て確率 1 で成功する．この改善により，成功確率は 2^{-2} となる．計算量に関して，改善方式では extra condition が増えるので数ステップ分だけ計算量が増えるが，確率が下がったので，全体の計算量は減少する．

5. 手戻りを削減したメッセージ変更法

5.1 アイデア

multi-step modification を用いた場合，メッセージ変更の影響が出るのは 1 ラウンド目のみなので，1 ラウンド目だけこの影響を打ち消すようにすればメッセージ変更は成功した．

しかし，3 ラウンド目で同様の方法によりメッセージ変更を行うと，変更の影響を 1 ラウンド目と 2 ラウンド目の両方について考えなければならないため困難である．文献 [10] では 3 ラウンド目の sufficient condition を満たすメッセージ変更方法は提案されていない．そこで，その困難を回避するため 2 ラウンド目のメッセージを複数回変更することにより 3 ラウンド目の sufficient condition を確率的に満たす方針を採用することにした．

Wang らの方式ではメッセージの生成に失敗した場合は 1 ラウンド目に戻って m_{14} と m_{15} を取り直し再び 2 ラウンド目の multi-step modification を行っていたが，提案方式では 2 ラウンド目のメッセージを取り直すことにより 2 ラウンド目の multi-step modification の処理数だけ手戻りを削減することが期待できる．

5.2 考　　察

$m_{11} \leftarrow m_{11} \pm 1_i$ を考えてみる．まず我々はこの変更を加えたとき 3 ラウンド目の sufficient condition が満たされる確率を計算機実験を用いて検証した．その結果，i の値に関わらず確率約 $\frac{1}{4}$ で修正できることを確認した．次に，m_{11} は 1 ラウンド目の 12 ステップ目で使われるので，m_{11} の変更を打ち消すようなメッセージ変更が必要になる．我々は表 7 のようなメッセージ変更を行い m_{11} の変更の影響を打ち消した．

この変更を行っても他の設定に影響を与えない i は $5, 6, 12, 14, 15, ..., 28, 31$ の 19 箇所である．よって，m_{11} を修正できる回数は，19 箇所の全ての組み合わせの数だけ存在する．すなわち 2^{19} 回である．複数回 m_{11} に変更を加えると 3 ラウンド目の sufficient condition を満たす確率は非常に高くなる．3 ラウンド目の sufficient condition を修正できる確率は

$$1 - \left(1 - \frac{1}{4}\right)^{2^{19}} \fallingdotseq 1$$

となる．よって 3 ラウンド目の sufficient condition は確率がほぼ 1 で修正可能になる．

Remark

m_{11} を変更する理由は，m_{11} は 2 ラウンド目の最後から 2 番目のステップで用いられるので，他のメッセージを変更するよりもステップ数を減らすことが期待できるからである．また，2 ラウンド目の最後のステップのメッセージ m_{15} を選ばなかった理由は，m_{15} を変更すると，この変更の 1 ラウンド目における影響を消すことができないからである．

6. 改良結果

我々は，4 章および 5 章で確率 1 で sufficient condition を満たすメッセージ変更法を提案した．改良箇所は以下の通りである．

- sufficinet condition に $b_{4,32} = c_{4,32}$ と $a_{6,30} = 0$ を追加
- $d_{5,19} = a_{5,19}$ と $c_{5,26} = d_{5,26}$ に関して同時にメッセージ変更できるように $d_{5,19}$ のメッセージ変更方法を改善
- $c_{5,32} = d_{5,32}$ と $c_{6,32} = d_{6,32} + 1$ に関して同時にメッセージ変更できるように $c_{5,32}$ のメッセージ変更方法を改善
- 3 ラウンド目の sufficient condition の成功確率を高めるための “手戻りを削減したメッセージ変更法” を提案

我々の改良方式では確率 1 でコリジョンメッセージを作る

表 8　改良方式で生成したコリジョンメッセージの例

M	0x24ce9d37de4dfca0a3b88fc39c9f9e5c92ee86ada2c9e8b088f3a020c5368a69
	0e503cc80c2368f978ff57bf21a1762ad018afb8daa431e9308bf382806a18a1
M'	0x368b9d377e2dfc60b5b88fcb0c8fbe5601a6662d9ecc3929aa35aabf887f929f
	2740a2c8c8c12039bbb401bdc1983331e45e1f61c150d565ee27d04af1dfec4c
M^*	0x368b9d37fe2dfc6025b88fcb0c8fbe5601a6662d9ecc3929aa35aabf887f929f
	2740a2c8c8c12039bbb401bdc1983331e45d1f61c150d565ee27d04af1dfec4c
$H(M')$	0x23082a62358603c776b33ed295e220d9
$H(M^*)$	0x23082a62358603c776b33ed295e220d9

ことができる．

我々の方式の計算量は，次の通りである．まず，標準で1回は必ずハッシュ関数を計算するので，1回の MD4 の演算は必ずかかる．single-step modification で数ステップかかり，multi-step modification はメッセージ変更で約 26 ステップ，内部変数の再計算に約 13 ステップかかる．3 ラウンド目のメッセージ変更に関しては，m_{11} を変更する繰り返しの平均回数は 3 回で，1 回の修正にかかるステップ数は 8 ステップなので，平均 24 ステップで 3 ラウンド目の修正が可能である．また，m_{11} の変更の影響を打ち消すための操作で数ステップかかる．よって，計算量は 1MD4 演算 +26 ステップ +13 ステップ +24 ステップ + 数ステップであり 3 回以下の MD4 の演算でコリジョンメッセージを作ることが可能になる．

Wang らの手法の計算量は最大確率 2^{-6} とメッセージ変更に 2 回の MD4 演算が必要なことから 2^8 以下なので，改良方式の計算量は $\frac{2^8}{3} \fallingdotseq 85$ 倍高速である．

Remark

始めにランダムメッセージを生成せずに，1 ラウンド目の sufficient condition を満たす内部変数を生成してからメッセージを生成すると計算量を減らすことができる．

7. ま と め

Wang らは確率 2^{-6} から 2^{-2}，計算量が 2^8 以下の MD4 の演算でコリジョンメッセージを作ることができると主張している．しかし，この確率評価は不十分であった．本論文では Wang らの方式を再評価した．その結果，Wang らの方式では確率 $2^{-5.61}$ でコリジョンメッセージを作ることができることを示した．次に，Wang らの方式の条件の記述漏れを修正した．その結果，確率 2^{-2} で攻撃に成功した．最後に，3 ラウンド目の sufficient condition を修正する方法を提案した．この結果，計算量が 3 回以下の MD4 の演算でコリジョンメッセージを作ることを示した．

表 8 に Wang らの手法では生成できないコリジョンメッセージを紹介する．M は変更する前のメッセージであり，M' はメッセージ変更後のメッセージであり，M^* は $M^* = M' + \Delta M$ である．$H(M')$ は M' を MD4 に入力したときのハッシュ値であり，$H(M^*)$ は M^* を MD4 に入力したときのハッシュ値である．

文　献

[1] E. Biham, A. Shamir: Differential Cryptanalyss of Snefru, Khafre, REDOC-II, LOKI and Lucifer, CRYPTO'91, LNCS 576, pp. 156–171, 1992

[2] E. Biham, R. Chen: Near collision of SHA-0, CRYPTO2004, LNCS 3152, pp290–305, 2004.

[3] E. Biham, R. Chen, A. Joux, P. Carribault: Collisions of SHA-0 and Reduced SHA-1, EUROCRYPT2005, LNCS3494, pp.36–57, 2005.

[4] B. den Boer, A. Bosselaers: Collisions for the compression function of MD5, EUROCRYPT'93, 1993

[5] F. Charband, A. Joux: Differential Collisions in SHA-0, CRYPTO'98, 1998

[6] H. Dobbertin: Cryptanalysis of MD4, First Softwere Encryption 1996, LNCS 1039, 1996

[7] H. Dobbertin: The First Two Rounds of MD4 are Not One-Way, Fast Software Encryption 1998, 1998.

[8] Y. Naito, Y. Sasaki, N. Kunihiro, K. Ohta: Improved Collision Attack on MD4, e-Print, 2005.

[9] R. Rivest: The MD4 Message Digest Algorithm, CRYPTO'90 Proceedings, 1991, http://theory.lcs.mit.edu/ rivest/Rivest-MD4.txt

[10] X. Wang, D. Feng, X. Lai, H. Yu: Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD, rump session, CRYPTO 2004, e-Print, 2003.

[11] X. Wang, X. Lai, D. Feng, H. Chen, X. Yu: Cryptanalysis of the Hash Functions MD4 and RIPEMD, Advances in EUROCRYPT2005, LNCS 3494, pp. 1–18, 2005.

[12] X. Wang, H. Yu: How to break MD5 and Other Hash Functions, Advances in EUROCRYPT2005, LNCS 3494, pp. 19–35, 2005.