doi:10.3969/j.issn.1000-436x.2014.02.006

# 带消息填充的 29 步 SM3 算法原根和伪碰撞攻击

王高丽[1,2]，申延召[1]
（1.东华大学 计算机科学与技术学院，上海 201620；
2.中国科学院 信息工程研究所 信息安全国家重点实验室，北京 100093）

**摘 要**：基于中间相遇攻击技术，提出了一种针对密码杂凑函数 SM 算法的原根攻击和伪碰撞攻击方法，给出了从第 1 步开始的带消息填充的 29 步 SM3 算法的原根攻击和伪碰撞攻击。结果表明：对于 29 步 SM3 算法的原根攻击的时间复杂度为 $2^{254}$；对于 29 步 SM3 伪碰撞攻击的时间复杂度为 $2^{125}$。说明从第 1 步开始的带消息填充的 29 步 SM3 算法不能抵抗原根攻击和伪碰撞攻击。

**关键词**：杂凑函数；原根攻击；碰撞攻击；中间相遇攻击；SM3
**中图分类号**：TP309 **文献标识码**：A **文章编号**：1000-436X(2014)02-0040-06

# Preimage and pseudo-collision attacks on 29-step SM3 hash function with padding

WANG Gao-li[1,2], SHEN Yan-zhao[1]
(1. School of Computer Science and Technology, Donghua University, Shanghai 201620, China;
2. State Key Laboratory of Information Security , Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China)

**Abstract:** The security of SM3 hash function was revaluated by using the meet-in-the-middle attack. The preimage and pseudo-collision attack on 29-step SM3 hash function (from the 1-st step) with padding was presented. The time complexities are $2^{254}$ and $2^{125}$ respectively. Therefore, the 29-step SM3 hash function is not immune to preimage and pseudo-collision attack.

**Key words:** hash function; preimage attack; collision attack; meet-in-the-middle attack; SM3

## 1 引言

杂凑函数在密码学中具有重要的地位，安全的杂凑函数能够抵抗碰撞攻击、原根攻击和第二原根攻击等。对于杂凑函数的攻击方法有很多，例如基于模差分的碰撞攻击[1,2]方法、基于中间相遇攻击[3]的原根攻击[4]方法等。对于杂凑函数的攻击中最为显著的是王小云等学者提出的对于 MD5 和 SHA-1 等国际通用杂凑函数的碰撞攻击[1,2]。随着 MD5 和 SHA-1 被破解[1,2]，美国国家标准与技术研究所(NIST)于 2007 年启动了新的杂凑函数标准 SHA-3 的征集活动[5]，并于 2012 年 10 月公布了新一代杂凑函数标准——Keccak 算法。

随着 SHA-3 征集活动的进行，各国都在制定自己的杂凑算法标准。2010 年中国国家密码管理局公布了中国商用密码杂凑算法标准——SM3 密码杂凑函数[6]。该算法由王小云等学者设计，消息分组长度为 512 bit，输出杂凑值长度为 256 bit，采用 Merkle-Damgard 结构。就压缩函数而言，SM3 密码杂凑函数与 SHA-256 具有相似的结构，但是 SM3 算法的压缩函数的每一步都使用 2 个消息字，每一步的扩散能力更强。

**收稿日期**：2012-11-29；**修回日期**：2013-01-31
**基金项目**：国家自然科学基金资助项目（61103238，61373142）；中央高校基本科研业务费专项基金资助项目；中国科学院信息工程研究所信息安全国家重点实验室开放课题基金资助项目
**Foundation Items:** The National Natural Science Foundation of China (61103238, 61373142); The Fundamental Research Funds for the Central Universities; The Opening Project of State Key Laboratory of Information Security of Institute of Information Engineering of Chinese Academy of Sciences

文献[7]首次给出了对于缩减频数的 SM3 算法的原根攻击。其中，对于从第 1 步开始的 28 步 SM3 算法的原根攻击的时间复杂度为 $2^{241.5}$；对于从第 7 步开始的 30 步 SM3 算法的原根攻击的时间复杂度为 $2^{249}$。因为前 16 步 SM3 算法中的函数 $FF(\cdot)$和 $GG(\cdot)$均为异或函数，其性质不利于分析的进行，所以文献[7]用到了第 16 步以后的函数 $FF(\cdot)$和 $GG(\cdot)$的性质，因此，对于 30 步 SM3 算法的原根攻击不能从第 1 步开始。文献[8]给出了对于 35 步 SM3 算法的飞来去器攻击，其时间复杂度为 $2^{117.1}$，证明了从第 1 步开始的 35 步 SM3 算法是非随机的。文献[9]使用差分—中间相遇攻击给出了对 29 步、30 步、31 步和 32 步 SM3 算法的原根攻击和伪碰撞攻击，均从第 1 步开始。但是，文献[9]在构造差分特征时用到了消息填充所需的消息字，因此文献[9]的攻击不能包含有效的消息填充。

在深入分析 SM3 算法的特点后，本文提出了一种带消息填充的、从第 1 步开始的 29 步 SM3 算法的原根攻击和伪碰撞攻击方法。本文采用中间相遇技术进行原根攻击，且攻击过程不依赖于 SM3 算法中函数 $FF(\cdot)$和 $GG(\cdot)$的性质，另外，对于 29 步 SM3 算法的原根攻击可以转化为对于 29 步 SM3 算法的伪碰撞攻击。带消息填充的、从第 1 步开始的 29 步 SM3 的原根攻击和伪碰撞攻击的时间复杂度分别为 $2^{254}$ 和 $2^{125}$。

## 2　SM3 算法

SM3 算法将长度为 $l(l<2^{64})$的消息 $M$(比特串)进行压缩，输出 256 bit 的杂凑值。对于消息 $M$，SM3 算法首先执行一个消息填充过程，在 $M$ 后添加 1 个“1”和 $k$ 个“0”，再添加上 64 bit $l$ 的二进制表示，$k$ 为等式 $l+k+65\equiv 448 \bmod 512$ 的最小正整数解。这样把 $M$ 填充成长度为 512 倍数的新消息 $M'$。接着将 $M'$按照 512 bit 进行分组，假定 $M'=(B^{(0)}, B^{(1)},\cdots,B^{(n-1)})$，$CF(\cdot)$为 SM3 算法的压缩函数，则 $M$ 杂凑值的计算过程如下

$$V_i=\begin{cases} IV, & i=0 \\ CF(V_{i-1},B^{(i-1)}), & 1\leqslant i\leqslant n \end{cases}$$

其中，$IV$ 为初始值，$V_n$ 为 $M$ 的杂凑值。SM3 算法的压缩函数 $V_i=CF(V_{i-1},B^{(i-1)})$由消息扩展过程和变量更新过程组成。

### 2.1　消息扩展过程

SM3 算法的消息扩展过程首先将 $B^{(i-1)}$分成 16 个长度为 32 bit 的消息字 $w_i$，$0\leqslant i\leqslant 15$，然后将其扩展成 68 个消息字 $w_i(0\leqslant i\leqslant 67)$和 64 个消息字 $w_i'(0\leqslant i\leqslant 63)$。具体过程如下。

$i$ 从 16~67:

$$temp=P_1(w_{i-16}\oplus w_{i-9}\oplus (w_{i-3}<<<15))$$
$$w_i=temp\oplus (w_{i-13}<<<7)\oplus w_{i-6}$$

$i$ 从 0~63:

$$w_i'=w_i\oplus w_{i+4}$$

其中，$P_1(X)=X\oplus (X<<<15)\oplus (X<<<23)$。

### 2.2　变量更新过程

对于给定的初始值 $IV=(A_0,B_0,C_0,D_0,E_0,F_0,G_0,H_0)$，SM3 算法变量更新的具体过程如下。

对于 $i$ 从 0~63:

$$SS_{1i}=((A_i<<<12)+E_i+(T_i<<<i))<<<7$$
$$SS_{2i}=SS_{1i}\oplus (A_i<<<12)$$
$$TT_{1i}=FF_i(A_i,B_i,C_i)+D_i+SS_{2i}+w_i'$$
$$TT_{2i}=GG_i(E_i,F_i,G_i)+H_i+SS_{1i}+w_i$$
$$A_{i+1}=TT_{1i},\quad B_{i+1}=A_i,$$
$$C_{i+1}=B_i<<<9,\quad D_{i+1}=C_i,$$
$$E_{i+1}=P_0(TT_{2i}),\quad F_{i+1}=E_i,$$
$$G_{i+1}=F_i<<<19,\quad H_{i+1}=G_i。$$

其中，$P_0(X)=X\oplus (X<<<9)\oplus (X<<<17)$。

$FF_i(A_i,B_i,C_i)$和 $GG_i(E_i,F_i,G_i)$的定义如下

$$FF_i(A_i,B_i,C_i)=\begin{cases} A_i\oplus B_i\oplus C_i, & 0\leqslant i\leqslant 15 \\ (A_i\wedge B_i)\vee (A_i\wedge C_i)\vee (B_i\wedge C_i), & 16\leqslant i\leqslant 63 \end{cases}$$

$$GG_i(E_i,F_i,G_i)=\begin{cases} E_i\oplus F_i\oplus G_i, & 0\leqslant i\leqslant 15 \\ (E_i\wedge F_i)\vee (\neg E_i\wedge G_i), & 16\leqslant i\leqslant 63 \end{cases}$$

### 2.3　符号说明

以下对本文所使用的符号进行说明。

1) $(A_i, B_i, C_i, D_i, E_i, F_i, G_i, H_i)$表示第 $i-1$ 步的输出变量值。特别地，$(A_0, B_0, C_0, D_0, E_0, F_0, G_0, H_0)$表示压缩函数的初始值。

2) $p_i[a\sim b]$表示 32 bit 字 $p_i$ 的第 $a$ 比特到第 $b$ 比特的值，$p_i$ 可以为 $A_i$，$B_i$，$C_i$，$D_i$，$E_i$，$F_i$，$G_i$，$H_i$，$w_i$ 或 $w_i'$，$0\leqslant a\leqslant b\leqslant 31$，第 0 比特为最低位，第 31 比特为最高位。

3) $h=(A, B, C, D, E, F, G, H)$表示 256 bit 的杂凑值。

4) $p_i[a\sim b, c\sim d]$表示 32 比特字 $p_i$ 的第 $a$ 比特到

第 $b$ 比特和第 $c$ 比特到第 $d$ 比特，$p_i$ 可以为 $A$，$B$，$C$，$D$，$E$，$F$，$G$，$H$，$A_i$，$B_i$，$C_i$，$D_i$，$E_i$，$F_i$，$G_i$，$H_i$，$w_i$ 或 $w_i'$，$0 \leqslant a \leqslant b \leqslant c \leqslant d \leqslant 31$。

## 3　相关技术介绍

### 3.1　中间相遇攻击

中间相遇攻击由文献[3]提出，文献[4]第一次将中间相遇攻击用于原根攻击，图 1 展示了文献[9]中的中间相遇攻击过程，其具体描述如下。

1) 选择中立的消息字 $w_f$ 和 $w_b$，将压缩函数分为 $c_f$ 和 $c_b$ 两部分，使 $c_f$ 不依赖于 $w_b$, $c_b$ 不依赖于 $w_f$。

2) 随机选择分割点处的中间变量值，填充除 $w_f$ 和 $w_b$ 之外的消息字。

3) 对于所有的 $w_f$，向后计算至相遇点，保存相遇点的变量值，记为 $L_f$。类似地，对于所有的 $w_b$，向前计算至相遇点，保存相遇点的变量值，记为 $L_b$。

4) 重复步骤 2)和步骤 3)，直到一个完全匹配产生为止。

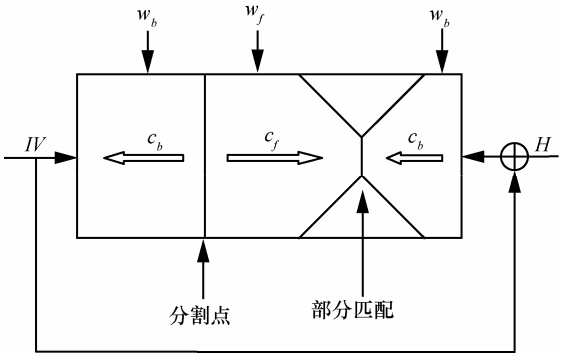



图 1　中间相遇攻击示意

### 3.2　伪原根转化为原根

给定压缩值 $h$，若能找到$(IV',M')$，使得在初始值 $IV'$下压缩 $M'$得到压缩值 $h$，则称 $M'$为 $h$ 的伪原根，其中，$IV'$不等于标准初始值 $IV$。文献[10]提出一种将伪原根攻击转化成原根攻击的方法，其基本过程如下：假设杂凑值的长度为 $n$，伪原根攻击的复杂度为 $2^k$，一方面，在标准初始值 $IV$ 下压缩 $2^{(n+k)/2}$ 个随机的消息，另一方面，计算出压缩值 $h$ 的 $2^{(n-k)/2}$ 个伪原根，根据生日攻击可知，中间相遇的期望值的个数是 $2^{(n+k)/2}\times 2^{(n-k)/2}/2^n=1$，从而找到原根的计算复杂度为 $2^{(n+k)/2}+2^{(n-k)/2}\times 2^k=2^{(n+k)/2+1}$ 次杂凑运算。

### 3.3　伪原根转化为伪碰撞

记杂凑函数为 $H$，如果能找到$(IV, IV', M, M')$，使得 $H(IV, M)=H(IV', M')$，其中，$(IV, M)\neq(IV', M')$，则称找到了杂凑函数 $H$ 的伪碰撞。伪原根转化成伪碰撞[11]的方法可以概括为：对于一个输出长度为 $n$ 的杂凑函数，假定存在一个复杂度为 $2^k$、部分匹配为 $t$ bit 的伪原根攻击，并且部分匹配位于压缩函数的最后一步的输出。对于给定的杂凑值，使用伪原根攻击的方法找到 $2^{(n-t)/2}$ 个不同的 $t$ bit 部分匹配消息，则根据生日攻击的原理，这些消息以很高的概率存在一个伪碰撞。此时，得到一个伪碰撞所需的复杂度为 $2^{(n-t)/2}\times 2^k$。例如，令部分匹配长度 $t$=6，向前向后计算的过程各进行 $2^5$ 次，这样可以得到 $2^4(=2^{5+5}/2^6)$个 6 bit 的部分匹配，也就是说，得到一个 6 bit 的部分匹配所需的复杂度为 $2(=2^5/2^4)$，从而得到一个伪碰撞所需的复杂度为 $2^{(n-6)/2}\times 2$。

## 4　29 步 SM3 算法的原根攻击和伪碰撞攻击

对于从第 1 步开始的 29 步 SM3 算法进行原根攻击时，向后的计算过程从第 15 步到第 29 步，向前的计算过程从第 14 步到第 1 步，这种分割方法使得中间相遇攻击中的部分匹配能够在第 29 步进行，从而保证伪碰撞攻击顺利进行，也为中间消息字的选取提供参考。中立的消息字为 $w_b=w_{13}$[26~31]，$w_f= w_2$[26~31]，这样选择中立消息字的优点在于使抵消中立消息字的条件相对较少，同时这些条件不影响消息的填充过程。其中，在向前的计算过程中，$w_f=w_2$[26~31]未知；在向后的计算过程中，$w_b=w_{13}$[26~31]未知。

### 4.1　设置消息字

根据消息拓展过程可知，$w_{16}$~$w_{29}$ 的拓展过程如下

$$w_{16}=P_1(w_0 \oplus w_7 \oplus (w_{13}\lll 15)) \oplus (w_3\lll 7) \oplus w_{10},$$
$$w_{17}=P_1(w_1 \oplus w_8 \oplus (w_{14}\lll 15)) \oplus (w_4\lll 7) \oplus w_{11},$$
$$w_{18}=P_1(w_2 \oplus w_9 \oplus (w_{15}\lll 15)) \oplus (w_5\lll 7) \oplus w_{12},$$
$$w_{19}=P_1(w_3 \oplus w_{10} \oplus (w_{16}\lll 15)) \oplus (w_6\lll 7) \oplus w_{13},$$
$$w_{20}=P_1(w_4 \oplus w_{11} \oplus (w_{17}\lll 15)) \oplus (w_7\lll 7) \oplus w_{14},$$
$$w_{21}=P_1(w_5 \oplus w_{12} \oplus (w_{18}\lll 15)) \oplus (w_8\lll 7) \oplus w_{15},$$
$$w_{22}=P_1(w_6 \oplus w_{13} \oplus (w_{19}\lll 15)) \oplus (w_9\lll 7) \oplus w_{16},$$
$$w_{23}=P_1(w_7 \oplus w_{14} \oplus (w_{20}\lll 15)) \oplus (w_{10}\lll 7) \oplus w_{17},$$
$$w_{24}=P_1(w_8 \oplus w_{15} \oplus (w_{21}\lll 15)) \oplus (w_{11}\lll 7) \oplus w_{18},$$
$$w_{25}=P_1(w_9 \oplus w_{16} \oplus (w_{22}\lll 15)) \oplus (w_{12}\lll 7) \oplus w_{19},$$
$$w_{26}=P_1(w_{10} \oplus w_{17} \oplus (w_{23}\lll 15)) \oplus (w_{13}\lll 7) \oplus w_{20},$$
$$w_{27}=P_1(w_{11} \oplus w_{18} \oplus (w_{24}\lll 15)) \oplus (w_{14}\lll 7) \oplus w_{21},$$
$$w_{28}=P_1(w_{12} \oplus w_{19} \oplus (w_{25}\lll 15)) \oplus (w_{15}\lll 7) \oplus w_{22},$$
$$w_{29}=P_1(w_{13} \oplus w_{20} \oplus (w_{26}\lll 15)) \oplus (w_{16}\lll 7) \oplus w_{23}。$$

由上述拓展过程可知，1) 从 $w_{16}$~$w_{29}$ 首先受到 $w_2$ 影响的消息字为 $w_{18}$，另一方面，在向前的计算过程中，从第 14 步到第 4 步不受 $w_2$ 的影响；2) 受 $w_{13}$ 影响的消息字为 $w_{16}$、$w_{19}$、$w_{22}$、$w_{26}$ 和 $w_{29}$($w_{16}$~$w_{29}$)，可以通过以下的条件来抵消 $w_{13}$ 对 $w_{16}$、$w_{19}$、$w_{22}$ 和 $w_{26}$ 的影响。

抵消 $w_{13}$ 对 $w_{26}$ 的影响。

由于 $w_{26}=P_1(w_{10}\oplus w_{17}\oplus(w_{23}<<<15))\oplus(w_{13}<<<7)\oplus w_{20}$，首先用 $w_{10}$ 抵消 $w_{13}$ 对 $w_{26}$ 的影响，需要增加条件

$$P_1(w_{10})\oplus(w_{13}<<<7)=C_1 \tag{1}$$

由条件(1)可知 $w_{10}$ 受到 $w_{13}$ 的影响，又根据 $w_{23}$ 的表达式可知，$w_{23}$ 受到 $w_{13}$ 的影响，笔者用条件(2)抵消 $w_{13}$ 对 $w_{23}$ 的影响，即

$$P_1(w_7)\oplus(w_{10}<<<7)=C_2 \tag{2}$$

类似地，根据 $w_{20}$ 和 $w_{17}$ 的表达式可知：$w_{20}$ 和 $w_{17}$ 受到 $w_{13}$ 的影响，笔者用条件(3)和条件(4)抵消 $w_{20}$ 和 $w_{17}$ 受到的影响，即

$$P_1(w_4)\oplus(w_7<<<7)=C_3 \tag{3}$$
$$P_1(w_1)\oplus(w_4<<<7)=C_4 \tag{4}$$

综上可知：$w_{14}$~$w_{28}$ 中除 $w_{16}$、$w_{19}$ 和 $w_{22}$ 之外，其他消息字均不受 $w_{13}$ 的影响。

抵消 $w_{13}$ 对 $w_{22}$、$w_{19}$ 和 $w_{16}$ 的影响。根据 $w_{22}=P_1(w_6\oplus w_{13}\oplus(w_{19}<<<15))\oplus(w_9<<<7)\oplus w_{16}$，$w_{19}=P_1(w_3\oplus w_{10}\oplus(w_{16}<<<15))\oplus(w_6<<<7)\oplus w_{13}$，$w_{16}=P_1(w_0\oplus w_7\oplus(w_{13}<<<15))\oplus(w_3<<<7)\oplus w_{10}$，首先用条件(5)抵消 $w_{13}$ 对 $w_{22}$ 的影响。

$$w_6\oplus w_{13}=C_5 \tag{5}$$

然后用条件(6)抵消 $w_{13}$ 对 $w_{19}$ 的影响。

$$P_1(w_3\oplus w_{10})\oplus(w_6<<<7)\oplus w_{13}=C_6 \tag{6}$$

由前面的分析可知：$w_{16}$ 表达式中的 $w_3$、$w_7$ 和 $w_{10}$ 均受到 $w_{13}$ 的影响，所以用条件(7)抵消 $w_{13}$ 对 $w_{16}$ 的影响。

$$P_1(w_0\oplus w_7\oplus(w_{13}<<<15))\oplus(w_3<<<7)\oplus w_{10}=C_7 \tag{7}$$

综上可知：通过条件(1)到条件(7)对 $w_0$、$w_1$、$w_3$、$w_4$、$w_6$、$w_7$ 和 $w_{10}$ 进行了设定，使得 $w_{14}$~$w_{28}$ 均不受 $w_{13}$ 的影响。因此，在向后的计算过程中，从第 15 步到第 25 步不受 $w_{13}$ 的影响。其中，$C_1$，…，$C_7$ 为常数。

### 4.2　第 14 步到第 1 步的计算

因为在向前的计算过程中，$w_2$[26~31]未知，根据消息扩展算法可知，$w_{16}$ 和 $w_{17}$ 已知，$w_{18}$ 未知，从而 $w_i$ 和 $w_i'$已知($i$=3,…,13），故从分割处的中间变量值($A_{14}$, $B_{14}$, $C_{14}$, $D_{14}$, $E_{14}$, $F_{14}$, $G_{14}$, $H_{14}$)向前逆向计算，可以很容易得到($A_3$, $B_3$, $C_3$, $D_3$, $E_3$, $F_3$, $G_3$, $H_3$)。由此再向前的计算受 $w_2$ 的影响，其过程如下。首先考虑

$$(A_3, B_3, C_3, D_3, E_3, F_3, G_3, H_3)\rightarrow (A_2, B_2, C_2, D_2, E_2, F_2, G_2, H_2)$$

由于 $w_2$ 参与此步的运算，且 $w_2$[26~31]未知，故可以计算出 $A_2$，$B_2$，$C_2$，$D_2$[0~25]，$E_2$，$F_2$，$G_2$ 和 $H_2$[0~25]。然后考虑

$$(A_2, B_2, C_2, D_2, E_2, F_2, G_2, H_2)\rightarrow (A_1, B_1, C_1, D_1, E_1, F_1, G_1, H_1),$$

根据 SM3 算法，易知：$A_1$、$B_1$、$C_1$[0~25]、$D_1$[0~25]、$E_1$、$F_1$、$G_1$[0~25]和 $H_1$[0~25]能够计算出来。最后考虑

$$(A_1, B_1, C_1, D_1, E_1, F_1, G_1, H_1)\rightarrow (A_0, B_0, C_0, D_0, E_0, F_0, G_0, H_0)$$

根据 SM3 算法，易知：$A_0$、$B_0$[0~16, 23~31]、$C_0$[0~25]、$D_0$[0~16]、$E_0$、$F_0$[0~6,13~31]、$G_0$[0~25]和 $H_0$[0~6]能够计算出来。

### 4.3　第 15 步到第 29 步的计算

因为在向后的计算过程中，$w_{13}$[26~31]未知，根据 4.1 节消息字的设置可得，$w_i$ 和 $w_i'$已知($i$=14,…,24)，故从分割处的中间变量值($A_{14}$, $B_{14}$, $C_{14}$, $D_{14}$, $E_{14}$, $F_{14}$, $G_{14}$, $H_{14}$)向后计算，很容易得到($A_{25}$, $B_{25}$, $C_{25}$, $D_{25}$, $E_{25}$, $F_{25}$, $G_{25}$, $H_{25}$)的值。

因为参与第 26 步计算的消息字为 $w_{25}$ 和 $w_{25}'=w_{25}\oplus w_{29}$，而 $w_{29}$ 受 $w_{13}$ 的影响，所以 $A_{26}$ 受 $w_{13}$ 的影响，而($B_{26}$, $C_{26}$, $D_{26}$, $E_{26}$, $F_{26}$, $G_{26}$, $H_{26}$)不受 $w_{13}$ 的影响。特别地，$E_{26}$ 不受 $w_{13}$ 的影响，亦即可以得到 $E_{26}$ 的值，从而根据 SM3 算法可知：$H_{29}=G_{28}=F_{27}<<<19=E_{26}<<<19$ 可以计算出来。

综合第 14 步到第 1 步的计算过程和第 14 步到第 29 步的计算过程，可以得到部分匹配的检测条件为 $H_0[0\sim6]\oplus H_{29}[0\sim6]=H[0\sim6]$是否成立。

### 4.4　29 步 SM3 算法的原根攻击

通过以上的分析可知，为了保证 $w_{13}$ 对 $w_{16}$、$w_{19}$、$w_{22}$ 和 $w_{26}$ 不产生影响，需要限定 7 个消息字 $w_0$、$w_1$、$w_3$、$w_4$、$w_6$、$w_7$ 和 $w_{10}$ 的条件，一共 224(=7×32) bit。对于一个消息分组来说，消息字剩余的自由度为 276(=512–224–2×6)，进一步考虑消息填充过程中用到的 $w_{13}$ 的第 0 位，$w_{14}$ 和 $w_{15}$ 共 65 bit，最终剩

余的消息字的自由度为 211(=276–65)。另一方面，中间变量($A_{14}$, $B_{14}$, $C_{14}$, $D_{14}$, $E_{14}$, $F_{14}$, $G_{14}$, $H_{14}$)没有任何条件限制，因此可以增加 256 个自由度，从而可用的自由度为 467(=211+256)>256。显然有足够的自由度来构造出一个有效的对 29 步 SM3 算法的伪原根攻击。对于给定的杂凑值 $h$=($A$, $B$, $C$, $D$, $E$, $F$, $G$, $H$)，29 步的 SM3 算法的伪原根攻击过程如下。

1) 随机选取中间变量($A_{14}$, $B_{14}$, $C_{14}$, $D_{14}$, $E_{14}$, $F_{14}$, $G_{14}$, $H_{14}$)。对于消息字 $w_0$~$w_{15}$，随机赋值给除 $w_2$[26~31]和 $w_{13}$[26~31]之外的其他消息字，使得 4.1 节的条件(1)~条件(7)满足，且消息填充满足。

2) 对于所有的 $w_{13}$[26~31]，向前计算得($A_3$, $B_3$, $C_3$, $D_3$, $E_3$, $F_3$, $G_3$, $H_3$)。用($w_2 \wedge$ 03ffffff)代替 $w_2$ 向前计算得 ($A_0$, $B_0$, $C_0$, $D_0$, $E_0$, $F_0$, $G_0$, $H_0$)。保存($A_3$, $B_3$, $C_3$, $D_3$, $E_3$, $F_3$, $G_3$, $H_3$, $H_0$, $w_{13}$[26~31])到列表 $L_b$ 中。

3) 对于所有的 $w_2$[26~31]，向后计算得 ($A_{25}$, $B_{25}$, $C_{25}$, $D_{25}$, $E_{25}$, $F_{25}$, $G_{25}$, $H_{25}$)。在第 26 步，计算 $E_{26}$ 的值。保存($A_{25}$, $B_{25}$, $C_{25}$, $D_{25}$, $E_{25}$, $F_{25}$, $G_{25}$, $H_{25}$, $E_{26}$<<<19, $w_2$[26~31])到列表 $L_f$ 中。

4) 判断 $H_0$[0~6] $\oplus$ $H_{29}$[0~6]是否等于 $H$[0~6]。若相等，则使用相应的($A_3$, $B_3$, $C_3$, $D_3$, $E_3$, $F_3$, $G_3$, $H_3$)和 $w_2$[26~31]向前计算得($A_0$, $B_0$, $C_0$, $D_0$, $E_0$, $F_0$, $G_0$, $H_0$)；使用相应的($A_{25}$, $B_{25}$, $C_{25}$, $D_{25}$, $E_{25}$, $F_{25}$, $G_{25}$, $H_{25}$)和 $w_{13}$[26~31]向后计算得($A_{29}$,$B_{29}$, $C_{29}$, $D_{29}$, $E_{29}$, $F_{29}$, $G_{29}$, $H_{29}$)。

5) 判断($A_0$, $B_0$, $C_0$, $D_0$, $E_0$, $F_0$, $G_0$, $H_0$) $\oplus$ ($A_{29}$, $B_{29}$, $C_{29}$, $D_{29}$, $E_{29}$, $F_{29}$, $G_{29}$, $H_{29}$)是否等于 $h$。若相等，则相应的消息分组($w_0$,…,$w_{15}$)即为 29 步 SM3 算法的一个伪原根。若不相等，则重复步骤 1)~步骤 4)，直到找到一个 29 步 SM3 的伪原根。

上述攻击的时间复杂度估计如下：由步骤 2)和步骤 3)可知，共有 $2^6\times2^6=2^{12}$ 个组合，而步骤 4)“中间相遇”的概率是 $2^{-7}$，故做一次步骤 2)和步骤 3)可以得到 $2^{12-7}=2^5$ 个“中间相遇”，而做一次步骤 2)和步骤 3)的复杂度约为 $2^6$ 次 29 步 SM3 算法压缩操作，故得到 7 bit“中间相遇”值的复杂度是 $2^6/2^5=2$。从而 29 步 SM3 算法的伪原根攻击的复杂度为 $2^{256-7}\times2=2^{250}$。

使用伪原根攻击转化原根攻击的方法(见 3.2 节)，可以得到对 29 步 SM3 算法的原根攻击，其时间复杂度为 $2^{254}$($=2^{256+250)/2+1}$)。

### 4.5 29 步 SM3 算法的伪碰撞攻击

根据伪原根攻击转化为伪碰撞攻击的方法(见 3.3 节)，为了提高效率，把伪原根攻击中部分匹配的判断条件改为 $H_0$[0~5] $\oplus$ $H_{29}$[0~5]是否等于 $H$[0~5]，可以得到 29 步 SM3 算法的伪碰撞攻击，其时间复杂度为 $2^{125}=2^{(256-6)/2}\times2^0$，具体 SM3 算法的攻击结果如表 1 所示。

**表 1 SM3 算法的攻击结果**

| 攻击类型 | 步数 | 是否填充 | 复杂度 | 起始步数 | 文献 |
|---|---|---|---|---|---|
| 原根攻击 | 28 | 是 | $2^{241.5}$ | 1 | [7] |
| | 30 | 是 | $2^{249}$ | 7 | [7] |
| | 29 | 否 | $2^{245}$ | 1 | [9] |
| | 30 | 否 | $2^{251.1}$ | 1 | [9] |
| | 29 | 是 | $2^{254}$ | 1 | 本文 |
| 伪碰撞攻击 | 29 | 否 | $2^{122}$ | 1 | [9] |
| | 30 | 否 | $2^{125.1}$ | 1 | [9] |
| | 31 | 否 | $2^{122}$ | 1 | [9] |
| | 32 | 否 | $2^{125.1}$ | 1 | [9] |
| | 29 | 是 | $2^{125}$ | 1 | 本文 |
| 飞来去器攻击 | 32 | 否 | $2^{14.4}$ | 1 | [8] |
| | 33 | 否 | $2^{32.4}$ | 1 | [8] |
| | 34 | 否 | $2^{53.1}$ | 1 | [8] |
| | 35 | 否 | $2^{117.1}$ | 1 | [8] |

## 5 结束语

本文给出了 29 步 SM3 算法的原根攻击，并把它转化为对 29 步 SM3 算法的伪碰撞攻击。与已知的攻击结果相比，本文的攻击是从第 1 步开始的，且满足消息填充。表 1 给出了 SM3 算法的攻击结果比较。虽然从第 1 步开始的 29 步 SM3 算法不能有效地抵抗原根攻击和伪碰撞攻击，但是由于 SM3 算法的快速扩散能力，完整的 SM3 算法仍具有抵抗各种已知攻击的能力，具有非常高的安全性。如何发现并利用 SM3 的其他特点来分析更多步数的算法是下一步的研究重点。

**参考文献：**

[1] WANG X, YU H. How to break MD5 and other hash functions[A]. EUROCRYPT 2005[C]. Aarhus, Denmark, 2005. 19-35.

[2] WANG X, YIN Y, YU H. Finding collisions in the full SHA-1[A]. CRYPTO 2005[C]. Santa Barbara, California, USA, 2005. 17-36.

[3] DIFFIE W, HELLMAN M. Exhaustive cryptanalysis of the NBS data encryption standard[J]. Computer, 1977,10(6):74-84.

[4] AOKI K, SASAKI Y. Preimage attacks on one-block MD4, 63-step MD5 and more[A]. SAC 2008[C]. New Brunswick, Canada, 2008. 103-119.

[5] National Institute of Standards and Technology. Announcing request for candidate algorithm nominations for a new cryptographic hash algorithm (SHA-3) family[EB/OL]. http://csrc.nist.gov/ groups/ST/hash/ documents/ FR_ Notice_Nov07.pdf, 2008.

[6] SM3 hash function[EB/OL]. http://www.oscca.gov.cn/UpFile/2010 1222141857786.pdf/, 2010.

[7] ZOU J, WU W, WU S, *et al*. Preimage attacks on step-reduced SM3 hash function[A]. ICISC 2011[C]. Seoul Korea, 2011.375-390.

[8] KIRCANSKI A, SHEN Y, WANG G, *et al*. Boomerang and slide-rotational analysis of SM3 hash function[A]. SAC 2012[C]. Windsor, Canada, 2012.305-321.

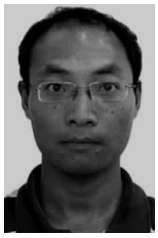
[9] WANG G, SHEN Y. Preimage and pseudo-collision attacks on step-reduced SM3 hash function[EB/OL]. http://eprint.iacr.org/2012/640/, 2012.

[10] MENEZES A J, OORSCHOT P C, VANSTONE S. Handbook of Applied Cryptography[M]. CRC Press,1996.

[11] LI J, ISOBE T, SHIBUTANI K. Converting meetin-the-middle preimage attack into pseudo collision attack: application to SHA-2[A]. FSE 2012[C]. Washington DC, USA, 2012.264-286.

**作者简介：**

**王高丽**（1982-），女，安徽宿州人，博士，东华大学副教授、硕士生导师，主要研究方向为密码学、对称密码算法的分析与设计。



**申延召**（1984-），男，河南汝州人，东华大学硕士生，主要研究方向为散列函数设计与分析。

（上接第 39 页）

[23] YU J, KONG F Y, CHENG X G, *et al*. Forward-secure identity-based public-key encryption without random oracles[J]. Fundamenta Informaticae, 2011, 111(2): 241-256.

[24] 杨浩淼, 孙世新, 李洪伟. 前向安全的基于身份加密方案[J]. 电子科技大学学报, 2007, 36(3):534-537.
YANG H M, SUN S X, LI H W. Forward-secure identity-based encryption scheme[J]. Journal of University of Electronic Science and Technology of China, 2007, 36(3):534-537.

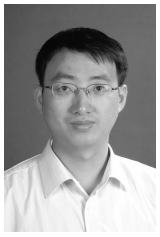
[25] LU Y, LI J G. Forward-secure certificate-based encryption and its generic construction[J]. Journal of Networks, 2010, 5(5):527-534.

[26] RACKOFF C, SIMON D. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack[A]. Proc of the Crypto 1991[C]. Heidelberg:Springer-Verlag, 1991. 433-444.

[27] GENTRY C. Practical identity-based encryption without random oracles[A]. Proc of the Eurocrypt 2006[C]. Heidelberg: Springer-Verlag, 2006. 445-464.

[28] LIU J K, ZHOU J Y. Efficient certificate-based encryption in the standard model[A]. Proc of the 6th International Conference on Security and Cryptography for Networks[C]. Heidelberg:Springer-Verlag, 2008. 144-155.

**作者简介：**



**陆阳**（1977-），男，江苏扬州人，博士，河海大学副教授、硕士生导师，主要研究方向为信息安全、密码学理论与技术。

**李继国**（1970-），男，黑龙江富裕人，博士，河海大学教授、博士生导师，主要研究方向为信息安全、密码学理论与技术。