软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2022,33(12):4784–4803 [doi: 10.13328/j.cnki.jos.006378]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

SHA-1 差分路径搜索算法和连接策略研究*

曾 光[1,3], 李婧瑜[1], 杨 阳[1,2]

[1](数学工程与先进计算国家重点实验室(中国人民解放军战略支援部队 信息工程大学), 河南 郑州 450001)

[2](中国科学院 软件研究所 可信计算与信息保障实验室, 北京 100190)

[3](华为技术有限公司, 北京 100195)

通信作者: 杨阳, E-mail: yangyang_wawa@126.com

摘 要: Hash 函数 SHA-1 的攻击技术研究一直受到密码分析者的广泛关注, 其中, 差分路径构造是影响攻击复杂度大小的重要环节. 提出了带比特条件的全轮差分路径构造方法, 统一了第 1 轮差分路径构造和后 3 轮的差分路径构造. 该方法既与原有第 1 轮路径构造相容, 又能省去后 3 轮路径约简、消息约简等繁琐技术环节, 具有良好的兼容性. 此外, 综合考虑状态差分、布尔函数差分与比特条件之间的制约关系, 提出了带比特条件的前向扩展、后向扩展和中间连接这 3 个子算法, 并提出 3 个指标——比特条件的更新次数、扩展结果的相容性和候选集合的正确率对中间连接的成功率进行评价, 结合提前终止策略, 提出了最优的中间连接算法. 理论分析结果表明, 该方法有助于提高 SHA-1 差分路径构造的成功率. 最后, 采用该算法进行路径搜索, 可以得到正确的可用于碰撞搜索的差分路径.

关键词: 密码学; Hash 函数; SHA-1 算法; 差分路径

中图法分类号: TP309

中文引用格式: 曾光, 李婧瑜, 杨阳. SHA-1 差分路径搜索算法和连接策略研究. 软件学报, 2022, 33(12): 4784–4803. http://www.jos.org.cn/1000-9825/6378.htm

英文引用格式: Zeng G, Li JY, Yang Y. Research on SHA-1 Differential Path Search Algorithms and Connect Strategies. Ruan Jian Xue Bao/Journal of Software, 2022, 33(12): 4784–4803 (in Chinese). http://www.jos.org.cn/1000-9825/6378.htm

Research on SHA-1 Differential Path Search Algorithms and Connect Strategies

ZENG Guang[1,3], LI Jing-Yu[1], YANG Yang[1,2]

[1](State Key Laboratory of Mathematical Engineering and Advanced Computing (PLA Strategic Force Information Engineering University), Zhengzhou 450001, China)

[2](Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

[3](Huawei Technology Co. Ltd., Beijing 100095, China)

Abstract: As one of the most widely used Hash functions, the research on related attack techniques on SHA-1 algorithm has been widely concerned by cryptographers since it was put forward. In the collision attack against SHA-1, the construction of the differential path is an important step that affects the complexity of the attack. This study proposes the concept of a differential path with bitconditions and its construction method. The path comprehensively describes the Boolean function difference, bitcondition, message difference, and working state difference of each step. It is not only compatible with the original first round path construction, but also can save the complicated technologies such as path reduction and message reduction of the last three rounds. Therefore, the differential path with bitconditions has good compatibility. In addition, before proposing a corresponding construction algorithm for the differential path with bitconditions, this study firstly analyzes the value of the output Boolean function difference and its input bitconditions when the three input working state differences are fixed. That is the foundation for the later work. The differential path construction algorithm is divided into three

* 基金项目: 国家自然科学基金(61972413); 国家重点研发计划(2017YFB0803203); 数学工程与先进计算国家重点实验室开放基金(2020A08)

收稿时间: 2020-10-29; 修改时间: 2021-01-10; 采用时间: 2021-05-13

sub-algorithms of forward expansion, backward expansion, and connect algorithm. The forward expansion and backward expansion mainly consider the relationship between the bitcondition on the working state and the output difference of the Boolean function. The forward of each step is based on the expansion result of the previous step, and so is the backward algorithm. The goal of the connect algorithm is to connect the results of forward expansion and backward expansion to form a complete and valid differential path. Whether the connect algorithm is successful determines whether the collision attack can be continued. If the connect algorithm fails, it is necessary to renew the forward expansion and backward expansion. In order to improve the success rate of connection algorithm, this study proposes three related indexes of update times to bitconditions, compatibility of extension results, and accuracy of candidate sets. Analyzing these three indexes, the relationship between the success rate and them is achieved. The number of forward expansions is proportional to the update times to bitconditions. As for the compatibility of extension results, the times to judge the consistency between the expansion result and original conditions is inversely proportional to the success rate. These two indexes are affected by the number of forward or backward expansions. The accuracy of candidate sets refers to the correct rate of the selectable of working state difference, Boolean function difference and message difference. Analyses finds that the order of the expansions can affect this accuracy. The accuracy of two expansions in opposite directions is higher than that in the same direction. So forward expansions and backward expansions are executed alternately at the first four expansions of the connect algorithm in this study. According to these conclusions, the optimal connect algorithm is selected from 25 possible algorithms. Combining early-abort strategy, the connect algorithm proposed in this study has higher success rate and lower complexity. Theoretical analysis results show that this method helps to improve the success rate of SHA-1 differential path construction. At last, valid differential paths which are used for collision search can be constructed using the algorithm in this study.

Key words: cryptography; Hash function; SHA-1 algorithms; differential path

作为现代密码学的基本组件之一, Hash 函数在密码协议、消息认证、数字证书和网络协议等方面有着重要应用价值. MD (merkle-damgård)结构是 Hash 函数中最常见的, 采取迭代计算的方法, 每次压缩的输入为固定长度的消息和上一次的压缩结果, 输出为固定长度的消息摘要. SHA-1 算法是典型的 MD 结构的 Hash 函数, 由美国国家安全局(NSA)设计，并由美国国家标准技术研究所(NIST)在 1995 年发布为联邦数据处理标准(FIPS) . 自发布以来该算法就被迅速地作为新的安全标准应用于政府、商业、军事等各个领域. 它的广泛应用也引来了众多密码分析者的关注. 近年来, 主流的对 SHA-1 的攻击方法是随机碰撞攻击和选择前缀攻击. 2017 年, Marc Stevens[2,3]以 $2^{63.1}$ 的复杂度给出了 SHA-1 的第 1 个真实的全轮碰撞. 2019 年, Gaëtan 等人[4,5]将选择前缀攻击分为两个阶段——生日攻击和多块消息近似碰撞, 以 $2^{63.4}$ 的复杂度完成了对 SHA-1 的选择前缀攻击, 这是目前对 SHA-1 的选择前缀攻击的最好结果.

随机碰撞攻击一般构造两个近似碰撞块, 第 2 个消息块可以抵消第 1 个消息块产生的差分. 具体过程包括扰动向量的选择、第 1 轮差分路径构造、后 60 步的差分分析、确定第 2 轮比特条件、推导充分条件和寻找消息块等步骤. 扰动向量的选择是进行碰撞攻击的前提. 2011 年, Manuel[6]提出了 EEM 算法, 并指出之前用到的扰动向量只有 Type-I[7,8]和 Type-II[9]这两类. Tang 等人[10]进一步证明了汉明重量为 25 的 60 步扰动向量只有 Type-I 和 Type-II 这两类. 扰动向量后 60 步的汉明重量是影响扰动向量优劣的重要因素, 因此, 为了进一步降低攻击复杂度, 还需寻找其他具有更低汉明重量的扰动向量.

差分路径的构造是影响整体攻击复杂度的重要环节, 其难点在于差分在路径多轮迭代后的难预测和空间爆炸导致计算量过大. 早期的差分路径大多是由手动推导得到的, 手动推导过程中, 可以利用比特扩展对差分作出改变, 也可以利用第 1 轮布尔函数对差分的吸收性质确定布尔函数的输出差分. 但这一方法需要对路径进行很多次推导, 耗时费力. 后来, 差分路径的自动化搜索成为研究热点, 一般分为 3 个子算法: 前向扩展、后向扩展和中间连接. 本文首先对差分路径的形式作出了改进, 提出了带比特条件的差分路径. 带比特条件的差分路径不仅包括每一步路径上的寄存器差分和消息差分, 还包括每一步的比特条件和布尔函数差分, 为差分路径进行了精确刻画. 令人兴奋地是, 差分路径描述变量的增加并没有导致路径构造方法变得复杂, 反而统一了原有第 1 轮和后 3 轮的不同构造方法, 给出的最终构造算法仍然分为 3 个子算法: 前向扩展、后向扩展和中间连接, 与原有的第 1 轮构造方式完成相容.

此外, 本文提出了影响中间连接算法的成功率的 3 个指标——比特条件的更新次数、扩展结果的相容性和候选集合的正确率, 基于此, 给出了中间连接算法的最优策略. 分析结果表明: 前向扩展的次数与比特条

件的更新次数成正比; 原始条件判断相容性的次数与中间连接的成功率成反比; 且两次方向相反的扩展可提高候选正确率. 本文提出的中间连接算法正是基于这些结论得出的最优算法. 经验证: 根据该算法搜索得到的差分路径是正确的, 可用于碰撞消息对的搜索.

1 相关工作

在早期的杂凑函数碰撞攻击过程中, 差分路径大都是由手动推导得到的. 首先, Chabaud 等人[11]在 SHA-0 算法的压缩函数中发现了 6 步的局部碰撞, 用扰动向量来标记 6 步局部碰撞的起始比特位, 然后根据扰动向量对 6 步局部碰撞进行复合, 结果得到了一条线性差分路径. 2004 年, 王小云等人[12]利用模差分攻击技术首次给出了 MD5 随机碰撞消息对, 奠定了 Hash 函数随机碰撞攻击方法中模差分攻击的主流地位. 在此基础上, 2005 年, 王小云等人[7]通过引入多消息修改技术, 再次利用模差分攻击技术把对 SHA-1 随机碰撞攻击的复杂度降到了 2^{69}, 这也是首次将 SHA-1 随机碰撞攻击的复杂度降到生日攻击以下. 随后, 王小云等人[13]同样也给出了复杂度为 2^{63} 的搜索数据. 由于该方法对其他 MD 结构的 Hash 函数也具有较好的攻击效果, 目前该方法已经成为针对 MD 结构 Hash 函数攻击的基本方法. 许多团队尝试利用该方法对 SHA-1 算法进行全轮或是减轮的碰撞攻击. 2006 年, Cannière 等人[14]基于寄存器和消息比特的性质, 通过限定消息空间, 给出了复杂度为 2^{35} 的 64 轮 SHA-1 随机碰撞消息对. 2007 年, Cannière 等人[15]通过进一步研究, 又给出了复杂度为 2^{44} 的 70 轮 SHA-1 随机碰撞消息对. 2011 年, Chen[16]利用中立比特和二阶差分技术, 将全轮攻击的复杂度降为 2^{58},但该方法并没有实际的攻击效果, 仅存在理论价值. 2012 年, Andrew 等人[17]借助 GPU 并行技术搜索得到了 75 轮的 SHA-1 随机碰撞消息对. Marc Stevens[2]给出了新的差分路径构造方法, 并且介绍了一种局部碰撞连接分析的新技术, 可以分析由扰动向量所得到的全部后 3 轮的差分路径. 随后, Marc Stevens[3,18]利用 64 个 GPU, 花费了 10 天的时间, 找到了第 1 个 SHA-1 算法的全轮自由碰撞, 复杂度为 $2^{57.5}$ 次 SHA-1 运算. 2017 年 3 月, Marc Stevens[19]公布了第 1 个 SHA-1 算法全轮的真实碰撞, 攻击复杂度为 $2^{63.1}$ 次 SHA-1 算法,相当于 6 500 个 CPU 年或是 100 个 GPU 年, 这是 SHA-1 算法碰撞攻击研究的一个里程碑.

为了实现差分路径的自动化构造, 很多学者做了相关的研究, 并取得了一定的成果. 2005 年, 王小云团队分别在文献[7,12]中给出了 MD5 和 SHA-1 的一条差分路径. 2007 年, Yajima[20]提出了针对 SHA-1 第 1 轮差分路径的自动化构造算法, 该算法与 MD5 选择前缀碰撞攻击的方法类似, 算法分为 3 部分:向前搜索、向后搜索以及中间连接, 输入王小云给出的扰动向量, 算法最终找到了 383 条不同于王的差分路径. 该算法的主要缺陷在于要求给定第 1 轮之后的差分路径, 故而算法适应性较弱. 2013 年, Marc Stevens[2,3]基于部分碰撞中比特之间的相互关系, 采用同时推导差分路径和充分条件的方法, 提出了一种新的 SHA-1 差分路径构造算法, 其中间连接过程采取了迭代构造的方法, 增加了中间连接过程的成功概率. 但该算法构造方法复杂, 且仍要求给定第 1 轮之后的差分路径. 此后, 针对 SHA-1 算法的差分路径自动化构造技术的研究没有更好的结果.

2 SHA-1 算法

SHA-1 算法采用 MD 迭代结构, 可以将任意长度的输入消息压缩成 160 比特的 Hash 值. 算法分成消息填充、消息扩展和压缩函数这 3 个部分.

- 消息填充

在输入消息的最后先填充一个 1, 再填充若干个 0, 使得填充后的消息长度 l 满足 $l \equiv 448 \bmod 512$, 再将原始消息的长度转换成 64 比特二进制的形式填充到消息末尾, 此时消息长度是 512 的整数倍, 这就是消息填充过程.

- 消息扩展

设填充后的消息长度为 512 比特倍数, 每个 512 比特为一个消息块, 分别记为 $M_0,M_1,\ldots,M_{K-1}$. 消息扩展过程是将每个 512 比特的消息块扩展成 80 个 32 比特的消息字. 每个消息块由 16 个 32 比特的消息字组成, 将第 k+1 个消息块 M_k 记为 $W_{0,k},W_{1,k},\ldots,W_{15,k}$, 则对 M_k 的消息扩展准则如下式:

$$W_{i,k}=\begin{cases}W_{i,k}, & 0\leqslant i<16\\ RL(W_{i-3,k}\oplus W_{i-8,k}\oplus W_{i-14,k}\oplus W_{i-16,k},1), & 16\leqslant i\leqslant 80\end{cases}.$$

接下来, 扩展后的消息块依次进入压缩函数, 当处理第 k+1 个消息块 M_k 时, 压缩函数以 M_k 和上一个消息块的压缩函数输出 $IHV_k=(a_k,b_k,c_k,d_k,e_k)$作为输入($a_k,b_k,c_k,d_k,e_k$ 为 32 比特消息字), 输出结果为 160 比特的消息摘要 IHV_{k+1}, 即 $IHV_{k+1}=Compress(IHV_k,M_k)$. 处理第 1 个消息块 M_0 的初始值取值为 $IHV_0=(a_0,b_0,c_0,d_0,e_0)$.

- a_0=0x67452301;
- b_0=0xefcdab89;
- c_0=0x98badcfe;
- d_0=0x10325476;
- e_0=0xc3d2e1f0.

这样, 最后一个消息块 M_{K-1} 进入压缩函数后的输出结果 IHV_k 即为最终的 SHA-1 哈希值.

- 压缩函数

第 k+1 个消息块 M_k 经过消息扩展后进入压缩函数的过程如下: 由上一步的压缩结果 $IHV_k=(a_k,b_k,c_k,d_k,e_k)$, 令 5 个寄存器的初始值为$(Q_0,Q_{-1},Q_{-2},Q_{-3},Q_{-4})=(a_k,b_k,RR(c_k,30),RR(d_k,30),RR(e_k,30))$.

然后进行 4 轮(每轮 20 步)的迭代计算, 每次迭代均输入一个扩展后的消息字, 第 t($0\leqslant t\leqslant 79$)步的迭代方程为

$$Q_{t+1}=RL(Q_t,5)+F_t+RL(Q_{t-4},30)+W_{t,k}+AC_t,$$

其中,

(1) $F_t=f_t(Q_{t-1},RL(Q_{t-2},30),RL(Q_{t-3},30))$, f_t 是以 3 个比特为输入的布尔函数, 其在每一轮的表达式分别为

$$f_t(X,Y,Z)=\begin{cases}(X\wedge Y)\oplus(\overline{X}\wedge Z), & 0\leqslant t<20\\ X\oplus Y\oplus Z, & 20\leqslant t<40\\ (X\wedge Y)\vee(Z\wedge(X\vee Y)), & 40\leqslant t<60\\ X\oplus Y\oplus Z, & 60\leqslant t<80\end{cases};$$

(2) AC_t 是轮常数, 其在每一轮的取值如下:

$$AC_t=\begin{cases}0x5a827999, & 0\leqslant t<20\\ 0x6ed9eba1, & 20\leqslant t<40\\ 0x8f1bbcdc, & 40\leqslant t<60\\ 0xca62c1d6, & 60\leqslant t<80\end{cases};$$

(3) $RL(Q_t,n)$是指对 32 比特寄存器 Q_t 循环移位 n 位的结果.

80 步迭代之后,压缩函数的输出为

$$IHV_{k+1}=(a_{k+1},b_{k+1},c_{k+1},d_{k+1},e_{k+1})=(a_k+Q_{80},b_k+Q_{79},c_k+RL(Q_{78},30),d_k+RL(Q_{77},30),e_k+RL(Q_{76},30)).$$

3 相关定义

本文的定义与 Marc Stevens 的博士论文中定义一致, 这里只列出重要的符号表示, 其他相关的符号可以参见文献[3].

3.1 比特差分和差分值

设 $X=\sum_{i=0}^{31}x_i2^i\in\mathbb{Z}/(2^{32})$, 满足 $\sum_{i=0}^{31}z_i2^i=X$ 的序列串 $Z=(z_i)_{i=0}^{31}\in\{-1,0,1\}^{32}$ 是 X 的一个有符号二进制表示(binary signed digit representation, BSDR), 由字符串 Z 计算 Z 对应的数值为 $\sigma(Z)=\sum_{i=0}^{31}z_i2^i\in\mathbb{Z}/(2^{32})$.

设两个 512 比特的消息块 M', M 在压缩函数计算中寄存器的取值分别为 Q'_t 和 Q_t($-4\leqslant t\leqslant 80$), 则寄存器上的比特差分为 $\Delta Q_t=(Q'_t[i]-Q_t[i])_{i=0}^{31}\in\{-1,0,1\}^{32}$.

由此可得比特差分的所有取值情况见表 1.

表 1 比特差分

$\Delta Q_t[i]$	$Q'_t[i]$ 和 $Q_t[i]$ 的取值情况
0	$Q'_t[i]=Q_t[i]$
+1	$Q'_t[i]=1,\ Q_t[i]=0$
−1	$Q'_t[i]=0,\ Q_t[i]=1$

设 NAF 是有符号二进制的非相邻表示, $\sigma(Z)$表示计算有符号二进制表示的数值, 扰动向量为 DV_i, 其中, i=0,…,79. 在允许进位扩展的条件下, 设进位扩展的位数最高为 u 位, 由扰动向量可以推导比特差分ΔQ_t 的所有可能取值的集合为

$$\boldsymbol{Q}_{u,t}=\left\{BSDRY\left|\begin{array}{l}\sigma(Y)=\sigma(Z)\\ Z[i]\in\{-DV_{t-1}[i],DV_{t-1}[i]\},i=0,...,31\\ w(Y)\leqslant w(NAF(\sigma(Y)))+u\end{array}\right.\right\},$$

其中, w 代表汉明重量量.

进一步, 将 $\boldsymbol{Q}_{u,t}$ 简记为 $\boldsymbol{Q}_{ut}$.

类似地, 消息差分$\Delta W_t\in\{-1,0,1\}^{32}$ 是 M', M 扩展后的第 t−1 个消息 W'_t 和 W_t 逐比特求差分得到的. 寄存器的差分值δQ_t 为 $Q'_t,Q_t\in\mathbb{Z}/2^{32}\mathbb{Z}$ 时两者的差值 Q'_t-Q_t. 消息的差分值δW_t 为 $W'_t,W_t\in\mathbb{Z}/2^{32}\mathbb{Z}$ 时两者的差值 W'_t-W_t. 消息差分值δW_t的所有可能取值的集合为 $\boldsymbol{W}_t=\{\sigma(\Delta W_t)|\Delta W_t[i]\in\{-DW_t[i],DW_t[i]\},i=0,\ldots,31\}$. 其中, DW_t 是无符号的消息差分, 可由扰动向量给出.

3.2 布尔函数差分和比特条件

设消息块 M', M 在压缩函数计算中第 t 步的布尔函数输出分别为 F'_t 和 F_t, 则布尔函数差分ΔF_t每一比特的取值方法与比特差分一致, 即:

$$\Delta F_t[b]=f_t(Q'_{t-1}[b],RL(Q'_{t-2},30)[b],RL(Q'_{t-3},30)[b])-f_t(Q_{t-1}[b],RL(Q_{t-2},30)[b],RL(Q_{t-3},30)[b]).$$

从而, $\Delta F_t[b]\in\{-1,0,1\}$. 给定布尔函数输入的比特差分$\Delta Q_{t-1}[b]$, $\Delta RL(Q_{t-2},30)[b]$, $\Delta RL(Q_{t-3},30)[b])$, 设 $V_{t,b}$ 是在满足 3 个比特差分的条件下$\Delta F_t[b]$的所有可能的取值的集合, 则 $V_{t,b}\subset\{-1,0,1\}$.

由$\Delta F_t[b]$的表达式可知, 其取值由 $Q'_{t-1}[b]$ 和 $Q_{t-1}[b]$、$RL(Q'_{t-2},30)[b]$ 和 $RL(Q_{t-2},30)[b]$、$RL(Q'_{t-3},30)[b]$ 和 $RL(Q_{t-3},30)[b]$共 6 个比特决定. 而比特差分只描述了同一个寄存器相同比特位上的两个取值之间的关系, 所以比特差分不能满足研究布尔函数差分的需要, 进而引入比特条件的概念, 以此描述上面 6 个比特之间的取值关系.

在同一寄存器上, 除比特差分的 3 种情况(对应 $q_t[b]$=·,+,−)外, 将 $Q_t[b]=Q'_t[b]$ 拆分成 $Q_t[b]=Q'_t[b]=0$ 和 $Q_t[b]=Q'_t[b]=1$ 这两种情况(对应 $q_t[b]$=0,1). 同一寄存器上的比特条件称为直接条件. 相对地, 两个不同寄存器上的比特条件称为间接条件.

在 SHA-1 算法中, 从布尔函数的 3 个输入之间的关系 $F_t=f_t(Q_{t-1},RL(Q_{t-2},30),RL(Q_{t-3},30))$可知: 对于相邻的两个寄存器, 可能是相同的比特位作为输入(即 $RL(Q_{t-2},30)$和 $RL(Q_{t-3},30)$), 也可能是移位 30 比特的关系(即 Q_{t-1} 和 $RL(Q_{t-2},30)$), 而相隔的两个寄存器之间一定是移位 30 比特的关系(即 Q_{t-1} 和 $RL(Q_{t-3},30)$). 再将每种情况分成前向相关和后向相关, 以此得到比特条件的所有取值见附录 1.

3.3 由比特差分确定比特条件和布尔函数差分

在$\Delta F_t[b]$的 3 个输入比特位上, 每个位置上的比特差分都有 3 个可能的取值(分别为 0,+1,−1), 所以共有 27 种取值情况. 在这 27 种取值组合中, 某些情况下布尔函数的输出差分不唯一(即$|V_{t,b}|>1$). 本文对于 SHA-1 的 3 个不同的轮函数在 27 种取值情况下的布尔函数输出差分以及相对应的比特条件的取值总结在附录 2 中.

在附录 2 的基础上, 可以在 3 个输入寄存器上的比特差分ΔQ_{t-3}, ΔQ_{t-2}, ΔQ_{t-1} 已知的情况下, 逐比特确定布尔函数差分ΔF_t和比特条件 q_{t-3}, q_{t-2}, q_{t-1}: 查询表 2-1(或表 2-2、表 2-3), $\Delta Q_{t-3}[b+2 \bmod 32]$, $\Delta Q_{t-2}[b+2 \bmod 32]$, $\Delta Q_{t-1}[b]$的取值下对应的 g 为布尔函数可能的输出差分, 所以, 所有的 g 值组成的集合即为 $V_{t,b}\subset\{-1,0,1\}$. 进一

步, 每个 g 的取值下的前向(后向)条件即为对应的比特条件 $q_{t-1}[b]$, $q_{t-2}[b+2 \bmod 32]$, $q_{t-3}[b+2\bmod 32]$. 当 b 跑集合{0,1,…,31}后, 布尔函数的输出差分ΔF_t 的集合即为 $(V_{t,b})_{b=0}^{31} \in \{-1,0,1\}^{32}$, $(V_{t,b})_{b=0}^{31}$ 中的每个元素ΔF_t 与对应的前向(后向)比特条件 $(q_k)_{k=t-3}^{t-1}$ 形成二元组 $(\Delta F_t,(q_k)_{k=t-3}^{t-1})$, 设所有这样的二元组组成的集合为 $FFq_t(BFq_t)$.

4 带比特条件的差分路径构造算法

本节前向扩展和后向扩展算法的核心是考虑布尔函数差分与比特条件之间的关系,也就是在布尔函数的 3 个输入寄存器上的比特差分已知的条件下, 如何确定布尔函数差分和比特条件. 对于差分路径, Marc Stevens 提出了第 t_b 到 t_e 上的差分路径的两种形式[3], 其中, $t_b,t_e\in\mathbb{Z}$, 且 $1\leqslant t_b\leqslant t_e\leqslant 80$:

$$P_1=\{(q_t^1)_{t=t_b-4}^{t_e+1},(\delta W_t^1)_{t=t_b}^{t_e}\},$$

$$P_2=\{\delta RL(Q_{t_b-4}^2,30),(\Delta Q_t^2)_{t_b-3}^{t_e},\delta Q_{t_e+1}^2,(\Delta F_t^2)_{t_b}^{t_e},(\delta W_t^2)_{t_b}^{t_e}\}.$$

第 1 种差分路径 P_1 由第 t_b 到 t_e 步上寄存器的比特条件和消息差分组成, 第 2 种差分路径 P_2 细致地记录了 M', M 在第 t_b 到 t_e 步路径上的各项差分, 包括了比特差分或差分值、布尔函数差分和消息差分, 单纯从描述的变量来看, P_1 的结构较 P_2 简单.

P_1 在第 1 轮差分路径搜索过程中得到了应用. P_1 虽然缺少对寄存器差分、布尔函数差分和消息差分的刻画, 但是在路径搜索过程中并没有避免确定布尔函数差分等的步骤. 对 P_1 的中间连接过程是逐比特确定未知的比特条件和布尔函数差分的过程, 但其前向扩展和后向扩展算法相对复杂, 并且在每一次扩展时对布尔函数的 3 个输入寄存器上的比特条件要求苛刻.

P_2 用于分析后 60 步局部碰撞的差分性质, 通过路径约减技术确定后 60 步的输入/输出差分. 其前向扩展和后向扩展过程简单且易于执行, 由于没有针对 P_2 进行路径搜索, 所以还没有针对 P_2 的中间连接算法. 但 P_2 中没有涉及比特条件, 不利于碰撞攻击后续步骤的进行(比如确定第 2 轮比特条件).

基于以上原因, 为了解决 P_1 和 P_2 存在的弊端, 同时融合两者的优势, 提出新的差分路径形如:

$$P=\{\delta RL(Q_{t_b-4},30),(\Delta Q_t)_{t_b-3}^{t_e},\delta Q_{t_e+1},(\Delta F_t)_{t_b}^{t_e},(\delta W_t)_{t_b}^{t_e},(q_t)_{t_b-4}^{t_e+1}\}.$$

以下称 P 为带比特条件的差分路径. P 全面地描述了差分路径上各个变量上的差分(值)和比特条件, 既可以用于后 60 步的差分分析, 也可以进行第 1 轮的差分路径搜索. 在进行后 60 步的差分分析时, 较 P_2 而言增加了对比特条件的分析, 可以更全面地对后 60 步输入/输出差分进行评估. 本节提出的带比特条件的差分路径搜索算法分为前向扩展、后向扩展和中间连接算法这 3 个部分. P 的前向和后向扩展算法基于 P_2 的前向和后向扩展过程发展而来, 解决了 P_1 的扩展过程复杂且对比特条件要求高的问题. 中间连接过程是将前向搜索和后向搜索结果结合起来形成完整的差分路径, 算法描述如下.

4.1 带比特条件的前向扩展算法

带比特条件的前向扩展算法是基于压缩函数对寄存器的更新迭代方程展开的. 已知第 i 步之前的差分路径, 首先确定第 i 步的布尔函数差分, 再通过迭代方程计算下一个寄存器上的差分, 同时对相应寄存器上的比特条件进行更新, 这样就完成了第 i 步的前向扩展过程. 扩展过程的细节如下.

对于带比特条件的差分路径, 为了构造第 t_b 到 t_e 步的差分路径 P, 第 $i(t_b\leqslant i\leqslant t_e)$步的前向扩展算法基于压缩函数中第 i 步的迭代方程: $Q_{i+1}=RL(Q_i,5)+F_i+RL(Q_{i-4},30)+W_i+AC_i$, 根据已知条件的不同将扩展过程分成 $i=t_b$ 和 $i\neq t_b$ 两种情况.

1) 当 $i\neq t_b$ 时, 进行扩展之前的已知条件应有 $(\Delta Q_t)_{t=i-4}^{i-1}$, δQ_i, $(\hat{q}_t)_{t=i-3}^{i-1}$ (本文中的 $\hat{q}$ 符号代表差分路径构造过程中既得的比特条件), 扩展过程分为 3 步.

   1. 由已知比特差分 $(\Delta Q_t)_{t=i-3}^{i-1}$ 确定集合 FFq_i, 对于 $(\Delta F_i,(q_k)_{k=i-3}^{i-1})\in FFq_i$, 且 $(q_k)_{k=i-3}^{i-1}$ 与 $(\hat{q}_t)_{t=i-3}^{i-1}$ 不产生矛盾, 确定这一步的布尔函数差分为ΔF_i, 并用 FFq_i 中的 $(q_k)_{k=i-3}^{i-1}$ 更新 $(\hat{q}_t)_{t=i-3}^{i-1}$;
   2. 对于$\delta W_i\in \boldsymbol{W}_i$ 和$\Delta Q_i\in \boldsymbol{Q}_i$ 满足$\sigma(\Delta Q_i)=\delta Q_i$, 计算$\delta Q_{i+1}$:

$$\delta Q_{i+1}=\sigma(RL(\Delta Q_i,5))+\sigma(\Delta F_i)+\sigma(RL(\Delta Q_{i-4},30))+\delta W_i;$$

3. 将 δW_i, ΔF_i, δQ_{i+1} 添加到差分路径中, 此时差分路径扩展变成:

$$P=\{\delta RL(Q_{t_b-4},30),(\Delta Q_t)_{t=t_b-3}^{i},\delta Q_{i+1},(\Delta F_t)_{t_b}^{i},(\delta W_t)_{t_b}^{i},(\hat{q}_t)_{t=t_b-4}^{i}\};$$

2) 当 $i=t_b$ 时, 进行扩展之前的已知条件应有 $\delta RL(Q_{t_b-4},30)$, $(\Delta Q_t)_{t=t_b-3}^{t_b}$, $(\hat{q}_t)_{t=t_b-3}^{t_b}$. 因为同样已知布尔函数的 3 个输入寄存器的比特差分, 所以扩展过程的第 1 步与 $i\neq t_b$ 时相同. 其与 $i\neq t_b$ 的不同点在于第 2 步: 对于 $\delta W_i\in \boldsymbol{W}_i$, 计算出 $\delta Q_{i+1}=\delta Q_{t_b+1}$: $\delta Q_{t_b+1}=\sigma(RL(\Delta Q_{t_b},5))+\sigma(\Delta F_{t_b})+\delta RL(Q_{t_b-4},30)+\delta W_{t_b}$.

综上, 构造第 t_b 到 t_e 步的差分路径 P 的前向扩展算法为算法 1.

算法 1. 带比特条件的前向扩展算法.

输入: $\delta RL(Q_{t_b-4},30)$, $(\Delta Q_t)_{t=t_b-3}^{t_b}$, $(\hat{q}_t)_{t=t_b-3}^{t_b}$;

输出: 第 t_b 到 t_e 步的所有差分路径 P.

1. $i\leftarrow t_b$;
2. **FOR** $(\Delta F_{t_b},(q_k)_{k=t_b-3}^{t_b-1})\in FFq_{t_b}$ 且与 $(\hat{q}_t)_{t=t_b-3}^{t_b-1}$ 不矛盾
   更新 $(\hat{q}_t)_{t=t_b-3}^{t_b-1}$;
3. **FOR** $\delta W_{t_b}\in \boldsymbol{W}_{t_b}$
   计算 δQ_{t_b+1}; $i\leftarrow i+1$;
4. **FOR** $(\Delta F_i,(q_k)_{k=i-3}^{i-1})\in FFq_i$ 与 $(\hat{q}_t)_{t=i-3}^{i-1}$ 不矛盾
   更新 $(\hat{q}_t)_{t=i-3}^{i-1}$;
5. **FOR** $\Delta Q_i\in \boldsymbol{Q}_i$ 满足 $\sigma(\Delta Q_i)=\delta Q_i$
6. **FOR** $\delta W_i\in \boldsymbol{W}_i$
   由 ΔQ_i 确定 $\hat{q}_i$, 计算 δQ_{i+1}; $i\leftarrow i+1$;
7. **IF** $i<t_e$, 返回步骤 4;
   ELSE Return: $P=\{\delta RL(Q_{t_b-4},30),(\Delta Q_t)_{t_b-3}^{t_e},\delta Q_{t_e+1},(\Delta F_t)_{t_b}^{t_e},(\delta W_t)_{t_b}^{t_e},(\hat{q}_t)_{t_b-3}^{t_e}\}$
8. **Return** //前向扩展失败

4.2 带比特条件的后向扩展算法

带比特条件的后向扩展算法是基于压缩函数对寄存器的反向迭代方程展开的, 与前向扩展算法类似. 已知第 i 步之后的差分路径, 首先确定第 i 步的布尔函数差分, 再通过反向迭代方程计算前一个寄存器上的差分, 同时对相应寄存器上的比特条件进行更新, 这样就完成了第 i 步的后向扩展过程. 扩展过程的细节如下.

对于带比特条件的差分路径, 为了构造第 t_b 到 t_e 步的差分路径 P, 第 $j(t_b\leqslant j\leqslant t_e)$步的后向扩展算法基于压缩函数中第 j 步的反向迭代方程: $RL(Q_{j-4},30)=Q_{j+1}-RL(Q_j,5)-F_j-W_j-AC_j$. 与前向扩展算法类似, 根据已知条件的不同, 可以将扩展过程分成 $j=t_e$ 和 $j\neq t_e$ 两种情况.

1) 当 $j\neq t_e$ 时, 进行扩展之前的已知条件应有 $\delta RL(Q_{j-3},30)$, $(\Delta Q_t)_{t=j-2}^{j+1}$, $(\hat{q}_t)_{t=j-2}^{j-1}$, 扩展过程分为 3 步.

1. 对于满足 $\sigma(RL(\Delta Q_{j-3},30))=\delta RL(Q_{j-3},30)$ 的比特差分 $\Delta Q_{j-3}\in \boldsymbol{Q}_{j-3}$, 由 ΔQ_{j-3} 确定 $\hat{q}_{j-3}$, 由比特差分 $(\Delta Q_t)_{t=j-3}^{j-1}$ 确定集合 BFq_j. 对于 BFq_j 中的元素 $(\Delta F_j,(q_k)_{k=j-3}^{j-1})$, 若 $(q_k)_{k=j-3}^{j-1}$ 与 $(\hat{q}_t)_{t=j-3}^{j-1}$ 不产生矛盾, 则得到这一步的布尔函数差分为 ΔF_j, 并用 BFq_j 中的 $(q_k)_{k=j-3}^{j-1}$ 更新比特条件 $(\hat{q}_t)_{t=j-3}^{j-1}$;
2. 对于 $\delta W_j\in \boldsymbol{W}_j$, 计算 $\delta RL(Q_{j-4},30)=\sigma(\Delta Q_{j+1})-\sigma(RL(\Delta Q_j,5))-\sigma(\Delta F_j)-\delta W_j$, 得到差分值 $\delta RL(Q_{j-4},30)$;
3. 将 δW_j, ΔQ_{j-3}, ΔF_j, $\delta RL(Q_{j-4},30)$, $\hat{q}_{j-3}$ 添加到差分路径中, 并对 $(\hat{q}_t)_{t=j-2}^{j-1}$ 进行更新, 这样就完成了第 $j\neq t_e$ 步的后向扩展过程, 此时, 差分路径扩展变成:

$$P=\{\delta RL(Q_{j-4},30),(\Delta Q_t)_{j-3}^{t_e},\delta Q_{t_e+1},(\Delta F_t)_{j}^{t_e},(\delta W_t)_{j}^{t_e},(\hat{q}_t)_{j-3}^{t_e}\};$$

2) 当 $j=t_e$ 时, 进行扩展之前的已知条件应有 $(\Delta Q_t)_{t=t_e-3}^{t_e}$, δQ_{t_e+1}, $(\hat{q}_t)_{t=t_e-3}^{t_e}$, 其与 $j\neq t_e$ 的不同点有两个.

1. 第 1 步中无需选择ΔQ_{j-3}, 可以直接确定 BFq_j;
2. 第 2 步$\delta RL(Q_{j-4},30)$的计算表达式为$\delta RL(Q_{j-4},30)=\sigma Q_{j+1}-\sigma(RL(\Delta Q_j,5))-\sigma(\Delta F_j)-\delta W_j$.

综上, 构造第 t_b 到 t_e 步的差分路径 P 的后向扩展算法为算法 2.

算法 2. 带比特条件的后向扩展算法.

输入: $(\Delta Q_t)_{t=t_e-3}^{t_e}$, δQ_{t_e+1}, $(\hat{q}_t)_{t=t_e-3}^{t_e}$;

输出: 第 t_b 到 t_e 步的所有差分路径 P.

1. $j\leftarrow t_e$;
2. **FOR** $(\Delta F_{t_e},(q_k)_{k=t_e-3}^{t_e-1})\in BFq_{t_e}$ 且与 $(\hat{q}_t)_{t=t_e-3}^{t_e-1}$ 不矛盾
   更新 $(\hat{q}_t)_{t=t_e-3}^{t_e-1}$;
3. **FOR** $\delta W_{t_e}\in \boldsymbol{W}_{t_e}$
   计算 $\delta RL(Q_{t_e-4},30)=\delta Q_{t_e+1}-\sigma(RL(\Delta Q_{t_e},5))-\sigma(\Delta F_{t_e})-\delta W_{t_e}$, $j\leftarrow j-1$;
4. **FOR** $\Delta Q_{j-3}\in \boldsymbol{Q}_{j-3}$ 满足$\sigma(RL(\Delta Q_{j-3},30))=\delta RL(Q_{j-3},30)$
   由ΔQ_{j-3} 确定 $\hat{q}_{j-3}$;
5. **FOR** $(\Delta F_j,(q_k)_{k=j-3}^{j-1})\in BFq_j$ 与 $(\hat{q}_t)_{t=j-3}^{j-1}$ 不矛盾
   更新 $(\hat{q}_t)_{t=j-3}^{j-1}$;
6. **FOR** $\delta W_j\in \boldsymbol{W}_j$,
   计算$\delta RL(Q_{j-4},30)=\sigma(\Delta Q_{j+1})-\sigma(RL(\Delta Q_j,5))-\sigma(\Delta F_j)-\delta W_j$, $j\leftarrow j-1$;
7. **IF** $j>t_b$, 返回步骤 4;
   ELSE RETURN: $P=\{\delta RL(Q_{t_b-4},30),(\Delta Q_t)_{t_b-3}^{t_e},\delta Q_{t_e+1},(\Delta F_t)_{t_b}^{t_e},(\delta W_t)_{t_b}^{t_e},(\hat{q}_t)_{t_b-3}^{t_e}\}$

4.3 带比特条件的中间连接算法

对于一组前向扩展到第 l 步和后向扩展到第 l+6 步的搜索结果, 两条路径中的比特条件、寄存器差分、布尔函数差分以及消息差分展示在表 2 中. 带比特条件的中间连接算法是为了确定空白部分和ΔQ_{l+1}, ΔQ_{l+2} 的取值,使其形成完整有效的差分路径.

表 2 中间连接的已知条件

比特条件	…	q_{l-2}	q_{l-1}	q_l			q_{l+3}	q_{l+4}	q_{l+5}	q_{l+6}	q_{l+6}	…
消息差分	…	δW_{l-2}	δW_{l-1}	δW_l						δW_{l+6}	δW_{l+7}	…
布尔函数差分	…	ΔF_{l-2}	ΔF_{l-1}	ΔF_l						ΔF_{l+6}	ΔF_{l+7}	…
寄存器差分	…	ΔQ_{l-2}	ΔQ_{l-1}	ΔQ_l	δQ_{l+1}	$\delta RL(Q_{l+2},30)$	ΔQ_{l+3}	ΔQ_{l+4}	ΔQ_{l+5}	ΔQ_{l+6}	ΔQ_{l+7}	…

本文的中间连接算法分为 5 个部分, 分别是第 1 次前向扩展、第 1 次后向扩展、第 2 次前向扩展、第 2 次后向扩展和第 3 次后向扩展. 每次扩展过程都会确定部分变量的取值. 对于这些取值, 寄存器和布尔函数上的比特差分一旦确定, 后面的扩展过程中都将其视为已知条件, 而比特条件在算法执行的过程中是不断更新的. 具体细节如下.

4.3.1 第 1 次前向扩展

首先进行第 1 次前向扩展, 这实际上是第 l+1 步的前向扩展过程.

1. 由已知比特差分 $(\Delta Q_t)_{t=l-2}^{l}$ 确定集合 FFq_{l+1}, 对于 $(\Delta F_{l+1},(q_k)_{k=l-2}^{l})\in FFq_{l+1}$, 且 $(q_k)_{k=l-2}^{l}$ 与已知的 $(\hat{q}_t)_{t=l-2}^{l}$ 不产生矛盾, 则确定这一步的布尔函数差分为ΔF_{l+1}, 并用 FFq_{l+1} 中的 $(q_k)_{k=l-2}^{l}$ 更新比特条件 $(\hat{q}_t)_{t=l-2}^{l}$;
2. 对于$\delta W_{l+1}\in \boldsymbol{W}_{l+1}$ 和满足$\sigma(\Delta Q_{l+1})=\delta Q_{l+1}$ 的$\Delta Q_{l+1}\in \boldsymbol{Q}_{l+1}$, 由$\Delta Q_{l+1}$ 确定 q_{l+1}, 并计算δQ_{l+2}:

$$\delta Q_{l+2}=\sigma(RL(\Delta Q_{l+1},5))+\sigma(\Delta F_{l+1})+\sigma(RL(\Delta Q_{l-3},30))+\delta W_{l+1}.$$

进行本次扩展之后, 由于本次扩展得到的δQ_{l+2} 和原有后向扩展结果$\delta RL(Q_{l+2},30)$均为差分值, 所以无法进行两者的判断.

4.3.2 第 1 次后向扩展

第 1 次后向扩展与第 l+5 步后向扩展类似, 扩展过程如下.

1. 对于同时满足下面两个条件的$\Delta Q_{l+2}\in \boldsymbol{Q}_{l+2}$:

① $\sigma(\Delta Q_{l+2})=\delta Q_{l+2}$;

② $\sigma(RL(Q_{l+2},30))=\delta RL(Q_{l+2},30)$;

由ΔQ_{l+2} 确定 $\hat{q}_{l+2}$, 由比特差分 $(\Delta Q_t)_{t=l+2}^{l+4}$ 确定集合 BFq_{l+5}, 对于 $(\Delta F_{l+5},(q_k)_{k=l+2}^{l+4})\in BFq_{l+5}$, 且 $(q_k)_{k=l+2}^{l+4}$ 与已知的 $(\hat{q}_t)_{t=l+2}^{l+4}$ 不产生矛盾, 则确定这一步的布尔函数差分为ΔF_{l+5}, 并用 BFq_{l+5} 中的 $(q_k)_{k=l+2}^{l+4}$ 更新比特条件.

2. 对于$\delta W_{l+5}\in \boldsymbol{W}_{l+5}$, 计算$\delta RL(Q_{l+1},30)$: $\delta RL(Q_{l+1},30)=\sigma(\Delta Q_{l+6})-\sigma(RL(\Delta Q_{l+5},5))-\sigma(\Delta F_{l+5})-\delta W_{l+5}$.

进行本次扩展后, 需判断$\delta RL(Q_{l+1},30)\in\{RL(\delta Q_{l+1},30),RL(\delta Q_{l+1},30)-2^{30},RL(\delta Q_{l+1},30)+1,RL(\delta Q_{l+1},30)-2^{30}+1\}$是否成立: 如果成立, 进行第 2 次前向扩展; 否则, 重新进行第 1 次后向扩展. 其中, 第 2 步的判断条件是基于下面一个引理.

引理 1. 给定寄存器 Q_l 上的比特差分为ΔQ_l, 则 Q_l 的移位差分值$\delta RL(Q_l,30)$有 4 种可能的取值, 分别为 $RL(\delta Q_l,30)$, $RL(\delta Q_l,30)-2^{30}$ 和 $RL(\delta Q_l,30)+1$, $RL(\delta Q_l,30)-2^{30}+1$(这里将$\sigma(RL(\Delta Q_l,30))$简记为 $RL(\delta Q_l,30)$).

4.3.3 第 2 次前向扩展

与第 l+2 步前向扩展类似, 第 2 次前向扩展的过程如下.

1. 由已知比特差分 $(\Delta Q_t)_{t=l-1}^{l+1}$ 确定集合 FFq_{l+2}, 对于 $(\Delta F_{l+2},(q_k)_{k=l-1}^{l+1})\in FFq_{l+2}$, 且 $(q_k)_{k=l-1}^{l+1}$ 与已经得到的 $(\hat{q}_t)_{t=l-1}^{l+1}$ 不产生矛盾, 则得到这一步的布尔函数差分ΔF_{l+2}, 并用 FFq_{l+2} 中的 $(q_k)_{k=l-1}^{l+1}$ 更新比特条件;

2. 对于$\delta W_{l+2}\in \boldsymbol{W}_{l+2}$, 计算$\delta Q_{l+3}$: $\delta Q_{l+3}=\sigma(RL(\Delta Q_{l+2},5))+\sigma(\Delta F_{l+2})+\sigma(RL(\Delta Q_{l-2},30))+\delta W_{l+2}$.

进行本次扩展之后, 需要判断$\delta Q_{l+3}=\sigma(\Delta Q_{l+3})$是否成立: 如果成立, 进行第 2 次后向扩展; 否则, 重新进行第 2 次前向扩展.

4.3.4 第 2 次后向扩展

与第 l+4 步后向扩展类似, 第 2 次后向扩展的过程如下.

1. 由比特差分 $(\Delta Q_t)_{t=l+1}^{l+3}$ 确定集合 BFq_{l+4}, 对于 $(\Delta F_{l+4},(q_k)_{k=l+1}^{l+3})\in BFq_{l+4}$, 且 $(q_k)_{k=l+1}^{l+3}$ 与已经得到的 $(\hat{q}_t)_{t=l+1}^{l+3}$ 不产生矛盾, 则得到这一步的布尔函数差分ΔF_{l+4}, 并用 BFq_{l+4} 中的 $(q_k)_{k=l+1}^{l+3}$ 更新比特条件;

2. 对于$\delta W_{l+4}\in \boldsymbol{W}_{l+4}$, 计算$\delta RL(Q_l,30)$: $\delta RL(Q_l,30)=\sigma(\Delta Q_{l+5})-\sigma(RL(\Delta Q_{l+4},5))-\sigma(\Delta F_{l+4})-\delta W_{l+4}$.

进行本次扩展之后, 需要判断$\delta RL(Q_l,30)\in\{RL(\delta Q_l,30),RL(\delta Q_l,30)-2^{30},RL(\delta Q_l,30)+1,RL(\delta Q_l,30)-2^{30}+1\}$是否成立: 如果成立, 进行第 3 次后向扩展; 否则, 重新进行第 2 次后向扩展.

4.3.5 第 3 次后向扩展

最后进行第 3 次后向扩展, 与第 l+3 步后向扩展类似, 扩展过程如下.

1. 由已知比特差分 $(\Delta Q_t)_{t=l}^{l+2}$ 确定集合 BFq_{l+3}, 对于 $(\Delta F_{l+3},(q_k)_{k=l}^{l+2})\in BFq_{l+3}$, 且 $(q_k)_{k=l}^{l+2}$ 与已经得到的 $(\hat{q}_t)_{t=l}^{l+2}$ 不产生矛盾, 则得到这一步的布尔函数差分ΔF_{l+3}, 并用 BFq_{l+3} 中的 $(q_k)_{k=l}^{l+2}$ 更新比特条件;

2. 对于$\delta W_{l+3}\in \boldsymbol{W}_{l+3}$, 计算$\delta RL(Q_{l-1},30)$: $\delta RL(Q_{l-1},30)=\sigma(\Delta Q_{l+4})-\sigma(RL(\Delta Q_{l+3},5))-\sigma(\Delta F_{l+3})-\delta W_{l+3}$.

进行本次扩展之后, 判断$\delta RL(Q_{l-1},30)\in\{RL(\delta Q_{l-1},30),RL(\delta Q_{l-1},30)-2^{30},RL(\delta Q_{l-1},30)+1,RL(\delta Q_{l-1},30)-2^{30}+1\}$是否成立: 如果成立, 则成功完成了中间连接; 否则, 重新进行第 3 次后向扩展.

综合以上分析, 带比特条件的中间连接算法流程如下.

算法 3. 带比特条件的中间连接算法.

输入: 前向扩展到第 l 步的差分路径 $\hat{q}_l,\hat{q}_{l-1},...,\delta W_l,\delta W_{l-1},...,\Delta F_l,\Delta F_{l-1},...,\delta Q_{l+1},\Delta Q_l,\Delta Q_{l-1},...$;

后向扩展到第 l+6 步的差分路径 $\hat{q}_{l+3},\hat{q}_{l+4},...,\delta W_{l+4},\delta W_{l+5},...,\Delta F_{l+4},\Delta F_{l+5},...,\delta RL(Q_{l+2},30),\Delta Q_{l+3},...$;

输出: $\{q'_{l-2},q''_{l-1},q'''_l,q'''_{l+1},q''_{l+2},q''_{l+3},q'_{l+4},(f_{l+i})_{i=1}^5,(w_{l+i})_{i=1}^5,Q_{l+1},Q_{l+2}\}$.

First forward extension (第 1 次前向扩展)

FOR $(\Delta F_{l+1},(q_k)_{k=l-2}^{l})\in FFq_{l+1}$ 且与 $(\hat{q}_t)_{t=l-2}^{l}$ 不矛盾 //(第 1 次前向扩展, first forward extension)

f_{l+1}←ΔF_{l+1}, 更新 $(\hat{q}_t)_{t=l-2}^{l}$;

$\hat{q}_{l-2} \leftarrow q'_{l-2}$, $\hat{q}_{l-1} \leftarrow q'_{l-1}$, $\hat{q}_l \leftarrow q'_l$;

FOR $\Delta Q_{l+1} \in \boldsymbol{Q}_{l+1}$ 满足 $\sigma(\Delta Q_{l+1})=\delta Q_{l+1}$

Q_{l+1}←ΔQ_{l+1}, 并由 Q_{l+1} 得到比特条件 q_{l+1};

FOR $\delta W_{l+1} \in \boldsymbol{W}_{l+1}$

w_{l+1}←δW_{l+1}, 并计算

$\delta Q_{l+2}=\sigma(RL(\Delta Q_{l+1},5))+\sigma(f_{l+1})+\sigma(RL(\Delta Q_{l-3},30))+w_{l+1}$

First backward extension;

IF (first backward extenstion 返回正确连接)

Return $\{q'_{l-2},q''_{l-1},q'''_l,q'''_{l+1},q'''_{l+2},q''_{l+3},q'_{l+4},(f_{l+i})_{i=1}^{5},Q_{l+1},Q_{l+2},(w_{l+i})_{i=1}^{5}\}$

//得到结果, 来自 First backward extension

Return; //整体连接失败

First backward extension (第 1 次后向扩展)

FOR $\Delta Q_{l+2} \in \boldsymbol{Q}_{l+2}$ 满足 $\sigma(\Delta Q_{l+2})=\delta Q_{l+2}$ 且 $\sigma(RL(\Delta Q_{l+2},30))=\delta RL(Q_{l+2},30)$

Q_{l+2}←ΔQ_{l+2}, 由 Q_{l+2} 确定 $\hat{q}_{l+2}$;

FOR $(\Delta F_{l+5},(q_k)_{k=l+2}^{l+4}) \in BFq_{l+5}$ 且与 $(\hat{q}_t)_{t=l+2}^{l+4}$ 不矛盾

f_{l+5}←ΔF_{l+5}, 更新 $(\hat{q}_t)_{t=l+2}^{l+4}$;

$\hat{q}_{l+2} \leftarrow q'_{l+2}$, $\hat{q}_{l+3} \leftarrow q'_{l+3}$, $\hat{q}_{l+4} \leftarrow q'_{l+4}$;

FOR $\delta W_{l+5} \in \boldsymbol{W}_{l+5}$

w_{l+5}←δW_{l+5}, 计算 $\delta RL(Q_{l+1},30)$:

$\delta RL(Q_{l+1},30)=\sigma(\Delta Q_{l+6})-\sigma(RL(\Delta Q_{l+5},5))-\sigma(f_{l+5})-w_{l+5}$

IF $\delta RL(Q_{l+1},30)=\sigma(RL(Q_{l+1},30))(-2^{30})(+1)$

Second forward extension;

IF (second forward extension 返回正确连接)

Return $\{q'_{l-2},q''_{l-1},q'''_l,q'''_{l+1},q'''_{l+2},q''_{l+3},q'_{l+4},(f_{l+i})_{i=1}^{5},Q_{l+1},Q_{l+2},(w_{l+i})_{i=1}^{5}\}$

//得到结果, 来自 Second forward extension

Return 连接失败; //未找到正确连接

Second forward extension (第 2 次前向扩展)

FOR $(\Delta F_{l+2},(q_k)_{k=l-1}^{l+1}) \in FFq_{l+2}$ 与 q'_{l-1}, q'_l, q_{l+1} 不矛盾

f_{l+2}←ΔF_{l+2}; 更新 q'_{l-1}, q'_l, q_{l+1}, 得到 q''_{l-1}, q''_l, q'_{l+1};

FOR $\delta W_{l+2} \in \boldsymbol{W}_{l+2}$

w_{l+2}←δW_{l+2}, 并计算 δQ_{l+3}: $\delta Q_{l+3}=\sigma(RL(\Delta Q_{l+2},5))+\sigma(f_{l+2})+\sigma(RL(\Delta Q_{l-2},30))+w_{l+2}$

IF $\delta Q_{l+3}=\sigma(\Delta Q_{l+3})$

Second backward extension;

IF (second backward extension 返回正确连接)

Return $\{q'_{l-2},q''_{l-1},q'''_l,q'''_{l+1},q'''_{l+2},q''_{l+3},q'_{l+4},(f_{l+i})_{i=1}^{5},Q_{l+1},Q_{l+2},(w_{l+i})_{i=1}^{5}\}$

//得到结果, 来自 Second backward extension

Return 连接失败; //未找到正确连接

Second backward extension (第 2 次后向扩展)

FOR $(\Delta F_{l+4},(q_k)_{k=l+1}^{l+3}) \in BFq_{l+4}$ 与 q'_{l+1}, q'_{l+2}, q'_{l+3} 不矛盾

f_{l+4}←ΔF_{l+4}; 更新 q'_{l+1}, q'_{l+2}, q'_{l+3}, 得到 q''_{l+1}, q''_{l+2}, q''_{l+3};

FOR $\delta W_{l+4}\in \boldsymbol{W}_{l+4}$

$w_{l+4}\leftarrow\delta W_{l+4}$, 计算$\delta RL(Q_l,30)$: $\delta RL(Q_l,30)=\sigma(\Delta Q_{l+5})-\sigma(RL(\Delta Q_{l+4},5))-\sigma(f_{l+4})-w_{l+4}$

IF $\delta RL(Q_l,30)=RL(\delta Q_l,30)(-2^{30})(+1)$

Third backward extension;

IF (Third backward extension 返回正确连接)

Return $\{q'_{l-2},q''_{l-1},q'''_l,q'''_{l+1},q'''_{l+2},q''_{l+3},q'_{l+4},(f_{l+i})_{i=1}^5,Q_{l+1},Q_{l+2},(w_{l+i})_{i=1}^5\}$

//得到结果, 来自 Third backward extension

Return 连接失败; //未找到正确连接

Third backward extention (第 3 次后向扩展)

FOR $(\Delta F_{l+3},(q_k)_{k=l}^{l+2})\in BFq_{l+3}$ 与 q''_l, q''_{l+1}, q''_{l+2} 不矛盾

$f_{l+3}\leftarrow\Delta F_{l+3}$; 更新 q''_l, q''_{l+1}, q''_{l+2}, 得到 q'''_l, q'''_{l+1}, q'''_{l+2};

FOR $\delta W_{l+3}\in \boldsymbol{W}_{l+3}$

$w_{l+3}\leftarrow\delta W_{l+3}$, 计算$\delta RL(Q_{l-1},30)$: $\delta RL(Q_{l-1},30)=\sigma(\Delta Q_{l+4})-\sigma(RL(\Delta Q_{l+3},5))-\sigma(f_{l+3})-w_{l+3}$

IF $\delta RL(Q_{l-1},30)=RL(\delta Q_{l-1},30)(-2^{30})(+1)$

Return $\{q'_{l-2},q''_{l-1},q'''_l,q'''_{l+1},q'''_{l+2},q''_{l+3},q'_{l+4},(f_{l+i})_{i=1}^5,Q_{l+1},Q_{l+2},(w_{l+i})_{i=1}^5\}$ //得到正确连接结果

Return 连接失败; //未找到正确连接

注: 对于 $a,b\in\mathbb{Z}/2^{32}\mathbb{Z}$, $a=b(-2^{30})(+1)$意为 $a=b$ 或 $a=b-2^{30}$ 或 $a=b+1$ 或 $a=b-2^{30}+1$.

按照上述中间连接算法, 每步扩展过程对比特条件和寄存器差分的求取见表 3, 其中, 将 5 次扩展简记为 1F, 1B, 2F, 2B, 3B.

表 3 中间连接过程

比特条件		q_{l-2}	q_{l-1}	q_l			q_{l+3}	q_{l+4}	q_{l+5}	q_{l+6}	q_{l+7}
	1F	q_{l-2}	q_{l-1}	q_l	q_{l+1}						
	1B					q_{l+2}	q_{l+3}	q_{l+4}			
	2F		q_{l-1}	q_l	q_{l+1}	q_{l+2}					
	2B				q_{l+1}	q_{l+2}	q_{l+3}				
	3B			q_l	q_{l+1}	q_{l+2}					
消息差分		δW_{l-2}	δW_{l-1}	δW_l	δW_{l+1}	δW_{l+2}	δW_{l+3}	δW_{l+4}	δW_{l+5}	δW_{l+6}	δW_{l+7}
布尔函数差分		ΔF_{l-2}	ΔF_{l-1}	ΔF_l	ΔF_{l+1}	ΔF_{l+2}	ΔF_{l+3}	ΔF_{l+4}	ΔF_{l+5}	ΔF_{l+6}	ΔF_{l+7}
寄存器差分		ΔQ_{l-2}	ΔQ_{l-1}	ΔQ_l	δQ_{l+1}	$\delta RL(Q_{l+2},30)$	ΔQ_{l+3}	ΔQ_{l+4}	ΔQ_{l+5}	ΔQ_{l+6}	ΔQ_{l+7}
	1F				ΔQ_{l+1}	δQ_{l+2}					
	1B				$\delta RL(Q_{l+1},30)$	ΔQ_{l+2}					
	2F					ΔQ_{l+2}	δQ_{l+3}				
	2B			$\delta RL(Q_l,30)$	ΔQ_{l+1}						
	3B		$\delta RL(Q_{l-1},30)$	ΔQ_l							

4.3.6 扩展顺序的选择策略分析

理论上, 在中间连接过程的 5 步中, 每步连接都有前向扩展和后向扩展两种选择, 因此一共有 $2^5=32$ 种可能的中间连接方法. 设所有中间连接方法组成的集合为 S_{Joint}, 用 1 表示前向扩展, 0 表示后向扩展, 则 $S_{Joint}=\{0,1\}^5$. 以 S_{Joint} 中的元素 01001 为例, 其代表的中间连接方法为: 先后进行第 l+5 步后向扩展、第 l+1 步前向扩展、第 l+4 步后向扩展、第 l+3 步后向扩展、第 l+2 步前向扩展. 为了从 S_{Joint} 中选出成功率最高的中间连接方法, 本文提出了比特条件的更新次数、扩展结果的相容性和寄存器候选集合的正确率这 3 个指标:

- 指标 1: 比特条件的更新次数

不管是前向扩展还是后向扩展, 都会对若干比特条件进行更新(一次前向扩展更新 4 个寄存器上的比特条件, 一次后向扩展更新 3 个寄存器上的比特条件). 对某个寄存器上的比特条件而言, 更新次数增加可能导致比特条件的冲突, 进而导致中间连接算法的失败. 即使中间连接成功, 比特条件的增加也会导致差分路径概率的减小, 所以中间连接过程中比特条件的更新次数越少越好.

下面对 32 种连接方法按照前向扩展的次数分类, 分析每一类连接方法对各比特条件的更新次数. 这里将需要进行中间连接的前向和后向扩展结果(包括寄存器差分、比特条件、布尔函数差分、消息差分)称为中间连接的原始条件, 而将中间连接过程中得到的各项结果称为中间条件. 以前向扩展次数为2为例, 其比特条件的更新情况见表 4(此时, 前向扩展和后向扩展的前后顺序不会影响比特条件更新次数).

表 4 2 次前向扩展时比特条件的更新情况

原始条件	q_{l-2}	q_{l-1}	q_l			q_{l+3}	q_{l+4}
前向扩展	q_{l-2}	q_{l-1}	q_l	q_{l+1}			
前向扩展		q_{l-1}	q_l	q_{l+1}	q_{l+2}		
后向扩展					q_{l+2}	q_{l+3}	q_{l+4}
后向扩展				q_{l+1}	q_{l+2}	q_{l+3}	
后向扩展			q_l	q_{l+1}	q_{l+2}		

所以, 当前向扩展次数为 2 时, 对 $q_{l-2},q_{l-1},\ldots,q_{l+4}$ 的更新次数分别为 1, 2, 3, 3, 3, 2, 1 (其中, 对 q_{l+1}, q_{l+2} 的更新次数是指在第 1 次得到 q_{l+1}, q_{l+2} 之后对其的更新次数). 同理, 分析前向扩展次数为 0, 1, 3, 4, 5 时比特条件的更新情况, 将分析结果汇总在表 5 中. 从分析结果可得, 前向扩展的次数与比特条件的更新次数成正比. 所以, 如果单纯从比特条件的更新次数考虑, 前向扩展次数为 0 时, 中间连接的成功率最高. 但中间连接策略应综合考虑 3 个指标, 基于下文对指标 2 和指标 3 的分析, 最终选择的中间连接算法共有 2 次前向扩展.

表 5 比特条件更新次数汇总

	q_{l-2}	q_{l-1}	q_l	q_{l+1}	q_{l+2}	q_{l+3}	q_{l+4}	q_{l+5}
0	1	2	3	2	2	2	1	0
1	1	2	3	3	2	2	1	0
2	1	2	3	3	3	2	1	0
3	1	2	3	3	3	3	1	0
4	1	2	3	3	3	3	2	0
5	1	2	3	3	3	3	2	1

- 指标 2: 扩展结果的相容性

第 2 个影响中间连接的成功率的因素是扩展结果的相容性. 考虑寄存器上的差分, 比特差分,…,ΔQ_{l-1}, ΔQ_l,ΔQ_{l+3},ΔQ_{l+4},…和差分值δQ_{l+1}, $\delta RL(Q_{l+2},30)$为原始条件, 而在中间连接过程中确定的ΔQ_{l+1}, ΔQ_{l+2} 为中间条件. 显然, 原始条件是不可更改的硬性条件, 而中间条件是可调节条件. 由第 4.3.2 节–第 4.3.5 节得知, 每次扩展后都需要判断扩展结果的相容性, 即将得到的寄存器差分值与原始条件或中间条件作对比. 所以, 在 5 次扩展中, 与原始条件相容性的判断次数越多, 中间连接的成功率越低. 以前向扩展次数为 2 为例, 寄存器差分的扩展结果见表 6(此时, 前向扩展和后向扩展的前后顺序不会影响比特条件更新次数).

表 6 2 次前向扩展时寄存器差分的扩展结果

原有条件	ΔQ_{l-2}	ΔQ_{l-1}	ΔQ_l	δQ_{l+1}	$\delta RL(Q_{l+2},30)$	ΔQ_{l+3}	ΔQ_{l+4}
前向扩展				ΔQ_{l+1}	δQ_{l+2}		
前向扩展					ΔQ_{l+2}	δQ_{l+3}	
后向扩展				$\delta RL(Q_{l+1},30)$	ΔQ_{l+2}		
后向扩展			$\delta RL(Q_l,30)$	ΔQ_{l+1}			
后向扩展		$\delta RL(Q_{l-1},30)$	ΔQ_l				

所以, 当扩展次数为 2 时, 判断与原始条件的相容性的次数为 3. 分析前向扩展次数为 0, 1, 2, 3, 4, 5 时扩展结果的相容性, 将结果汇总在表 7 中. 结果显示: 当前向扩展次数为 2, 3, 4 时, 判断与原始条件相容性的次数较低.

表 7 原始条件相容性的判断次数汇总

前向扩展次数	0	1	2	3	4	5
原始条件相容性的判断次数	4	3	3	3	3	4

- 指标 3: 候选集合的正确率

前两个指标分析了前向扩展和后向扩展在 5 次连接中的次数与中间连接的成功率的关系, 而没有涉及前向扩展和后向扩展的顺序问题, 下面分析 5 次连接的前后关系对成功率的影响.

中间连接过程实际上是从寄存器差分、消息差分、布尔函数差分的候选集合中, 选出能够连接前向扩展和后向扩展结果的过程, 那么候选集合的正确率就成为影响中间连接的成功率的重要因素. 就两次扩展而言, 如果两次扩展的方向相反, 以先进行前向扩展、后进行后向扩展为例, 前向扩展的结果可以作为后向扩展的已知条件和判断依据, 所以在进行后向扩展时, 可以将与前向扩展结果相矛盾的候选种子筛除掉, 这种提前终止的方法在缩小候选集合范围的同时, 还提高了候选集合的正确率, 进而减少错误路径的搜索. 如在第 4.3.2 节中, 在进行第 1 次后向扩展后得到差分值$\delta RL(Q_{l+1},30)$, 此时判断该值与上一次扩展(也就是第 1 次前向扩展)的结果ΔQ_{l+1}的一致性, 这一判断过程实际上就是减少错误路径搜索的过程.

相反地, 如果连续进行两次相同方向的扩展, 则第 2 次扩展只是在第 1 次扩展的基础上又扩展了一次, 而没有筛选的过程, 所以其候选集合较前一种情形的候选集合更大, 不能保证正确率的提高. 所以, 连续两次方向相反的扩展可以提高中间连接的成功率.

综合 3 个指标, 选择最优的中间连接方法. 虽然从比特条件的更新次数来看, 5 次后向扩展的更新次数最少, 但除了第 1 次后向扩展外, 其余每次扩展之后都有判断原始条件相容性的过程, 这会大大降低中间连接的成功率. 根据第 3 个指标, 本文采取的策略是前 4 次扩展为两次前向扩展和两次后向扩展交叉进行. 至于最后一次扩展, 无论选择前向扩展还是后向扩展, 扩展后都是判断其与原始条件的相容性. 考虑到比特条件更新次数的影响, 选择后向作为最后一次的扩展方向. 最终将前向扩展的次数确定为 2, 后向扩展的次数为 3. 此外, 在本文的中间连接算法中, 比特差分ΔQ_{l+1}和ΔQ_{l+2}分别是在第 1 次前向扩展和第 1 次后向扩展过程中确定的, 其取值空间 $\boldsymbol{Q}_{l+1}$和 $\boldsymbol{Q}_{l+2}$允许进位扩展, 这在一定程度上也会提高中间连接的成功率.

4.3.7 与前人的差分路径搜索算法对比分析

本文的差分路径搜索算法框架与前人方法类似, 分为前向扩展、后向扩展和中间连接这 3 个子算法. 本文的方法有 4 点改进, 具体如下.

1) 对间接比特条件的限制更少

Marc Stevens 在第 i 步前向扩展算法中, 要求比特条件 q_{i-2}, q_{i-1} 为直接条件, q_{i-3} 可为间接条件, 且当 q_{i-3} 为间接条件时, q_{i-3} 为前向条件, 只与 Q_{i-2} 相关. 在第 j 步后向扩展算法中, 要求比特条件 q_{j-3}, q_{j-2} 为直接条件, q_{j-1} 可为间接条件, 且当 q_{j-1} 为间接条件时, q_{j-1} 为后向条件, 只与 Q_{i-2} 相关. 而在本文的前向扩展和后向扩展算法中, 只要求路径中的比特条件为间接条件时相关方向与扩展的方向一致. 从差分路径的搜索空间来看, 本文算法去掉如上限制后, 搜索空间更大, 可以挖掘构造更多差分路径, 进一步探索差分路径之间的内在联系和特征.

2) 避免了中间连接过程的字长扩展

Marc Stevens 在中间连接算法中, 采用迭代的方法求解未知寄存器差分和布尔函数差分, 每次迭代求解未知量的一个比特. 对于一组前向扩展到第 l 步和后向扩展到第 l+6 步的搜索结果, 需求解的未知变量为 3 个寄存器差分ΔQ_l, ΔQ_{l+1}, ΔQ_{l+2}和 5 个布尔函数差分 FW_1, FW_2, FW_3, FW_4, FW_5, 其中, $FW_i=\delta F_{l+i}+\delta W_{l+i}$且 $1\leqslant i\leqslant 5$. 由于布尔函数某个比特的 3 个输入寄存器比特相差两个比特位, 因此共需要迭代 40 步才能得到所有比特的值, 这就产生了字长扩张, 将 32 比特的寄存器扩展到 40 比特才可以求解. 其中间连接算法的具体迭代过程总结在表 8 中, 表 8 列出了每一步求解的比特差分, U_i, i=0,…,39 为每一步求解的所有可能差分的集合. 与 Marc Stevens 的算法不同的是, 本文的中间连接算法中每次迭代以一步前向扩展或后向扩展为基本单元, 不仅与前向(后向)扩展数据结构一致, 同时避免了 32 比特到 40 比特的扩展. 虽然本文的中间连接迭代过程不是以比特为基本单位, 但在连接过程中, 同样遍历了所有可能的差分值和比特条件, 因此没有缩小路径搜索空间. 另外, 从实现的角度看, 本文的中间连接算法以前向扩展和后向扩展为基础, 更易于实现.

3) 统一的全轮差分路径构造方法.

本文提出了带比特条件的全轮差分路径构造方法, 统一了第 1 轮差分路径构造和后 3 轮的差分路径构造,

该方法既与原有第 1 轮路径构造相容, 又能省去后 3 轮路径约简、消息约简等繁琐技术环节, 具有良好的兼容性.

4) 给出中间连接最优扩展策略.

综合考虑 3 个影响中间连接成功率的相关指标, 得出了最优的中间连接扩展策略.

表 8 Marc Stevens 中间连接算法的迭代求解过程

	ΔQ_l	ΔQ_{l+1}	ΔQ_{l+2}	FW_1	FW_2	FW_3	FW_4	FW_5
$U_0 \to U_1$	$\Delta Q_l[0]$	–	–	$FW_1[0]$	–	–	–	–
$U_1 \to U_2$	$\Delta Q_l[1]$	–	–	$FW_1[1]$	–	–	–	–
$U_2 \to U_3$	$\Delta Q_l[2]$	$\Delta Q_{l+1}[0]$	–	$FW_1[2]$	$FW_2[0]$	–	–	–
$U_3 \to U_4$	$\Delta Q_l[3]$	$\Delta Q_{l+1}[1]$	–	$FW_1[3]$	$FW_2[1]$	–	–	–
$U_4 \to U_5$	$\Delta Q_l[4]$	$\Delta Q_{l+1}[2]$	$\Delta Q_{l+2}[0]$	$FW_1[4]$	$FW_2[2]$	$FW_3[0]$	–	–
$U_5 \to U_6$	$\Delta Q_l[5]$	$\Delta Q_{l+1}[3]$	$\Delta Q_{l+2}[1]$	$FW_1[5]$	$FW_2[3]$	$FW_3[1]$	–	–
$U_6 \to U_7$	$\Delta Q_l[6]$	$\Delta Q_{l+1}[4]$	$\Delta Q_{l+2}[2]$	$FW_1[6]$	$FW_2[4]$	$FW_3[2]$	$FW_4[0]$	–
$U_7 \to U_8$	$\Delta Q_l[7]$	$\Delta Q_{l+1}[5]$	$\Delta Q_{l+2}[3]$	$FW_1[7]$	$FW_2[5]$	$FW_3[3]$	$FW_4[1]$	–
$U_k \to U_{k+1}$ $(8 \leqslant k \leqslant 31)$	$\Delta Q_l[k]$	$\Delta Q_{l+1}[k-2]$	$\Delta Q_{l+2}[k-4]$	$FW_1[k]$	$FW_2[k-2]$	$FW_3[k-4]$	$FW_4[k-6]$	$FW_5[k-8]$
$U_{32} \to U_{33}$	–	$\Delta Q_{l+1}[30]$	$\Delta Q_{l+2}[28]$	–	$FW_2[30]$	$FW_3[28]$	$FW_4[26]$	$FW_5[24]$
$U_{33} \to U_{34}$	–	$\Delta Q_{l+1}[31]$	$\Delta Q_{l+2}[29]$	–	$FW_2[31]$	$FW_3[29]$	$FW_4[27]$	$FW_5[25]$
$U_{34} \to U_{35}$	–	–	$\Delta Q_{l+2}[30]$	–	–	$FW_3[30]$	$FW_4[28]$	$FW_5[26]$
$U_{35} \to U_{36}$	–	–	$\Delta Q_{l+2}[31]$	–	–	$FW_3[31]$	$FW_4[29]$	$FW_5[27]$
$U_{36} \to U_{37}$	–	–	–	–	–	–	$FW_4[30]$	$FW_5[28]$
$U_{37} \to U_{38}$	–	–	–	–	–	–	$FW_4[31]$	$FW_5[29]$
$U_{38} \to U_{39}$	–	–	–	–	–	–	–	$FW_5[30]$
$U_{39} \to U_{30}$	–	–	–	–	–	–	–	$FW_5[31]$

5 实验结果与验证

5.1 实验结果

采用第 4 节的算法搜索 SHA-1 的第 1 轮差分路径. 在进行前向扩展算法之前, 需要确定算法具体的输入内容, 即 $\delta RL(Q_{-4},30)$, $(\Delta Q_t)_{t=-3}^{0}$, $(\hat{q}_t)_{t=-3}^{0}$, 此处将初始化内容设定为与 Marc Stevens 的第 1 块碰撞消息采用的差分路径的第 0 步寄存器状态一致. 在进行后向扩展算法之前, 需要确定具体的输入内容为 $(\Delta Q_t)_{t=16}^{19}$, δQ_{20}, $(\hat{q}_t)_{t=16}^{19}$, 此处将初始化内容同样设定为与 Marc Stevens 的第 1 块碰撞消息采用的差分路径的第 19 步寄存器状态一致, 搜索得到的差分路径见表 9. 表 9 中第 2 列列出了差分路径在每一步上寄存器的比特差分或(循环移位后的)差分值, 具体地: 当 $t=-4$ 时, 表 9 第 2 列的值为循环移位差分值$\delta RL(Q_t,30)$; 当$-3 \leqslant t \leqslant 19$ 时, 表格的第 2 列列出了比特差分ΔQ_t; 当 $t=20$ 时, 表格中第 2 列的值为差分值δQ_t. 表 9 第 3 列列出了每一步的布尔函数输出差分情况, 第 4 列列出了每一步上的消息差分, 第 5 列则列出了每一步寄存器上的 32 位比特条件.表中除寄存器上的(循环移位)差分值和比特条件外, 每个数字均代表差分不为零的比特位(若前面有负号则代表是负差分, 否则为正差分), "——"代表 32 位全零差分. 例如, ΔF_3 在表中对应为"4, –9, –10, 14, –16, 26, 30, 31", 则ΔF_3 在第 4 位、第 9 位、第 10 位、第 14 位、第 16 位、第 26 位、第 30 位、第 31 位上存在非零差分, 且在第 9 位、第 10 位、第 16 位上为负差分, 在第 4 位、第 14 位、第 26 位、第 30 位、第 31 位为正差分; 而ΔF_1 在表中对应为"——", 这代表ΔF_1 的 32 位上差分均为 0.

将前向扩展和后向扩展的初始化内容分别设定为与 Marc Stevens 的第 2 块碰撞消息采用的差分路径的第 0 步和第 19 步寄存器状态一致, 进行与表 9 差分路径相同过程的搜索, 得到附录 3 的差分路径 II.

表 9 差分路径 I

t	$\delta RL(Q_t,30)$或ΔQ_t或δQ_t	ΔF_t	δW_t	q_t
–4	0			....
–3	——			....
–2	——			....
–1	——			···1···· .... ····0···
0	——	——	1, 26, 27	·^·0·1·· ·····0·1 ···00·10 ·1··1··1
1	1, –26, 28	——	4, –30, 31	·0·+^–^^ ^^^^^1^0 ^^^11^10 ·0··1·+0

表 9 差分路径 I(续)

t	$\delta RL(Q_t,30)$或ΔQ_t或δQ_t	ΔF_t	δW_t	q_t
2	1, −4, −6~−25, 26, −30	1, 26	−2, 3, −4, −26, 28, 29, 31	1—···+—— ———————— ———————— ——·—1·+0
3	−1, −4, −6, 7, 9, 10, −28, −30	4, −9, −10, 14, −16, 26, 30, 31	−2, 26, 27, 28, −29	·—·—·0·1 11111111 ····0++· +−1−00−0
4	1, 6~10, −11, −30	−4~−10, −12~−23, 24, 26, −28	1, −3, 4, 26, −27, −28, −29, 31	·—···1·0 11111111 1111−+++ ++0·1·+1
5	1, 4, 6, −30	−1, −2, −4, −5, −9, −11, 24, −26, −28, 31	−4, −29	·····0·· ····· ·····0· ·+·+10+0
6	1, −4, 28, −30	1, −2, 5, 7, 8, −26, −28, −30, 31	2, 3, 4, 26, −29	·—·+···· ····· ·····01 100—·0+·
7	−31	4, 5, 6, −9, −28, 30, 31	2, 4, −26, −27, 29, 30, 31	−1···1·· ····· ····· ···0·0··
8	−28	2, 4, 26, −28, 31	−1, 26, −27	1·1—·1·· ····· ····· ·····1··
9	−29	−28, −29	−4, 30, 31	··—··0·· ····· ····· ·····
10	——	−29	−2, −3, 4, −26, −28, 29, 31	^····· ····· ····· ·····1
11	−29	−26	2, −26, 27, −29	···—·1··· ····· ····· ·····0
12	−30	——	−3, 4, −26, −27, −28, 29, 31	0—···1··· ····· ····· ·····!·
13	31	−27, 30	−4, 28, 29, 31	+··01··· ····· ····· ·····
14	−29	31	−2, 3	···—1···· ····· ····· ·····!·
15	31	——	−4, −27, −28, −29, 31	+·0·1··· ····· ····· ·····!^
16	−30, 31	−27, 29, 31	3, −4, −27	+−0·0··· ····· ····· ·····!·
17	31	−27, 31	−4, −27, −28, −29, 30	+··1···· ····· ····· ·····^·
18	29, −31	−28, 29, 31	−2, 4, −27	−·+0···· ····· ····· ·····
19	−31	−28, 29	4, 28, 29, −30	−····· ····· ····· ·····
20	2^{29}			

5.2 差分路径的正确性验证

接下来验证表 9 中的路径是否真实有效. 本文将从两个方面验证差分路径的正确性: (1) 验证布尔函数差分与比特条件之间的关系; (2) 验证每一步上各项差分是否满足迭代方程.

首先验证布尔函数差分与比特条件之间的关系. 该步验证包含两个方面的内容: 一是验证路径已指定的布尔函数差分能否在寄存器差分给定的情况下出现, 即布尔函数差分的成立性; 二是布尔函数输出差分给定时, 查验所需的比特条件与路径中的比特条件是否一致. 在验证第 t 步的布尔函数$\Delta F_t=\Delta F_t(Q_{t-1},Q_{t-2},Q_{t-3})$时, 逐比特按照本文第 3.2 节的方法进行分析: 对于正整数 $0\leqslant b\leqslant 31$ 和差分路径上 3 个输入寄存器比特上的差分 $(\Delta Q_{t-1}^P[b],\Delta Q_{t-2}^P[b+2 \bmod 32],\Delta Q_{t-3}^P[b+2 \bmod 32])$ (本节中, 右上标为 P 的符号均代表差分路径给出的变量取值), 对照表 2-1 可得该差分向量下$\Delta F_t[b]$的取值集合 $V_{t,b}^P$, 判断是否有 $\Delta F_3^P[i]\in V_{t,b}^P$.

然后, 验证比特条件是否一致, 在前步判断通过的情况下, 查看 $\Delta F_t^P[b]$ 取值下所需的比特条件是否与 $(q_{t-1}^P[b],q_{t-2}^P[b+2 \bmod 32],q_{t-3}^P[b+2 \bmod 32])$ 一致. 经过查验, 表 9 差分路径全部通过该步验证.

然后验证每一步上各项差分是否满足 SHA-1 的迭代方程. 第 i 步需要满足方程:

$$\delta Q_{i+1}=\sigma(RL(\Delta Q_i,5))+\sigma(\Delta F_i)+\sigma(RL(\Delta Q_{i-4},30))+\delta W_i.$$

以第 10 步为例, 等式中的各项差分取值见表 10.

表 10 第 10 步的各项差分取值

δQ_{11}	$\sigma(RL(\Delta Q_{10},5))$	ΔF_{10}	$\sigma(RL(\Delta Q_6,30))$	ΔW_{10}
−29	——	−29	31, −2, 26, −28	−2, −3, 4, −26, −28, 29, 31

因为$-2^{29} \bmod 2^{32}=-2^{29}+2^{31}-2^2+2^{26}-2^{28}-2^2-2^3+2^4-2^{26}-2^{28}+2^{29}+2^{31}$, 所以第 10 步的迭代方程成立. 采用相同方法, 验证所有 20 步迭代方程是否成立. 经过验证, 表 8 差分路径满足所有迭代方程.

综上, 表 9 中的差分路径是有效的, 可以用于后续的碰撞寻找过程. 另外, 附录 3 中的差分路径也可以通过本节验证. 从结果来看, 本文的算法是有意义的.

6 总 结

本文分析了 SHA-1 的布尔函数输出和比特条件之间的关系, 提出了带比特条件的差分路径, 可以更细致地刻画消息对在 SHA-1 算法中形成的各项差分, 且具有良好的兼容性. 进一步, 针对带比特条件的差分路径, 提出了相关的前向扩展、后向扩展和中间连接算法. 其中, 前向扩展和后向扩展算法过程简单易于实现. 提出了与中间连接的成功率相关的 3 个指标, 根据相关结论, 选择最优的中间连接算法. 结果表明: 按照本文提出的算法, 可以搜索得到正确的真实有效的差分路径. 这些基础算法可以为碰撞攻击过程中接下来的部分提供保障. 从对 SHA-1 碰撞攻击的过程来看, 构造概率高的差分路径有利于提高消息搜索步骤的成功率, 衡量差分路径成立概率的重要因素之一为比特条件的个数. 而在本文的差分路径构造算法中, 中间连接过程将比特条件的更新次数(指标 1)纳入考量范围, 即本文的中间连接策略选取了比特条件更新次数最少的扩展顺序, 进一步搜索得到的差分路径的概率较高, 对碰撞攻击的后续流程有积极作用.

References:

[1] NIST. FIPS 180-1: Secure hash standard. 1995. https://nvlpubs.nist.gov/nistpubs/Legacy/FIPS/fipspub180-1.pdf

[2] Stevens M. New collision attacks on SHA-1 based on optimal joint local-collision analysis. In: Johansson T, Nguyen PQ, eds. Proc. of the Advances in Cryptology (EUROCRYPT 2013). Berlin, Heidelberg: Springer-Verlag, 2013. 245−261.

[3] Stevens M. Attacks on hash functions and applications [Ph.D. Thesis]. Leiden: Leiden University, 2012.

[4] Leurent G, Peyrin T. From collisions to chosen-prefix collisions application to full SHA-1. In: Ishai Y, Rijmen V, eds. Proc. of the Advances in Cryptology (EUROCRYPT 2019). Berlin, Heidelberg: Springer-Verlag, 2019. 527−555.

[5] Leurent G, Peyrin T. SHA-1 is a shambles: first chosen-prefix collision on SHA-1 and application to the PGP Web of trust. In: Capkun S, Roesner F, eds. Proc. of the 29th USENIX Security Symp. Berkeley: USENIX, 2020. 1839−1856.

[6] Manuel S. Classification and generation of disturbance vectors for collision attacks against SHA-1. Designs, Codes and Cryptography, 2011, 59(1): 247−263.

[7] Wang XY, Yin YL, Yu HB. Finding collisions in the full SHA-1. In: Shoup V, ed. Proc. of the Advances in Cryptology (CRYPTO 2005). Berlin, Heidelberg: Springer-Verlag, 2005. 17−36.

[8] Joux A, Peyrin T. Hash functions and the (amplified) boomerang attack. In: Menezes A, ed. Proc. of the Advances in Cryptology (CRYPTO 2007). Berlin, Heidelberg: Springer-Verlag, 2007. 244−263.

[9] Yajima J, Iwasaki T, Naito Y, *et al*. A strict evaluation on the number of conditions for SHA-1 collision search. IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences, 2009, 92-A(1): 87−95.

[10] Tang YC, Zeng G, Han WB. Classification of disturbance vectors for collision attack in SHA-1. Science China Information Sciences, 2015, 58(11): 1−10.

[11] Chabaud F, Joux A. Differential collisions in SHA-0. In: Hrawczyk H, ed. Proc. of the Advances in Cryptology (CRYPTO '98). Berlin, Heidelberg: Springer-Verlag, 1998. 56−71.

[12] Wang XY, Yu HB. How to break MD5 and other Hash functions. In: Cramer R, ed. Proc. of the Advances in Cryptology (EUROCRYPT 2005). Berlin, Heidelberg: Springer-Verlag, 2005. 19−35.

[13] Wang XY, Yao AC, Yao F. Cryptanalysis of SHA-1. In: Proc. of the Presented at the Cryptographic Hash Workshop Hosted by NIST. 2005. https://csrc.nist.gov/Events/2005/First-Cryptographic-Hash-Workshop

[14] Cannière CD, Rechberger C. Finding SHA-1 characteristics: general results and applications. In: Lai XJ, Chen KF, eds. Proc. of the Advances in Cryptology (ASIACRYPT 2006). Berlin, Heidelberg: Springer-Verlag, 2006. 1−20.

[15] Cannière CD, Mendel F, Rechberger C. Collisions for 70-step SHA-1: On the full cost of collision search. In: Adams C, Miri A, Wiener M, eds. Proc. of the Selected Areas in Cryptography. Berlin, Heidelberg: Springer-Verlag, 2007. 56−73.

[16] Chen R. New techniques for cryptanalysis of cryptographic hash functions [Ph.D. Thesis]. Haifa: The Senate of the Technion—Israel Institute of Technology, 2011.

[17] Adinetz AV, Grechnikov EA. Building a collision for 75-round reduced SHA-1 using GPU clusters. In: Kaklamanis C, Papatheodorou T, Spirakis PG, eds. Proc. of the Euro-Par 2012 Parallel Processing. Berlin, Heidelberg: Springer-Verlag, 2012. 933−944.

[18] Stevens M, Karpman P, Peyrin T. Freestart collision for full SHA-1. In: Fischlin M, Coron JS, eds. Proc. of the Advances in Cryptology (EUROCRYPT 2016). Berlin, Heidelberg: Springer-Verlag, 2016. 459–483.

[19] Stevens M, Bursztein E, Karpman P, Albertini A, Markov Y. The first collision for full SHA-1. In: Katz J, Shacham H, eds. Proc. of the Advances in Cryptology (CRYPTO 2017). Berlin, Heidelberg: Springer-Verlag, 2017. 570–596.

[20] Yajima J, Sasaki Y, Naito Y, *et al*. A new strategy for finding a differential path of SHA-1. In: Pieprzyk J, Ghodosi H, Dawson E, eds. Proc. of the Information Security and Privacy. Berlin, Heidelberg: Springer-Verlag, 2007. 45–58.

附录 1. 比特条件

表 1-1 SHA-1 的比特条件

q_t	$(Q_t[b],Q'_t[b])$ 的取值条件	直接/间接条件	前向/后向相关
·	$Q_t[b]=Q'_t[b]$	直接条件	
+	$Q_t[b]=0,\ Q'_t[b]=1$	直接条件	
–	$Q_t[b]=1,\ Q'_t[b]=0$	直接条件	
0	$Q_t[b]=Q'_t[b]=0$	直接条件	
1	$Q_t[b]=Q'_t[b]=1$	直接条件	
^	$Q_t[b]=Q'_t[b]=Q_{t-1}[b]$	间接条件	后向相关
v	$Q_t[b]=Q'_t[b]=Q_{t+1}[b]$	间接条件	前向相关
!	$Q_t[b]=Q'_t[b]=\overline{Q_{t-1}[b]}$	间接条件	后向相关
y	$Q_t[b]=Q'_t[b]=\overline{Q_{t+1}[b]}$	间接条件	前向相关
r	$Q_t[b]=Q'_t[b]=RL(Q_{t-1},30)[b]$	间接条件	后向相关
u	$Q_t[b]=Q'_t[b]=RR(Q_{t+1},30)[b]$	间接条件	前向相关
R	$Q_t[b]=Q'_t[b]=\overline{RL(Q_{t-1},30)[b]}$	间接条件	后向相关
U	$Q_t[b]=Q'_t[b]=\overline{RR(Q_{t+1},30)[b]}$	间接条件	前向相关
s	$Q_t[b]=Q'_t[b]=RL(Q_{t-2},30)[b]$	间接条件	后向相关
c	$Q_t[b]=Q'_t[b]=RR(Q_{t+2},30)[b]$	间接条件	前向相关
S	$Q_t[b]=Q'_t[b]=\overline{RL(Q_{t-2},30)[b]}$	间接条件	后向相关
C	$Q_t[b]=Q'_t[b]=\overline{RR(Q_{t+2},30)[b]}$	间接条件	前向相关

附录 2. 布尔函数比特条件

附录 2 给出了在 3 个输入比特差分$\Delta X,\Delta Y,\Delta Z\in\{\cdot,+,-\}$的所有取值情况下(共 $27=3^3$ 种), 不同布尔函数输出差分 $g=\Delta f(X,Y,Z)$和对应 3 个输入比特的比特条件, 括号中数字代表输入寄存器值的取值个数.

表 2-1 布尔函数 $f(X,Y,Z)=(X\wedge Y)\oplus(\bar{X}\wedge Z)$ 的比特条件

输入寄存器差分	前向相关			后向相关		
	g=0	g=+1	g=–1	g=0	g=+1	g=–1
···(8)	···(8)					
··+(4)	1·+(2)	0·+(2)		1·+(2)	0·+(2)	
···–(4)	1·–(2)		0·–(2)	1·–(2)		0·–(2)
·+·(4)	0+·(2)	1+·(2)		0+·(2)	1+·(2)	
·++(2)		·++(2)			·++(2)	
·+–(2)		1+–(1)	0+–(1)		1+–(1)	0+–(1)
·–·(4)	0–·(2)		1–·(2)	0–·(2)		1–·(2)
·–+(2)		0–+(1)	1–+(1)		0–+(1)	1–+(1)
·––(2)			·––(2)			·––(2)
+··(4)	+·v(2)	+10(1)	+01(1)	+^·(2)	+10(1)	+01(1)

表 2-1 布尔函数 $f(X,Y,Z)=(X\wedge Y)\oplus(\overline{X}\wedge Z)$ 的比特条件(续)

输入寄存器差分	前向相关			后向相关		
	g=0	g=+1	g=−1	g=0	g=+1	g=−1
+·+(2)	+0+(1)	+1+(1)		+0+(1)	++1(1)	
+·−(2)	+1−(1)		+0−(1)	+1−(1)		+0−(1)
++·(2)	++1(1)	++0(1)		++1(1)	++0(1)	
+++(1)		+++(1)			+++(1)	
++−(1)	++−(1)			++−(1)		
+−·(2)	+−0(1)		+−1(1)	+−0(1)		+−1(1)
+−+(1)	+−+(1)			+−+(1)		
+−−(1)			+−−(1)			+−−(1)
−··(4)	−·v(2)	−01(1)	−10(1)	−^·(2)	−01(1)	−10(1)
−·+(2)	−1+(1)	−0+(1)		−1+(1)	−0+(1)	
−·−(2)	−0−(1)		−1−(1)	−1−(1)		−1−(1)
−+·(2)	−+0(1)	−+1(1)		−+0(1)	−+1(1)	
−++(1)		−++(1)			−++(1)	
−+−(1)	++−(1)			++−(1)		
−−·(2)	−−1(1)		−−0(1)	−−1(1)		−−0(1)
−−+(1)	−−+(1)			−−+(1)		
−−−(1)			−−−(1)			−−−(1)

表 2-2 布尔函数 $f(X,Y,Z)=X\oplus Y\oplus Z$ 的比特条件

输入寄存器差分	前向相关			后向相关		
	g=0	g=+1	g=−1	g=0	g=+1	g=−1
···(8)	···(8)			···(8)		
··+(4)		·u+(2)	·U+(2)		r·+(2)	R·+(2)
··−(4)		·U−(2)	·u−(2)		R·−(2)	r·−(2)
·+·(4)		·+c(2)	·+C(2)		s+·(2)	S+·(2)
·++(2)	·++(2)			·++(2)		
·+−(2)	·+−(2)			·+−(2)		
·−·(4)		·−C(2)	·−c(2)		S−·(2)	s−·(2)
·−+(2)	·−+(2)			·−+(2)		
·−−(2)	·−−(2)			·−−(2)		
+··(4)		+·u(2)	+·U(2)		+r·(2)	+R·(2)
+·+(2)	+·+(2)			+·+(2)		
+·−(2)	+·−(2)			+·−(2)		
++·(2)	++·(2)			++·(2)		
+++(1)		+++(1)			+++(1)	
++−(1)			++−(1)			++−(1)
+-·(2)	+-·(2)			+-·(2)		
+−+(1)			+−+(1)			+−+(1)
+−−(1)		+−−(1)			+−−(1)	
−··(4)		−·U(2)	−·u(2)		−R·(2)	−r·(2)
−·+(2)	−·+(2)			−·+(2)		
−·−(2)	−·−(2)			−·−(2)		
−+·(2)	−+·(2)			−+·(2)		
−++(1)			−++(1)			−++(1)
−+−(1)		−+−(1)			−+−(1)	
−−·(2)	−−·(2)			−−·(2)		
−−+(1)		−−+(1)			−−+(1)	
−−−(1)			−−−(1)			−−−(1)

表 2-3 布尔函数 $f(X,Y,Z)=(X\wedge Y)\vee(Z\wedge(X\vee Y))$的比特条件

输入寄存器差分	前向相关			后向相关		
	g=0	g=+1	g=−1	g=0	g=+1	g=−1
···(8)	···(8)			···(8)		
··+(4)	·u+(2)	·U+(2)		r·+(2)	R·+(2)	
··−(4)	·u−(2)		·U−(2)	r·−(2)		R·−(2)
·+·(4)	·+c(2)	·+C(2)		s+·(2)	S+·(2)	
·++(2)		·++(2)			·++(2)	

表 2-3 布尔函数 $f(X,Y,Z)=(X\wedge Y)\vee(Z\wedge(X\vee Y))$的比特条件(续)

输入寄存器差分	前向相关			后向相关		
	g=0	g=+1	g=−1	g=0	g=+1	g=−1
·+−(2)	·+−(2)			·+−(2)		
·−·(4)	·−c(2)		·−C(2)	s−·(2)		S−·(2)
·−+(2)	·−+(2)			·−+(2)		
·−−(2)			·−−(2)			·−−(2)
+··(4)	+·v	+·y(2)		+^·(2)	+!·(2)	
+·+(2)		+·+(2)			+·+(2)	
+·−(2)	+·−(2)			+·−(2)		
++·(2)		++·(2)			++·(2)	
+++(1)		+++(1)			+++(1)	
++−(1)		++−(1)			++−(1)	
+−·(2)	+−·(2)			+−·(2)		
+−+(1)		+−+(1)			+−+(1)	
+−−(1)			+−−(1)			+−−(1)
−··(4)	−·v(2)		−·y(2)	−^·(2)		−!·(2)
−·+(2)	−·+(2)			−·+(2)		
−·−(2)			−·−(2)			−·−(2)
−+·(2)	−+·(2)			−+·(2)		
−++(1		−++(1)			−++(1)	
−+−(1)			−+−(1)			−+−(1)
−−·(2)			−−·(2)			−−·(2)
−−+(1)			−−+(1)			−−+(1)
−−−(1)			−−−(1)			−−−(1)

附录 3. 差分路径 II

表 3-1 差分路径 II

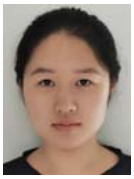
t	$\delta RL(Q_t,30)$或ΔQ_t或δQ_t	ΔF_t	δW_t	q_t
−4	-2^1			11001110 00101001 01101001 111011−1
−3	3			01111011 00011111 10101100 1101+001
−2	——			10101111 00100001 01100100 01010111
−1	−4, 5, 7, −8			11111111 11101101 0101001− +1+−0010
0	−1, −2, −4, 5, −9, −10, 11, −12	−5	1, 26, 27	10001101 01100100 110−+−−0 00+−0−−1
1	−1, 2, −5, 6, 8, 14, 17, 18, −19, −26, −28, 29, 31	5, −10, −11	4, −30, −31	+0+−0−10 110·−++1 1+1··00+ 0+−00+−0
2	−2, −3, 4, −7, 8, −11, 13, −19, −20, −21, −30	−2, 3, 5, −6, −14, −17, 26, 28, −31	−2, −3, −4, 26, 28, −29, −31	0−010110 01−−−010 01+1−11+ −1·+−−0
3	6, −12, 13, 14, 15, 24	−0, 4, 6, −8, −11, 12, −17, 19, −21, −26, 30, −31	−2, −26, 27, −28, 29	1010001+ 01000001 +++−0111 1+001000
4	−0, −4, 6, −8, −12, −13, −14, −15, −16, −17, −18, −19, −20, −21, −22, −23, −24, −25, 26, −28	0, 4, 6, −9, −13, −14, 15, −17, −26, 27	1, −3, 4, 26, 27, 28, 29, 31	·00−1+−− −−−−−−−− −−−−·01− 1+1−·11−
5	−2, 6, 7, 30	4, 6, 11, 12, −14, 15, 24, −26	4, 29	·+000110 00110010 00100··· ++···−10
6	0, 4, 5, −30	−2, 4, −6, −10, 11, 12, −13, −17, −20, −21, 22, −26	2, 3, 4, 26, −29	0−00·0·1 10111111 110001···00++·0·+
7	0, 5, 28, −30	−0, −2, 4, −6, −11, −12, −13, −22, −26	−2, −4, −26, −27, −29, -30, 31	·−0+00·· ····· ····· 0·−001·+
8	0, −5, 28	−0, 2, 4	−1, −26, 27	···+·0·· ····· ····· ···−··1·+
9	−0, −30	0, 3, −28, 30	−4, 30, 31	·−·1···· ····· ····· ····1·−−
10	——	−3, 26, 30	2, 3, −4, −26, 28, 29, 31	^1·1·1·· ····· ····· ····0···
11	29	−3, −28, −30	−2, 26, −27, 29	·0+1···· ····· ····· ·····
12	——	−30	−3, 4, 26, 27, 28, −29, 31	1···0··· ····· ····· ·····!·
13	31		−4, 28, −29, 31	+···1··· ····· ····· ·····

表 3-1 差分路径 II(续)

t	$\delta RL(Q_t,30)$或ΔQ_t或δQ_t	ΔF_t	δW_t	q_t
14	29	(+或–)31	2, –3	··+····· ····· ····· ·····!··
15	–31	无差分	4, 27, 28, 29, 31	–·1·1··· ····· ····· ·····^^
16	30, –31	27, (+或–)31	–3, 4, –27	–+0·0··· ····· ····· ·····!··
17	–31	27	4, 27, 28, –29, 30	–···1···· ····· ····· ·····^·
18	–29, –31	28, –29, (+或–)31	2, 4, 27	–·–0···· ····· ····· ·····
19	–31	28, –29	4, –28, –29, –30	–···S··· ····· ····· ·····
20	–29, –31	27, (+或–)31	–2, 3, 4, –27, –28, 29, 31	–·–·r··· ····· ····· ·····
21	——	–27, –29, (+或–)31	27, –29, 30, 31	····s··· ····· ····· ·····
22	–29	+27	2, 28, 29, 31	···–····· ····· ····· ·····
23	$\delta Q_{23}=-2^{31}$			

注: 表中各项的含义与表 8 一致, “(+或–)”表示该差分可为正差分也可以是负差分

曾光(1980－), 男, 博士, 副教授, 主要研究领域为密码学与算法分析, 密码工程技术.



李婧瑜(1996－), 女, 硕士生, 主要研究领域为 Hash 函数攻击技术.

杨阳(1980－), 女, 博士, 副教授, 主要研究领域为密码学与算法分析, 密码算法设计.