RECTANGLE-80 的相关密钥差分分析

王沙沙[1,2], 张文涛[1,2], 向泽军[1,2]

[1] 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 中国 100093
[2] 中国科学院大学网络空间安全学院 北京 中国 100049

摘要 轻量级分组密码 RECTANGLE 采用 SPN 结构, 分组长度是 64 比特, 密钥长度是 80 或 128 比特, 迭代轮数是 25 轮。其采用比特切片技术, 在软硬件实现方面均有很好的性能。本文以 Matsui 和 Moriai 等人的自动化搜索算法为基础, 采用包珍珍等人提出的 2 种优化策略, 对 RECTANGLE-80 版本进行相关密钥差分分析。我们对最窄点处的密钥状态差分进行限制, 使最窄点密钥状态差分的汉明重量取值范围分别属于区间[1,1], [1,2], [1,3], [1,4], [1,5]五种情况, 目的是求得此五种情况下前 9 轮相关密钥差分最大概率及其对应的路径。我们获得了此 5 种情况前 8 轮的最大概率及其对应的路径, 前 2 种情况 9 轮最大概率及其对应路径和后 3 种情况 9 轮最大概率的上界。以上 5 种情况的结果显示, 当取值范围属于后三种情况时, 前 8 轮的最大概率是相同的, 由此说明随着取值范围的扩大, 最大概率趋向稳定。当最窄点密钥状态差分的汉明重量取值范围属于[1,1]或[1,2]时, 9 轮的最大概率为 2^{-42}。当取值范围分别是[1,3], [1,4]和[1,5]时, 9 轮最大概率的上界分别是 2^{-41}, 2^{-37}, 2^{-34}。我们预测 9 轮最大概率的上界是 2^{-41}, 由此可以预测 18 轮的最大概率的上界是 2^{-82}, 从而 RECTANGLE-80 可以抵抗相关密钥差分分析。这是目前 RECTANGLE 抵抗相关密钥密码分析安全性评估最好结果。

关键词 轻量级分组密码; RECTANGLE; 相关密钥差分分析; 自动化搜索; 差分特征
中图法分类号 TP309.7 DOI 号 10.19363/J.cnki.cn10-1380/tn.2019.07.07

The Related-Key Differential Cryptanalysis of RECTANGLE-80

WANG Shasha[1,2], ZHANG Wentao[1,2], XIANG Zejun[1,2]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China

Abstract RECTANGLE is a 25-round SP-network with a 64-bit block length and a 80-bit or 128-bit seed key. It uses bit-slice technique to have good performance on both hardware and software platforms. Based on Matsui and Moriai et al's approaches and two strategies proposed by Zhenzhen Bao et al., we investigate the security of RECTANGLE against related-key differential cryptanalysis by restricting the Hamming weights of the key difference at the narrowest point to the following 5 ranges: [1,1], [1,2], [1,3], [1,4], or [1,5]. Our purpose is to obtain the best reduced-round related-key differential characteristics in RECTANGLE. As a result, we obtain the best related-key differential characteristics of the first eight rounds in the five cases, the best related-key differential characteristics of nine rounds in the first two cases, and an upper bound on probabilities of the best related-key differential characteristic of nine rounds in the last three cases. Our results show that the probabilities of the best characteristics on the first eight rounds are the same in the last three cases. Hence, with the expansion of the range on Hamming weights, the probability of the best characteristics tend to be stable. When the Hamming weights belong to [1,1] or [1,2], the probability of the best 9-round characteristic is 2^{-42}. When Hamming weights belong to [1,3], [1,4] or [1,5], the probability of the best 9-round characteristic is 2^{-41}, 2^{-37}, 2^{-34} respectively. We predict that the upper bound on probability of the best 9-round related-key differential characteristic is 2^{-41}. Therefore, the upper bound on probability of the best 18-round related-key differential characteristic is 2^{-82}, which shows that RECTANGLE-80 have enough security against related-key differential cryptanalysis.

Key words lightweight block cipher; RECTANGLE; related-key differential cryptanalysis; automatic search; differential characteristic

1 引言

在物联网的应用背景下, AES 等密码算法不能完全适用。特别是在资源受限的应用场景中(如 RFID、物联网、智能电网等), AES 等密码算法标准的实现代价过高。因此, 轻量级密码成为密码学的一个研究热

通讯作者: 张文涛, 博士, 研究员, Email: zhangwentao@iie.ac.cn
本课题得到国家自然科学基金 (No.61379138), 信息保障技术重点实验室开放基金 (No.KJ-15-003)资助。
收稿日期: 2017-11-03; 修改日期: 2018-06-06; 定稿日期: 2019-06-06

点, 此处的“轻量级”是指与传统密码算法相比实现代价更小、计算资源消耗更少。近十年来, 研究者对轻量级分组密码进行了大量的研究, 提出了许多不同种类的轻量级分组密码算法, 比如 KLEIN[1], Lblock[2], PRESENT[3], RECTANGLE[4], SKINNY[5], SIMON[6]，LED[7]等。其中, PRESENT[3]算法的硬件实现代价很低, 国际标准化组织(International Organization for Standardization, ISO)和国际电工委员会(International Electrotechnical Commission, IEC)于 2012 年将其定为轻量级分组密码标准。随着对轻量级密码算法应用场景的进一步认识, 密码学研究者们发现: 对于轻量级密码算法, 软件实现性能也是一个重要指标。在一些低端微处理器上, 包括 AVR、MSP、ARM 等, 软件实现比硬件实现更加灵活、成本更低, 对算法进行修改更加容易。2013 年, 美国国家安全局(National Security Agency, NSA)提出了轻量级分组密码 SIMON 和 SPECK[6], 他们宣称这两个密码算法兼顾了软件和硬件实现性能; 2014 年, 张文涛等人[4]提出了 RECTANGLE 密码算法, 此算法采用比特切片技术, 软件和硬件的实现性能均较好, 可在多平台快速实现。

对于一个密码算法, 安全性是第一位的, 因此对密码算法的安全性分析和评估至关重要。1990 年, Biham 和 Shamir 针对 DES 算法提出一种选择明文攻击方法即差分分析[8,9]。差分分析是最有效的密码分析方法之一, 其基本思想是通过分析明文差分对密文差分的影响来恢复部分密钥。之后, 差分分析方法衍生出许多变形, 如不可能差分分析[10]、截断差分分析[11]、高阶差分分析[12]等, 这些方法被广泛地应用至各种密码算法的安全性分析中。1994 年, Matsui 提出一种自动化搜索方法[13-15]对 DES 进行差分分析, 其目的是找到 DES 的最优差分特征(即差分概率最大)。Matsui 的方法是一种分支限界的深度优先搜索, 可提前排除概率小的差分特征, 极大地提高了穷举搜索的效率, 使得找到 16 轮 DES 的最优差分特征成为可能。但 Matsui 的算法应用于 FEAL[16]时, 搜索效率非常低。考虑到对于 Feistel 结构的密码算法, 搜索算法的时间复杂度由前两轮需要进行搜索的特征的数量主导, Moriai 和 Aoki 等人提出了搜索模式的概念和预搜索方法[17,18], 其方法在搜索 FEAL 时极大地提高了搜索效率。2015 年, 包珍珍等人[19]在 Moriai 等人算法的基础上提出 3 种优化策略, 分别是**从最窄点开始搜索、具体化并重新组合搜索模式、以最小变动次序进行搜索**, 很大程度上提高了搜索效率。

早期的安全性分析大都采用单密钥攻击模型, 而在相关密钥密码分析模型下[20], 攻击者可以利用密码算法在多个不同的相关的密钥下的明文-密文对进行密码分析, 攻击者不知道密钥, 但知道不同密钥之间的关系。相关密钥-差分攻击是相关密钥攻击环境下的差分攻击, 它需要同时考虑加密算法和密钥编排算法, 挖掘和利用密钥编排算法中的弱点, 从而恢复密钥。相关密钥密码分析与单密钥密码分析相比, 给予了攻击者更高的权限。目前对密码算法进行相关密钥差分分析, 更多的是根据密码算法本身的轮函数和密钥编排算法特性, 依据启发式的思维进行分析。原因在于一般分组密码的分组长度不会低于 64 比特, 密钥长度不会低于 80 比特, 穷举搜索的空间高达 2^{144}。由于搜索空间太大, 单纯的依据目前已有的搜索方法(如 Matsui 的自动化搜索算法或利用混合整数线性规划技术)对一些密码算法而言, 无法算出结果。文献[21-25]对 AES、DES、PRESENT 的分析均是基于 Matsui 自动化搜索算法再根据算法自身特性优化改进: 文献[21-23]对 AES 的分析结果是面向字节分析的, 且分别获得了 AES-128(共 10 轮)、AES-192(共 12 轮)、AES-256(共 14 轮)的 5 轮、11 轮、14 轮的最优相关密钥差分特征; 文献[24]对 DES(共 16 轮)的分析是面向比特的, 且获得了 15 轮最优相关密钥差分特征; 文献[25]对 PRESENT(共 31 轮)的分析是面向比特的, 并且应用了分而治之[25]的思想, 获得了 29 轮最大概率的上界, 从而证实其可以抵抗相关密钥差分攻击。文献[5]对 SKINNY 的分析采用了混合整数线性规划技术, 并且是面向字节的, 通过获得缩短轮最大概率的上界证实其可以抵抗相关密钥差分攻击。本文依据 RECTANGLE 自身的特点采用面向比特的方式进行分析。由文献[4, 26]可知, 目前 RECTANGLE 可以抵抗单密钥环境下的攻击, 然而相关密钥环境下的分析结果甚少。2015 年, 单进勇等人曾对旧版 RECTANGLE 算法提出 15 轮相关密钥区分器, 设计者之后对 RECTANGLE 算法进行了修改。

本文以 Matsui 和 Moriai 等人的搜索算法为基础对 RECTANGLE-80 进行相关密钥差分分析, 并将包珍珍等人单密钥分析下提出的 2 种优化策略(**从最窄点开始搜索、以最小变动次序进行搜索**)应用至 RECTANGLE 相关密钥差分分析来提高算法搜索的效率。我们对最窄点处的密钥状态差分进行限制, 使其汉明重量的取值范围分别属于区间[1,1], [1,2], [1,3], [1,4], [1,5]五种情况, 目的是求得此五种情况下前 9 轮 RECTANGLE 相关密钥差分的最大概率或上界。

本文组织结构如下: 1) 第二节介绍符号及变量; 2) 第三节介绍RECTANGEL的加密算法和密钥编排算法; 3) 第四节介绍本文用到的两种优化策略; 4) 第五节给出 RECTANGLE 相关密钥差分路径搜索算法; 5) 第六节是小结和讨论。

2 符号

我们在本节定义本文需要用到的符号及变量。

||: 连接符

⊕: 异或符

<<<: 左循环移位

S: 轮函数中的非线性层

P: 轮函数中的置换层

S^{-1}: 轮函数中非线性层的逆

P^{-1}: 轮函数中置换层的逆

SK: 密钥编排算法中的非线性层

PK: 密钥编排算法中的置换层

SK^{-1}: 密钥编排算法中非线性层的逆

PK^{-1}: 密钥编排算法中置换层的逆

Δx: 轮函数中 S(或 S^{-1})层输入差分

Δy: 轮函数中 P(或 P^{-1})层输入差分

Δz: 轮函数中状态与密钥异或前的差分

Δv: 80 比特密钥差分, 我们称之为密钥状态差分

Δk: 从Δv 中提取来的 64 比特信息, 我们称为子密钥差分

Δu: 生成下一轮密钥差分过程中的中间状态, 长度为 80 比特

p^r: 第 r 轮的差分概率。本文索引从 0 轮开始, 给定一个 n 轮的路径, 编号分别为第 0 轮、第 1 轮一直到第 n-1 轮

w^r: 第 r 轮的权重, $w^r = -\log_2 p^r$。由此概率越大, 权重越小。本文用权重代替概率, 求最大概率即是求最小权重, 且 $\sum_{i=0}^{n-1} w^i = -\log_2 \prod_{i=0}^{n-1} p^i$

Bw^n: n 轮的最小权重

Bwc^n: n 轮最小权重的候选值

3 算法描述

RECTANGLE 是一种基于 SPN 结构的轻量级分组密码, 其分组长度为 64 比特, 密钥长度为 80 比特和 128 比特两个版本, 迭代轮数为 25 轮, 本文只针对 80 比特的版本展开研究。64 比特的明文、密文及中间状态统称为消息状态, 80 比特的密钥称为密钥状态。

消息状态表示: 给定 64 比特的消息状态 $STATE = m_{63}||...||m_1||m_0$, 将 $STATE$ 看成是 4×16 的矩阵, 以一列为一个单元, 则 $STATE$ 可以表示成 $STATE=(col_{15}, col_{14}, ..., col_1, col_0)$, 其中 $col_i = m_{i+48}||m_{i+32}||m_{i+16}||m_i, i\in[0,15]$, 如图 1 所示。

$$STATE = \begin{pmatrix} col_{15} & \dots & col_1 & col_0 \end{pmatrix} = \begin{pmatrix} m_{15} & \dots & m_1 & m_0 \\ m_{31} & \dots & m_{17} & m_{16} \\ m_{47} & \dots & m_{33} & m_{32} \\ m_{63} & \dots & m_{49} & m_{48} \end{pmatrix}$$

图 1 消息状态

Figure 1 Cipher State

密钥状态表示: 80 比特的密钥用 V 表示, 其中 $V= v_{79}||...||v_1||v_0$, 此 80 比特可看成 5×16 的矩阵,令 $row_i = v_{i*16+15}||...||v_{i*16+1}||v_{i*16}, i\in[0,4]$, 则 $V = row_4||row_3||row_2||row_1||row_0$, 如图 2 所示。

$$V = \begin{pmatrix} row_0 \\ row_1 \\ row_2 \\ row_3 \\ row_4 \end{pmatrix} = \begin{pmatrix} v_{15} & \dots & v_1 & v_0 \\ v_{31} & \dots & v_{17} & v_{16} \\ v_{47} & \dots & v_{33} & v_{32} \\ v_{63} & \dots & v_{49} & v_{48} \\ v_{79} & \dots & v_{65} & v_{64} \end{pmatrix}$$

图 2 密钥状态

Figure 2 Key State

3.1 加密算法

算法 1 描述了 RECTANGLE 加密过程, 明文 M 先经过 25 轮的迭代运算, 再经过一步密钥加运算得到密文。由算法 1 可知, RECTANGLE 的轮函数包括三个部分, 分别是子密钥加层, 非线性层, 置换层。

1) 子密钥加层($addRoundKey$): 64 比特消息与 64 比特子密钥进行简单的按位异或运算。

2) 非线性层(S): 16 列并行经过 16 个相同的 S 盒运算, 即 $col_i' = sbox(col_i), i\in[0,15]$。

3) 置换层(P): 通过行移位来完成置换运算, 如图 3 所示。第 1 行不做任何操作, 第 2 行左循环移动 1 位, 第 3 行左循环移动 12 位, 第 4 行左循环移动 13 位。

$$\begin{pmatrix} m_{15} & \dots & m_1 & m_0 \\ m_{31} & \dots & m_{17} & m_{16} \\ m_{47} & \dots & m_{33} & m_{32} \\ m_{63} & \dots & m_{49} & m_{48} \end{pmatrix} \begin{matrix} \\ \xrightarrow{<<<1} \\ \xrightarrow{<<<12} \\ \xrightarrow{<<<13} \\ \\ \end{matrix} \begin{pmatrix} m'_{15} & \dots & m'_1 & m'_0 \\ m'_{31} & \dots & m'_{17} & m'_{16} \\ m'_{47} & \dots & m'_{33} & m'_{32} \\ m'_{63} & \dots & m'_{49} & m'_{48} \end{pmatrix}$$

图 3 行移位

Figure 3 ShiftRow Operates

算法 1. RECTANGLE 加密算法

输入: 明文 M, 种子密钥 $Seed$
输出: 密文 C
procedure *cipher_RECTANGLE*(M, $Seed$)
generateRoundKeys($Seed$)
$STATE := M$
FOR $i = 0$ TO 24 DO
    $STATE := addRoundKey(STATE, K^i)$
    $STATE := S(STATE)$
    $STATE := P(STATE)$
END FOR
$C := addRoundKey(STATE, K^{25})$

算法 2. 密钥编排算法

输入: 种子密钥 $Seed$
输出: 子密钥 $K^0, K^1, \ldots, K^{25}$
procedure *generateRoundKeys* ($Seed$)
$V^0 := Seed$
$K^0 := extract64(V^0)$
FOR $i = 0$ TO 24 DO
    $U := SK(V^i)$
    $U := PK(U)$
    $V^{i+1} := addRoundConstant(U)$
    $K^{i+1} := extract64(V^{i+1})$
END FOR

3.2 密钥编排算法

RECTANGLE-80 种子密钥的长度是 80 比特。算法 2 描述了 RECTANGLE-80 密钥编排算法, 给定种子密钥, 通过密钥编排算法可生成加密过程所需要的 26 个子密钥。密钥编排算法的一轮迭代包括 4 步, 分别是非线性层, 置换层, 与轮常数异或, 提取 64 比特密钥信息。在相关密钥差分分析下, 轮常数不改变差分, 本文简单介绍与轮常数异或这一步骤, 详细内容请参考文献[4]。

1) 非线性层(*SK*): 只对 80 比特密钥矩阵后四列中的 16 比特进行 S 盒运算, 剩余的比特在当前层不参与任何运算, 变换过程如图 4 所示。

2) 置换层(*PK*): 置换过程如下,

$Row_0' = (Row_0 <<< 8) \oplus Row_1$
$Row_1' = Row_2$
$Row_2' = Row_3$
$Row_3' = (Row_3 <<< 12) \oplus Row_4$
$Row_4' = Row_0$

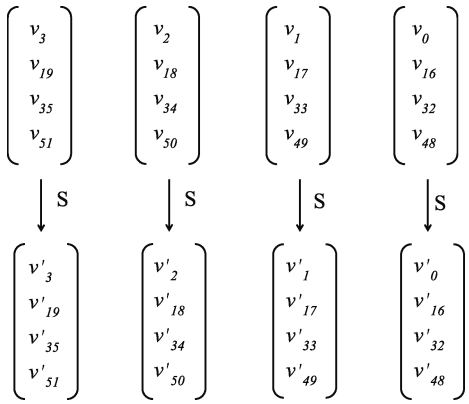

图 4 密钥编排算法的非线性层运算
Figure 4 Substitution Layer on Key Schedule

3) 与轮常数异或(*addRoundConstant*): 5 比特的密钥与 5 比特的轮常数异或, 即 $v_4'||\ v_3'||\ v_2'||\ v_1'||\ v_0' = (v_4||v_3||v_2||v_1||v_0) \oplus \mathrm{RC}[i]$, 其中 RC[$i$]表示第 i 轮的轮常量, $0 \leqslant i \leqslant 24$。

4) 提取 64 比特: 如图 5 所示, 从 80 比特密钥矩阵中提取前 4 行作为当前轮子密钥。

$$K = \begin{pmatrix} v_{15} & \ldots & v_1 & v_0 \\ v_{31} & \ldots & v_{17} & v_{16} \\ v_{47} & \ldots & v_{33} & v_{32} \\ v_{63} & \ldots & v_{49} & v_{48} \end{pmatrix}$$

图 5 子密钥状态
Figure 5 Round Key State

4 文献[19]中的两种策略

包珍珍等人[19]在 Matsui 和 Moriai 等人算法的基础上, 对自动化搜索算法提出三种优化策略。我们对 RECTANGLE 进行相关密钥差分分析时, 采用了其中的两种策略来提高算法的运行效率。这两种策略是**从最窄点开始搜索、以最小变动次序进行搜索**。定义 1 给出了搜索模式的定义, 定义 2 给出了最窄点以及最窄点权重的定义。

定义 1.搜索模式.我们称 $W^n = (w^0, w^1, \cdots, w^{n-1})$ 为一个 n 轮的搜索模式, 其中 w^i 表示第 i 轮差分的权重, $||W^n|| = \sum_{i=0}^{n-1} w^i$ 为 n 轮的总权重。

定义 2. 给定一个 n 轮的搜索模式 $W^n = (w^0, w^1, \cdots, w^{n-1})$, 令 $w^k = \mathbf{min}\{w^0, w^1, \cdots, w^{n-1}\}$。如果有多个最小值, k 取下标最小的值。第 k 轮便是最窄点,

w^k 为最窄点权重。

给定一个 n 轮搜索模式 $W^n = (w^0, w^1, \ldots, w^{n-1})$, 最窄点搜索是从最窄点处开始搜索, 最窄点的位置由给定的搜索模式确定, 可能是 n 轮中的任何一轮, 也就是说搜索的起始轮不一定在第 0 轮, 可能在中间, 也可能在最后一轮。对于一个 n 轮的模式, 假定第 k 轮是最窄点, 则从第 k 轮处开始搜索, 顺序正向搜索到第 n–1 轮。正向搜索完毕后, 开始逆向搜索。从第 k–1 轮处开始逆向搜索, 一直搜索到第 0 轮。

在单密钥差分分析下, 密钥差分为 0, 因此不用考虑密钥编排算法, 也不用考虑子密钥加层, 差分传播仅发生在轮函数中的 S 层和 P 层。图 6 描述了 RECTANGLE 单密钥差分分析下的差分传播过程。假设第 k 轮是最窄点, 先确定最窄点处一对合理的输入输出差分, 即$(\Delta x^k, \Delta x^{k+1})$。$\Delta x^{k+1}$ 作为第 k+1 轮的输入差分, 在满足搜索条件的情况下, 得到第 k+1 轮的一种输出差分Δx^{k+2}, 同理Δx^{k+2} 作为下一轮的输入差分, 重复进行, 一直搜索完第 n–1 轮, 此时正向搜索完毕。正向搜索时, S 层和 P 层同属于一轮其中 S 层在前 P 层在后, 即先搜索 S 层再搜索 P 层, 如此重复进行。逆向搜索时, 文献[19]中将 S^{-1} 层和 P^{-1} 层同样看成是一轮, 先搜索 S^{-1} 层再搜索 P^{-1} 层, 如此重复进行。由此, 逆向搜索时, Δx^k 先经过 P^{-1} 变换得到Δx^{k-1}, Δx^{k-1} 作为第 k-1 轮的逆向输入差分, 在满足搜索条件的情况下, 通过 S^{-1} 层和 P^{-1} 层获得第 k-1 轮的一种逆向输出差分Δx^{k-2}, Δx^{k-2} 作为第 k-2 轮的逆向输入差分, 重复此过程, 直到搜索完第 0 轮。

在具体实现时, S 层和 P 层可分别搜索, 也可以将 S 层与 P 层结合进行搜索, 原因在于对 S 层的搜索和对 P 层的搜索均可以通过查询对应的表和一些异或操作实现。但 S 层的表和 P 层的表可以结合成一张新表, 因此, 对 S 层和 P 层的搜索可以通过查询一张表和一些异或操作实现。这两种搜索方式获得的结果是一样的, 但是在异或次数一样的情况下, 后者会减少查表的次数, 从而提高搜索效率。因此, 作者将正向搜索时的 S 层和 P 层相结合, 其中 S 层在前 P 层在后, 反向搜索时的 S^{-1} 层和 P^{-1} 层相结合, 其中 S^{-1} 层在前 P^{-1} 层在后, 如图 7 所示, 注意反向搜索之前, Δx^k 先经过 P^{-1} 变换。当正向搜索一轮时, 以第 k+1 轮为例, 已知 S 层的输入差分Δx^{k+1}, 通过差分分布表可以获得 S 层的输出差分Δy^{k+1}, Δy^{k+1} 经过 P 变换得到当前轮的输出差分Δx^{k+2}。对于 S 层, 一个输入差分对应多种输出差分, 因此Δy^{k+1} 有多种取值, 其每种取值需要经过 P 置换。在搜索过程中会进行大量的 P 置换运算, 因此耗费许多时间。P 层输入差分的长度是 64 比特, 则有 2^{64} 种可能的输入, 若生成简单的置换表, 所需的存储空间极大, 由此不能简单地生成如此大的置换表。文献[19]中生成了一个小的置换表, 已知单个活跃 S 盒对应的输出差分, 通过查询小置换表, 求得其输出差分对应的置换层输出。已知 S 层的输出差分, 我们可知每个活跃 S 盒的输出差分, 对每个活跃 S 盒的输出差分, 通过查询小置换表, 可求得每个活跃 S 盒输出差分对应的置换层输出, 将对应的所有置换层输出异或起来获得的结果便是置换层最终的输出。由此, 已知 S 层的输出差分, 通过查询小置换表和一些异或操作可求得 P 层的输出差分。文献[19]中, 将差分分布表与小置换表复合, 生成一个新表。新表的作用是减少了查表的次数, 提升了算法的执行效率。已知 S 层的输入差分, 通过查询新表和一些异或操作, 可获得 P 层的输出差分。由于 S 层是非线性层, P 层的输出结果会有多种可能。即已知 S 层的输入差分, 可获得所有活跃 S 盒的输入差分, 对每个活跃 S 盒的输入差分查询新表, 可求得每个活跃 S 盒输入差分对应的置换层输出(有多种可能)。从每个活跃 S 盒输入差分对应的所有置换层输出中选择一种, 将选出的异或起来获得的结果就是置换层最终的一种输出结果。为了提高算法的执行效率, 在获得所有可能的置换层最终输出差分过程中, 文献[19]采用**以最小变动次序进行搜索**策略。反向搜索亦是如此, 将 S^{-1} 层与 P^{-1} 层结合起来, 如图 7 所示。

5 RECTANGLE-80 抵抗相关密钥差分密码分析的安全性评估

我们对 RECTANGLE-80 版本的相关密钥差分分析是以 Matsui 算法为基础, 使用了 Moriai 等人[19]提出

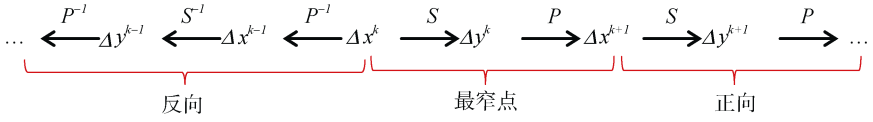

图 6 RECTANGLE 单密钥差分传播过程

Figure 6 Difference Propagation on Single-key

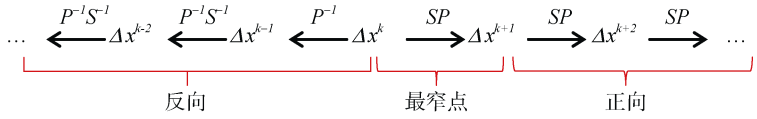

图 7　RECTANGLE 单密钥差分传播过程, S 层与 P 层复合(或 S^{-1} 与 P^{-1} 复合)

Figure 7　Difference Propagation on Single-key, S and P Combination(or S^{-1} and P^{-1} Combination)

的搜索模式的概念, 并采用文献[19]中提出的 2 个优化策略来提升算法的执行效率。Moriai 等人[18]提出的预搜索方法, 可提前存储缩减轮中无效的搜索模式(即满足 Matsui 自动化搜索算法的限制条件, 但不存在一条差分特征的搜索模式), 在具体搜索某一轮之前, 通过判定缩减轮无效的搜索模式是否是当前轮搜索模式的子模式来提前判定当前轮搜索模式是否有效, 从而提前排除一些无效的搜索模式, 提高算法的执行效率。

5.1　利用预搜索算法的困难

在对 RECTANGLE-80 进行相关密钥差分分析时, 我们对最窄点密钥状态差分的汉明重量进行了限制。因为此限制, 我们没有应用 Moriai 等人提出的预搜索算法。对于预搜索算法而言, 假如我们已经获得前 n–1 轮的最小权重, 并且已经存储了缩减轮无效的搜索模式, 在对 n 轮搜索之前, 我们先生成所有可能的 n 轮搜索模式, 每生成一个 n 轮搜索模式, 我们会检测缩短轮无效的搜索模式是否是当前搜索模式的子模式, 若是, 则舍弃当前模式, 否则保留。那么对于我们的算法, 就面临一个问题。假如我们限制最窄点密钥状态差分的汉明重量为 1, 需要搜索 7 轮的最小权重及其对应路径。在生成 7 轮的搜索模式之前, 已经存储了一个 5 轮无效的搜索模式, 例如 W^5 = (4, 3, 4, 5, 4), 显然第 1 轮为最窄点, 3 为最窄点权重值, 在对此 5 轮的搜索模式进行搜索时, 我们限制了第一轮密钥状态差分的汉明重量为 1, 在此种限制下, 我们发现此模式是无效的。给定一个 7 轮的搜索模式, 例如 W^7 = (4, 3, 4, 5, 4, 5, 2)。W^5 是 W^7 的一个子模式, 若采用预搜索方法, 我们应舍弃 W^7 模式。但是 W^7 的最窄点是第 6 轮。对于 W^7 模式而言, 只会限制第 6 轮密钥状态差分的汉明重量为 1, 其他轮不限制。但是 W^5 模式限制了第 1 轮的密钥状态差分汉明重量为 1, 由此 W^7 可能是一个合法的模式, 但被删除。基于以上原因, 我们没有在我们的算法中应用 Moriai 的预搜索算法。

5.2　RECTANGLE 相关密钥差分搜索的具体实现

Matsui 的自动化搜索算法是一种归纳递归方法, 首先得到 n 轮的最小权重, 再求 n+1 轮的最小权重。假定我们已经求得 n 轮的最小权重 Bw^n, 考虑到在相关密钥差分分析下, 子密钥差分与消息状态差分异或值可能为 0, 可能导致搜索的总轮数增加 1 轮却没有增加权重, 因此令 n+1 轮的候选权重 Bwc^{n+1}=Bw^n。我们求得所有可能的 n+1 轮搜索模式, 每一个搜索模式的权重总和为 Bwc^{n+1}。且每一个搜索模式中, 任意的连续 r 轮权重之和均满足式(1), 其中 $i\in[0,$

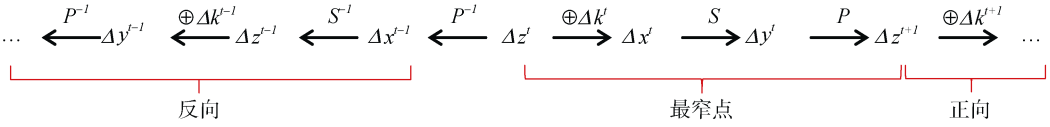

图 8-1　RECTANGLE 相关密钥差分分析下, 轮函数中差分传播过程, 其中第 t 轮为最窄点

Figure 8-1　Difference Propagation of Round Function is Based on Related-key Difference Propagation and the Narrowest Point is the t-round

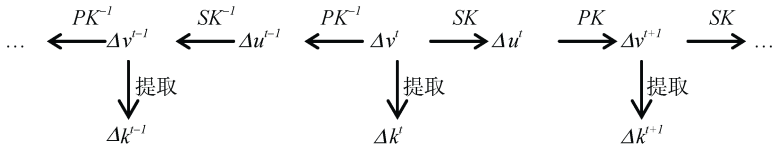

图 8-2　RECTANGLE 相关密钥差分分析下, 密钥编排算法中密钥状态差分传播过程, Δv^t 为最窄点的密钥状态差分, Δk^t 为最窄点的子密钥差分

Figure 8-2　Difference Propagation of Key Schedule is Based on Related-key Difference Propagation and Key State Difference of the Narrowest Point is Δv^t and Round Key State Difference of the Narrowest Point is Δk^t

n-r+1]。将满足(1)式的搜索模式称之为合法的搜索模式。

$$w^i + w^{i+1} + ... + w^{r+i-1} \geqslant Bw^r \quad (1)$$

对所有合法的搜索模式进行搜索, 判定是否存在一条满足模式的差分路径。若存在, 我们就能确定n+1 轮的最小权重为 Bwc^{n+1} 的值。若不存在, 则 Bwc^{n+1}= Bwc^{n+1}+1, 重复之前的操作, 一直到找到一条差分路径为止, 最终更新 Bw^{n+1} 为 Bwc^{n+1}。

在相关密钥差分分析下, 密钥的差分值非零, 因此在分析时, 我们需要考虑子密钥差分对加密算法的影响。假如存在一个搜索模式, 第 t 轮是最窄点, 图 8-1 描述了 RECTANGLE 在相关密钥差分分析下, 轮函数中差分传播过程。与图 6 中 RECTANGLE 单密钥差分传播不同之处在于每轮多了一步操作, 即与子密钥差分的异或。图 8-2 与图 8-1 对应, 其描述了密钥编排算法中密钥状态差分的传播过程, 由此最窄点的密钥状态差分为 Δv^t, 从 Δv^t 中提取 64 比特得到 Δk^t, Δk^t 是最窄点的子密钥差分。由图 8-1 和图 8-2 差分传播过程可知, 轮函数的 S 层和密钥编排算法的 SK 层为非线性操作, 此时差分传播会引入权重。对于单密钥差分分析而言, 已知一个搜索模式, 将每轮的权重分配给对应轮函数的 S 层即可。然而在 RECTANGLE 相关密钥差分分析下, 我们需要将搜索模式中的权重分配给轮函数的 S 层和密钥编排算法的 SK 层。由此我们分别给出子密钥差分权重、消息状态差分权重和相关密钥搜索模式的定义来阐明我们权重分配以及搜索过程, 见定义 3、4 和 5。由定义 3 可知, 给定子密钥差分权重, 其有多种子密钥差分权重的表现形式。给定一种子密钥差分权重的表现形式, 可唯一确定子密钥差分权重。同理, 由定义 4 可知, 给定消息状态差分权重, 其有多种消息状态差分权重的表现形式。给定一种消息状态差分权重的表现形式, 可唯一确定消息状态差分权重。

定义 3.子密钥差分权重. wk^i 表示生成第 i 轮子密钥差分时经过 SK 层产生的权重, wk_j^i 表示第 j 个 S 盒产生的权重, 其中 j∈[0,3]。我们称 wk^i 为第 i 轮子密钥差分权重, (wk_0^i, wk_1^i, wk_2^i, wk_3^i)为第 i 轮子密钥差分权重的表现形式, 由此 $wk^i = \sum_{j=0}^{3} wk_j^i$ 。

定义 4.消息状态差分权重. wr^i 表示第 i 轮加密过程中经过 S 层产生的权重, wr_j^i 表示第 j 个 S 盒产生的权重, 其中 j∈[0,15]。我们称 wr^i 为第 i 轮消息状态差分权重, (wr_0^i, wr_1^i, …, wr_{15}^i)为第 i 轮消息状态差分权重的表现形式, 由此 $wr^i = \sum_{j=0}^{15} wr_j^i$ 。

定义 5. 相关密钥搜索模式. 给定一个 n 轮的搜索模式 $W^n = (w^0, w^1, ..., w^{n-1})$, 第 i 轮的相关密钥搜索模式为 $wrel^i$ = (wk_0^i, wk_1^i, wk_2^i, wk_3^i | wr_0^i, wr_1^i, …, wr_{15}^i), 由此, $w^i = wk^i + wr^i$ 。

在搜索的过程中, 本文先生成某一轮的子密钥差分, 再进行某一轮加密算法中的轮函数运算。生成子密钥差分时, (wk_0^i, wk_1^i, wk_2^i, wk_3^i)有多种可能, 只有满足 wk^i<=w^i, 子密钥差分权重的表现形式才是合法的。在进行加密算法的轮函数运算时, 由于子密钥差分已经获得, wk^i 的值已经确定, 我们可以唯一确定消息状态差分权重 wr^i= w^i - wk^i, 从而可以获得所有合法的(wr_0^i, wr_1^i, …, wr_{15}^i)。在搜索开始时, 我们会给定最窄点的密钥状态差分, 最窄点的子密钥差分只需从最窄点的密钥状态差分中提取 64 比特信息, 没有涉及到 SK 层, 因此最窄点权重只分配给轮函数的 S 层。其他轮的权重分配给其对应轮的 S 层, 以及生成对应轮子密钥差分时经过的 SK 层。若第 t 轮是最窄点, 最窄点的相关密钥搜索模式为 $wrel^t$ =(0, 0, 0, 0 | wr_0^t, wr_1^t, …, wr_{15}^t), $w^t = \sum_{j=0}^{15} wr_j^t$ 。密钥的长度为 80 比特, 若搜索所有可能的最窄点密钥状态差分, 时间复杂度至少为 2^{80}, 以现有计算机的计算能力无法搜索出来。但 RECTANGLE-80 具有以下特性:

a. 在 RECTANGLE 单密钥差分分析的最优差分路径中, 每一轮消息状态差分的汉明重量取值比较小。

b. RECTANGLE-80 的密钥编排算法扩散性比较弱。

c. 由最窄点的定义可知, 在一个搜索模式中, 最窄点的权重最小, 由此最窄点的扩散性比较弱。

基于以上特性, 我们对最窄点的密钥状态差分加以限制, 在此限制内, 搜索所有可能子密钥差分。我们使最窄点密钥状态差分的汉明重量取值范围分别属于区间[1,1], [1,2], [1,3], [1,4], [1,5]五种情况, 此五种情况是经验选取的。随着取值范围的扩大, 最窄点密钥状态差分的搜索空间成指数增长, 当所属区间为[1,5]时, 搜索空间达到 $\binom{80}{1}+\binom{80}{2}+\binom{80}{3}+\binom{80}{4}+\binom{80}{5} \approx 2^{25}$ 。对于 SPN 结构的密码算法, 搜索算法的时间复杂度由搜索的第一轮(本文指的最窄点处)的差分特征数量主导。本文的搜索算法在开始时会遍历给定的最窄点密钥状态差分搜索空间, 每一个最窄点密钥状态差分可能对应很多最窄点的输入输出差分对, 从而一个最窄点密钥状态差分可能对应很多差分特征, 实际的时间复杂度远大于 2^{25}。由此我

们选择在这 5 个区间内进行搜索。最终的实验结果也表明, 我们选择这 5 个区间是合理的。下面我们分别讲述我们的搜索过程、轮函数和密钥编排算法差分搜索的具体实现过程。

5.2.1 搜索过程

我们的基本思想如下, 2)-4)确定最窄点的密钥状态差分、最窄点的子密钥差分以及最窄点的输入输出差分, 5)-6)为正向搜索的思想, 7)为逆向搜索的思想。

1) 已知 Bwc^{n+1}, 生成所有满足条件的搜索模式。

2) 给定最窄点的密钥状态差分, 使得密钥状态差分的汉明重量不超过某个值。我们假定不超过 3, 则密钥状态差分取值有 $\binom{80}{1}+\binom{80}{2}+\binom{80}{3}$ 种。其中, $\binom{n}{m}=n!/(m!)(n\text{-}m)!$。

3) 对于每一个最窄点的密钥状态差分, 对所有模式按照**从最窄点开始搜索**策略进行搜索。对于给定的一个最窄点密钥状态差分, 下面以一个搜索模式为例来说明具体的搜索。例如, 我们给定了一个最窄点的密钥状态差分, 对某一个搜索模式 $W^m=(w^0, w^1, \cdots, w^{n-1})$进行搜索, 假如此搜索模式的第 t 轮是最窄点, 则 w^t 为最窄点权重, 给定的最窄点密钥状态差分便是第 t 轮的密钥状态差分, 从而也可获得第 t 轮的子密钥差分。

4) 将 w^t 分配给轮函数的 S 层, 即 $wr^t=w^t$。由第 4 节可知, 通过查表和一些异或操作可求的所有可能的最窄点 S 层输入差分和 P 层输出差分$(\Delta x^t, \Delta z^{t+1})$, 且每一对差分的权重均为 w^t, 获得所有差分对的过程中采用最小变动策略。3)中已经求的最窄点的子密钥差分Δk^t, 每获得一个差分对$(\Delta x^t, \Delta z^{t+1})$, 由于$\Delta z^t=\Delta x^t \oplus \Delta k^t$, 可唯一确定最窄点的一个输入输出差分$(\Delta z^t, \Delta z^{t+1})$。每确定一个最窄点的输入输出差分, 进行第 5)步。我们用变量 r 表示目前正在进行搜索的轮, 则 r=t。

5) 获得下一轮的子密钥差分。已知第 r 轮的密钥状态差分Δv^r, 生成第 r+1 轮的密钥状态差分Δv^{r+1}, 通过提取, 获得第 r+1 轮的子密钥差分Δk^{r+1}。密钥编排算法中存在 SK 层, 因此Δv^{r+1} 有多种取值, 同理 Δk^{r+1} 有多种取值。获得每一个合法的Δv^{r+1} 取值时, 需保证 $\sum_{j=0}^{3} wk_j^{r+1} \leqslant w^{r+1}$。从$\Delta v^{r+1}$ 中提取 64 比特密钥信息得到Δk^{r+1}。Δk^{r+1} 与Δz^{r+1} 异或, 得到第 r+1 轮 S 层的输入差分Δx^{r+1}。

6) 轮函数搜索。已知第 r+1 轮 S 层的输入差分, 得到第 r+1 轮 P 层的所有可能的输出差分。获得每一个合法的输出差分时, 需满足 $\sum_{j=0}^{15} wr_j^{r+1}=w^{r+1}-\sum_{j=0}^{3} wk_j^{r+1}$。若 r+1=n-1, 说明正向搜索完毕, 下面进入反向搜索 7)。否则令 r=r+1, 重复执行 5)和 6)。

7) 反向搜索与正向搜索的思想是一样的, 在搜索某一轮之前, 先生成当前轮需要的子密钥差分, 并且生成子密钥差分时的权重不能大于搜索模式中给定的当前轮权重。剩余的权重分配给当前轮的 S 层。具体搜索过程, 我们将在 5.2.3 节详细说明。

5.2.2 轮函数中差分搜索的具体实现

在我们对 RECTANGLE 抵抗相关密钥差分密码分析的评估算法中, 轮函数中差分搜索的具体实现与文献[19]中单密钥差分分析的实现相似, 但我们的算法中多了子密钥异或的步骤, 因此对轮函数差分搜索的实现可以直接应用文献[19]的方法, 只需要稍加改动即可, 改动方法如下: 如图 7 所示, 在单密钥差分下, S 层和 P 层可以结合在一起实现, 给定 S 层的输入差分, 通过查表可以得到所有可能的 P 层的输出差分。同理, 在相关密钥差分分析下, 我们在具体搜索的时候, 也将 S 层和 P 层合并, 以提高搜索算法的执行效率。如图 8-1 所示, 从图中可以看出, 正向搜索时 S 层和 P 层衔接, 因此 S 层与 P 层可以结合在一起。给定 S 层的输入, 通过查表得到 P 层的输出。P 层的输出差分与下一轮子密钥差分进行异或得到下一轮 S 层的输入差分, 如此往复进行正向搜索, 直到正向搜索完毕。在图 7 中, 反向搜索时, 文献[19]将 S^{-1} 和 P^{-1} 层结合起来。然而在图 8-1 中, 反向搜索时, 当 S^{-1} 层在前 P^{-1} 层在后时, S^{-1} 层和 P^{-1} 层没有衔接, 求得 S^{-1} 层的输出差分, 需要先与子密钥差分异或, 得到的结果再进行 P^{-1} 层置换。这样的话, 反向搜索时, S^{-1} 层和 P^{-1} 层需分开来做。我们为了在实现的时候提高算法运行效率, 按照如下方式, 将 S^{-1} 层和 P^{-1} 层合并。由于异或是线性操作, 因此有等式(2)成立, Δx 是 S^{-1} 层输入, Δz 是 S^{-1} 层的输出, Δy 是 p^{-1} 层输入, Δk 是子密钥差分。由下面的式子:

$$\begin{aligned} & P^{-1}(\Delta y) \\ &= P^{-1}(\Delta z \oplus \Delta k) \\ &= P^{-1}(\Delta z) \oplus P^{-1}(\Delta k) \\ &= P^{-1}(S^{-1}(\Delta x)) \oplus P^{-1}(\Delta k) \end{aligned} \tag{2}$$

可以将 S^{-1} 层和 P^{-1} 层合并, 即 $P^{-1}(S^{-1}(\Delta x))$, 给定 S^{-1} 层的输入差分, 得到 P^{-1} 层的输出差分, 但这不是 P^{-1} 层的最终输出差分, 需要与 $P^{-1}(\Delta K)$相异或, 得到真正的 P^{-1} 层的输出差分。其中 $P^{-1}(\Delta K)$的生成方式, 我们在 5.2.3 节中讲解。图 9 展示了轮函数中差分搜索的具体实现。正向搜索与反向搜索的过程类似, 都是将非线性层和置换层合并在一起, 并且置换层

的输出与一个值异或作为下一轮非线性层的输入。不同点在于正向搜索中的异或值是子密钥差分，反向搜索中的异或值是子密钥差分经过 P^{-1} 层变换后的值。

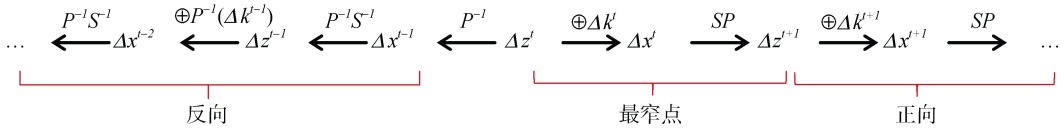

图 9　RECTANGLE 相关密钥差分分析下，轮函数中差分传播的具体实现

Figure 9　Realization of Difference Propagation in Round Function is Based on Related-key Difference Propagation

5.2.3　密钥编排算法中密钥状态差分搜索的具体实现

介绍密钥编排算法中已知某一轮的密钥状态差分，求的所有轮的子密钥差分的具体实现过程。由于采用**从最窄点开始搜索**策略，搜索的起始轮是最窄点。我们初始给定了最窄点的密钥状态差分，已知最窄点的密钥状态差分，需要生成所有轮的子密钥差分。搜索分为正向搜索和逆向搜索，由此，分为正向生成子密钥差分和逆向生成子密钥差分。

$$\Delta a=\begin{pmatrix} 0 & \cdots\cdots & 0 & v_3 & v_2 & v_1 & v_0 \\ 0 & \cdots\cdots & 0 & v_{19} & v_{18} & v_{17} & v_{16} \\ 0 & \cdots\cdots & 0 & v_{35} & v_{34} & v_{33} & v_{32} \\ 0 & \cdots\cdots & 0 & v_{51} & v_{50} & v_{49} & v_{48} \\ 0 & \cdots\cdots & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

图 10-1　Δa 的 16 比特有效位表示 Δv 中参与 *SK* 层运算的 16 比特

Figure 10-1　16-bit Significant Bits in Δa Represent 16 Bits Involved in SK Layer Operations in Δv

$$\Delta b=\begin{pmatrix} v_{15} & \cdots\cdots & v_4 & 0 & 0 & 0 & 0 \\ v_{31} & \cdots\cdots & v_{20} & 0 & 0 & 0 & 0 \\ v_{47} & \cdots\cdots & v_{36} & 0 & 0 & 0 & 0 \\ v_{63} & \cdots\cdots & v_{52} & 0 & 0 & 0 & 0 \\ v_{79} & \cdots\cdots & v_{68} & v_{67} & v_{66} & v_{65} & v_{64} \end{pmatrix}$$

图 10-2　Δb 中 64 比特有效位表示 Δv 中剩余的 64 比特

Figure 10-2　64-bit Significant Bits in Δb Represent the Remaining 64 Bits in Δv

正向生成子密钥差分

由算法 2 可知，已知第 i 轮的密钥状态差分 Δv^i，我们需要生成所有可能的下一轮密钥状态差分 Δv^{i+1}，然后提取 64 比特信息得到子密钥差分 Δk^{i+1}。按照密钥编排算法，生成 Δv^{i+1} 需要两步。第一步是 Δv^i 中的 16 比特经过 4 个 S 盒运算，剩余比特保持不变，结果用 Δu^i 表示。第二步 Δu^i 经过线性变换，结果是 Δv^{i+1}。如图 10-1 和 10-2 所示，Δa 中 16 比特有效位表示 Δv^i 中参与非线性运算的 16 比特，剩余的比特用 Δb 中的有效位表示。我们称 Δa 为非线性部分，Δb 为线性部分。下面等式(3)是成立的，其中 Δx_1 和 Δx_2 分别表示 *PK* 层的输入差分。Δx_1 和 Δx_2 异或后再经过 *PK* 层变换得到的结果与 Δx_1 和 Δx_2 分别经过 *PK* 层变换再进行异或得到的结果是相同的，由此等式(4)是成立的。由等式(4)可以看出，Δa 和 Δb 可以分开来做。Δa 是非线性部分，输出的可能性有多种，Δb 是线性部分，输出是唯一确定的。由此我们可以先求出 Δb 对应的输出，再求所有 Δa 对应的输出。求 Δa 对应的多种输出时，我们采用文献[19]中提出的**最小变动策略**来提升算法的运行效率。

$$PK(\Delta x_1 \oplus \Delta x_2)=PK(\Delta x_1)\oplus PK(\Delta x_2) \quad (3)$$

$$PK(SK(\Delta V))=PK(SK(\Delta a))\oplus PK(\Delta b) \quad (4)$$

1) 先求得 Δb 经过 *PK* 层的输出，输出值是确定的。这个过程可以通过查询 *PK* 层的置换表求得，与轮函数中 *P* 层的置换表是一样的原理。如图 2 所示，将 80 比特的密钥分成 16 个列，每一列 5 个比特。我们生成一个置换表 *PKTable*[16][32]，其中 16 表示 16 个列的位置，32 指的是每一列有 2^5=32 种输入。已知一列的位置及其对应值，通过查询 *PKTable*，可以得到对应 *PK* 层的输出。由此，已知 Δb，将 Δb 分割成 16 列。对每一非零列查询 *PKTable* 表，将查表得到的结果都异或起来即为 *PK*(Δb)。我们可以容易地得到等式(5)、(6)、(7)，其中 Δb_i 的长度为 5 比特，Δc_i 的长度为 80 比特。Δb 是 *PK* 层的输入，等式(7)给出了输出结果。由此我们便得到了线性部分的输出。

$$\Delta b=\Delta b_{15}\|\ldots\|\Delta b_1\|\Delta b_0 \quad (5)$$

$$\Delta c_i=PKTable[i][\Delta b_i], i\in[0,15] \quad (6)$$

$$PK(\Delta b)=\Delta c_{15}\oplus\cdots\oplus\Delta c_1\oplus\Delta c_0 \quad (7)$$

2) 1)中已经获得等式(4)中的 *PK*(Δb)，且其对应的值是唯一确定的。此部分我们需获得 *PK*(*SK*(Δa))。Δa 先经过非线性层运算，再经过置换层运算。此过程与我们对轮函数的处理类似。在轮函数中，我们将 *S* 层和 *P* 层结合在一起实现。同理，为了减少查表次数，我们将 *SK* 层与 *PK* 层结合在一起生成一张表

SPKTable[4][2^4][8], 其中 4 代表 *S* 盒的总数, 2^4 表示每个 *S* 盒可能的输入差分的数目, 8 代表一种输入差分最多对应的输出差分个数。对于一个 4 比特的双射 *S* 盒, 当给定该 *S* 盒的输入差分时, 其输出差分的可能取值不超过 8 种。已知Δa 的值, 可获得活跃 *S* 盒的个数和每个 *S* 盒对应的输入差分, 通过查询 *SPKTable* 可获得每个活跃 *S* 盒对应的 *PK* 层的输出差分(可能有多种输出差分)。将所有活跃 *S* 盒对应的一种输出差分异或起来, 便是 $PK(SK(\Delta a))$的一个可能值。我们可以得到等式(8)-(10), 等式(8)式中我们省略了Δa 中无效的 64 比特, 其中Δa_i的长度为 4 比特, $a_i = v_{i+48}||\ v_{i+32}||v_{i+16}||v_i$, $i \in [0,3]$。Δd_i的长度为 80 比特, *j* 表示当前输入差分对应的第 *j* 个输出差分。对于每一个活跃 *S* 盒, 当 *j* 均取 0 时, 得到的$PK(SK(\Delta a))$是非线性部分第一种输出差分。

$$\Delta a = (\Delta a_3, \Delta a_2, \Delta a_2, \Delta a_1) \tag{8}$$

$$\Delta d_i = SPKTable[i][\Delta a_i][j], i \in [0,3] \tag{9}$$

$$PK(SK(\Delta a)) = \Delta d_3 \oplus \Delta d_2 \oplus \Delta d_1 \oplus \Delta d_0 \tag{10}$$

3) 将 1)得到的结果与 2)中得到的第一种结果相异或, 最终获得了Δv^{i+1} 的第一种取值, 从而获得了Δk^{i+1} 的一种取值。接下来, 2)中采用最小变动策略来获得Δv^{i+1} 所有可能的取值, 从而获得Δk^{i+1} 所有可能的值。

反向生成子密钥差分

已知密钥状态差分Δv^i, 求逆向下一轮密钥状态差分Δv^{i-1}, 再提取 64 比特, 生成子密钥差分Δk^{i-1}。Δk^{i-1} 再经过轮函数中的 P^{-1} 层运算, 运算结果参与到轮函数运算中去。我们需要求得两个值, 分别是Δv^{i-1}和 $P^{-1}(\Delta k^{i-1})$。我们的做法如下:

1) 逆向搜索与正向搜索不同, 不能简单将 PK^{-1} 层与 SK^{-1} 层结合。Δv^i 先经过 PK^{-1} 层运算, 得到的结果分为两部分, 参与 SK^{-1} 运算的 16 比特以及不参加 SK^{-1} 运算的 64 比特, 如 10-1 和 10-2, 我们分别用Δa 和Δb 表示。之后的操作, Δa 和Δb 可以分别进行。

2) Δb 是Δv^{i-1} 的一部分, 因此可以确定Δv^{i-1} 中的 64 比特。由密钥编排算法可知, Δb 中的 48 个比特是Δk^{i-1} 的一部分, 由此可以确定Δk^{i-1} 中的 48 个比特。这 48 比特正好是 12 个列, 且每列包含 4 个比特, 因此可以通过查询P^{-1}层的表得到$P^{-1}(\Delta k^{i-1})$的一部分。获得 $P^{-1}(\Delta k^{i-1})$并不是最终的差分, 只是线性部分Δb 经过变换之后的差分, 并且这个值是唯一确定的。

3) 对于非线性部分Δa 来说, 经过 SK^{-1} 层可能有多种输出差分, 我们在 2)中已经确定线性部分的值, 如此只需遍历非线性部分。已知Δa, 通过查询差分分布表, 求得第一种 SK^{-1} 输出差分, 这个便是Δv^{i-1} 中剩余的 16 比特。从而得到了Δv^{i-1} 的第一种完整取值, 此 16 比特正好可分成 4 个部分, 通过查询 *P* 层的置换表, 将对应结果异或起来, 便是非线性部分Δa 对应的 $P^{-1}(\Delta k^{i-1})$输出。将Δa 对应的 $P^1(\Delta k^{i1})$输出与 2)中得到的 $P^{-1}(\Delta K^{i-1})$异或, 得到了 $P^{-1}(\Delta k^{i-1})$第一种完整值。之后采用文献[19]中**最小变动策略**来遍历所有可能的Δv^{i-1} 和 $P^{-1}(\Delta k^{i-1})$。

6 RECTANGLE-80 抵抗相关密钥差分密码分析的结果分析及讨论

RECTANGLE-80 的搜索结果如表 1 所示, 从中我们可以得到下面的结果:

1) 对于 8 轮 RECTANGLE, 表中最窄点密钥状态差分汉明重量所属的 5 种范围分别对应的最小权重是相同的;

2) 对于轮数小于 8 的缩减轮, 随着最窄点密钥状态差分的汉明重量取值范围的增大, 最小权重趋向稳定。当范围取[1,3], [1,4], [1,5]时, 所获得的各轮最小权重是相同的;

当设置范围是[1,1]或[1,2]时, 9 轮的最小权重均为 42。当范围是[1, 3]时, 最小权重的下界是 41。基于上述情况, 我们推断 9 轮的最小权重至少为 41。由此, 18 轮最小权重至少为 82。

对于一个分组长度为 n 比特、密钥长度为 m 比特的分组密码, 穷举破译该密码的时间复杂度是 2^{-m}。在相关密钥差分分析下, 要利用相关密钥差分密码分析方法成功破译该密码, 首先要找到一条概率大于 2^{-m} 的 r 轮区分器(注: r 的取值和密码算法的总轮数相差不大。对于 RECTANGLE, r 大致为 20)。RECTANGEL-80 的密钥长度是 80 位, 总轮数为 25 轮。通过我们的分析, 18 轮最大差分概率至多为 2^{-82}。考虑到下面两个方面:

1) 文献[4]中对 RECANGLE 的差分聚集效应(differential clustering) 的分析结果表明, RECTANGLE 的差分聚集效应很弱。因此, 在相关密钥密码分析下, 我们可以认为 RECTANGLE 的差分聚集效应也很弱。

2) 由表 1 的结果, 在对最窄点密钥状态差分的汉明重量的限制下, 3 轮~9 轮 RECTANGLE 的相关密钥差分路径的最小权重分别为 3、7、11、15、21、30、41。注意到相邻轮数的最小权重的差值分别是 4、4、4、6、9、11, 可见, 随着轮数的增加, 相邻轮数的最小权重的差也在增加。因此, 我们估计, 实际上, 18 轮 RECTANGLE-80 的相关密钥差分路径的最小权重要比 82 大很多。

表 1 RECTANGLE-80 相关密钥差分分析结果

Table 1 Results of Related-key Differential Cryptanalysis of RECTANGLE-80

轮数(n)	最窄点密钥状态差分的汉明重量				
	1	⩽2	⩽3	⩽4	⩽5
1	0	0	0	0	0
2	2	0	0	0	0
3	4	3	3	3	3
4	7	7	7	7	7
5	11	11	11	11	11
6	17	15	15	15	15
7	22	**22**	21	21	**21**
8	30	30	30	30	**30**
9	42	**42**	⩾41	⩾37	⩾34

(注: 表中元素表示最窄点密钥状态差分的汉明重量限制在某一范围时, 搜索 n 轮的最小权重, $n\in[1,9]$。⩾对应的值表示下界)

由以上分析, 我们认为, 25 轮 RECTANGLE- 80 足够抵抗相关密钥差分密码分析。后续我们将展开对RECTANGLE-128的相关密钥差分密码分析工作。

参考文献

[1] Z. Gong, S. Nikova and Y.W. Law, "KLEIN: a new family of lightweight block ciphers," in Proc. *RFID Security and Privacy* (*RFIDSec'11*), pp. 1-18, 2011.

[2] W.L. Wu and L. Zhang, "LBlock:a lightweight block cipher," in Proc. *Applied Cryptography and Network Security(ACNS'11)*, pp. 327-344, 2011.

[3] A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin and C. Vikkelsoe, "PRESENT: An Ultra-Lightweight Block Cipher ," in Proc. *Cryptographic Hardware and Embedded Systems(CHES'07)*, pp. 450-466, 2007.

[4] W.T. Zhang, Z.Z Bao, D.D Lin, V. Rijmen, B.H. Yang and I. Ver-bauwhede, "RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms," *Science China Information Sciences*, vol. 58, no. 12, pp. 1-15, Dec. 2015.

[5] C. Beierle, J. Jean, S. Kölbl, G. Leander, A. Moradi, T. Peyrin, Y. Sasaki and P. Sasdrich, "The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS," in Proc. *Advances in Cryptology (CRYPTO'16)*, pp. 123-153, 2016.

[6] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks and L. Wingers, "The SIMON and SPECK lightweight block ciphers," in Proc. *Design Automation Conference, 2015 52nd ACM/EDAC/IEEE*, pp. 1-6, 2015.

[7] J Guo, T Peyrin, A Poschmann and Matt Robshaw, "The LED block cipher," in Proc. *Cryptographic Hardware and Embedded Systems(CHES'11)*, PP. 326-341, 2011.

[8] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," *Journal of Cryptology*, vol. 4, no. 1, pp. 3-72, 1991.

[9] E. Biham and A. Shamir, "Differential Cryptanalysis of the Data Encryption Standard," *Crystal Research & Technology*, vol. 17, no. 1, pp. 79-88, 2010.

[10] E. Biham, A. Biryukov and A. Shamir. "Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials," *Journal of Cryptology,* vol. 18, no. 4, pp. 291-311, 2005.

[11] L.R. Knudsen and T.A. Berson. "Truncated Differentials of SAFER," *Fast Software Encryption*. Springer Berlin Heidelberg, 1996.

[12] L.R. Knudsen. "Truncated and higher order differentials". *Lecture Notes in Computer Science*, vol. 1008, pp.196-211, 1994.

[13] M. Matsui, "The First Experimental Cryptanalysis of the Data Encryption Standard," in Proc. *Advances in Cryptology (CRYPTO '94)*, pp.1-11, 1994.

[14] M. Matsui, "Linear Cryptanalysis Method for DES Cipher," in Proc. *Advances in Cryptology (EUROCRYPT'93)*, pp. 386-397, 1993.

[15] M. Matsui, "On correlation between the order of S-boxes and the strength of DES," in Proc. *Advances in Cryptology (EUROCRYPT'94)*, pp. 366-375, 1994.

[16] A. Shimizu and S. Miyaguchi, "Fast data encipherment algorithm FEAL," in Proc. *Advances in Cryptology (EUROCRYPT'87),* pp.267-278, 1987.

[17] K. Ohta, S. Moriai and K. Aoki, "Improving the search algorithm for the best linear expression," in Proc. *Advances in Cryptology (CRYPTO'95)*, pp. 157-170, 1995.

[18] K. Aoki, K. Kobayashi and S. Moriai, "Best differential characteristic search of FEAL," in Proc. *Fast Software Encryption*, pp. 41-53, 1997.

[19] Z.Z Bao, W.T. Zhang and D.D. Lin, "Speeding Up the Search Algorithm for the Best Differential and Best Linear Trails," in Proc. *Information Security and Cryptology (Inscrypt'14)*, pp. 259-285, 2014.

[20] E. Biham, "New Types of Cryptanalytic Attacks Using Related Keys," *Journal of Cryptology*, vol. 7, no. 4, pp. 229-246, 1994

[21] A. Biryukov, D. Khovratovich and I. Nikolić, "*Distinguisher and Related-Key Attack on the Full AES-256*," *Advances in Cryptology - CRYPTO 2009*, Springer Berlin Heidelberg, 2009.

[22] A. Biryukov and I. Nikolić, "Automatic search for related-key differential characteristics in byte-oriented block ciphers: application to AES, Camellia, Khazad and others," in Proc. *International Conference on Theory and Applications of Cryptographic Techniques* Springer-Verlag, pp. 322-344, 2010.

[23] P.A. Fouque, J. Jean, and T. Peyrin, "*Structural Evaluation of AES, and Chosen-Key Distinguisher of 9-Round AES-128,"* in Proc. *Advances in Cryptology – CRYPTO 2013*. Springer Berlin Heidel-

berg, pp. 183-203, 2013.

[24] A. Biryukov and I. Nikolić, "Search for Related-Key Differential Characteristics in DES-Like Ciphers," in Proc. *International Conference on FAST Software Encryption* Springer-Verlag, pp. 18-34, 2011.

[25] Emami, Sareh, S. Ling, I. Nikolić, J. Pieprzyk and H.X. Wang, "The resistance of PRESENT-80, against related-key differential attacks," *Cryptography & Communications*, vol. 6, no. 3, pp. 171-187, 2014.

[26] Z.J. Xiang, W.T. Zhang, Z.Z Bao and D.D Lin, "Applying MILP Method to Searching Integral Distinguishers Based on Division Property for 6 Lightweight Block Ciphers," in Proc. A*dvances in Cryptology – ASIACRYPT 2016.* Springer Berlin Heidelberg, 2016.

王沙沙 于 2015 年在吉林大学计算机科学与技术(网络与信息安全)专业获得学士学位, 现在中国科学院信息工程研究所信息安全专业攻读硕士学位, 研究领域为对称密码算法的安全性分析。Email: wangshasha @iie.ac.cn

张文涛 于 2004 年在中国科学院软件研究所获得博士学位, 现为中国科学院信息工程研究所第一研究室研究员, 研究领域为对称密码算法的设计与分析。Email: zhangwentao@iie.ac.cn

向泽军 于 2012 年在武汉大学基础数学专业获得学士学位, 现在中国科学院信息工程研究所信息安全专业硕博连读, 研究领域为对称密码的安全性分析。Email: xiangzejun@iie.ac.cn

附录

我们在附录中分别给出了表 1 中加黑部分对应的差分路径, 即:

1) 最窄点密钥状态差分的汉明重量取值范围是[1, 2], 7 轮的最优路径。

2) 最窄点密钥状态差分的汉明重量取值范围是[1, 5], 7 轮的最优路径。

3) 最窄点密钥状态差分的汉明重量取值范围是[1, 5], 8 轮的最优路径。

4) 最窄点密钥状态差分的汉明重量取值范围是[1, 2], 9 轮的最优路径。

对 RECTANGLE 算法进行相关密钥差分分析时, 我们需要考虑轮函数差分传播和密钥编排差分传播两部分, 由此对于一个差分路径, 我们在两张表中分别给出轮函数中消息的差分传播和密钥编排中密钥的差分传播路径。于轮函数而言, 固定 S 层的输入差分, S 层的输出差分有多种取值。固定 P 层的输入差分, P 层的输出差分是唯一确定的。因此在给定轮函数中消息的传播路径时, 我们只给定每一轮的子密钥差分和 S 层的输入输出差分。S 层的输出差分即为 P 层的输入差分, 由此可容易获得 P 层的输出差分, P 层的输出差分与子密钥差分异或, 可获得下一轮 S 层的输入差分。同理, 给定密钥编排算法中密钥状态的差分传播路径时, 我们只给定 SK 层的输入输出差分。

1) 当最窄点密钥状态差分的汉明重量不大于 2 时, 7 轮最小权重为 22。表 2-1 和 2-2 中给出了其对应的差分路径。表 2-1 给出了每一轮子密钥差分、S 层的输入输出差分及权重, 表 2-2 给出了每一轮 SK 层的输入输出差分以及生成其他密钥状态差分过程中 SK 层的权重。

2) 同理, 当最窄点密钥状态差分汉明重量不大于 5 时, 7 轮的最小权重为 21, 表 3-1 和 3-2 给出了其对应的差分路径。

3) 同理, 当最窄点密钥状态差分汉明重量不大于 5 时, 8 轮的最小权重为 30, 表 4-1 和 4-2 给出了其对应的差分路径。

4) 同理, 当最窄点密钥状态差分汉明重量不大于 2 时, 9 轮的最小权重为 42, 表 5-1 和 5-2 给出了其对应的差分路径。

表 2-1、表 3-1、表 4-1 和表 5-1 加黑部分表示的是最窄点, 表 2-2、表 3-2、表 4-2 和表 5-2 加黑部分表示的是最窄点密钥状态差分。消息 $M = m_{63}||...||m_1||m_0$, 如图 1 所示, 为了方便实现 S 层的运算, 我们是按照列的方式存储的, 即存储的形式为 $(col_{15} \parallel col_{14} \parallel ... \parallel col_1 \parallel col_0)$。例如当差分为 0x0000044004000040, 我们的存储结果为 0x0000060005000000, 见表 2-1 中第 0 轮 S 层的输入差分。同理, 如图 2 所示, 密钥差分按照列的方式存

储, 区别在于密钥每 1 列 5 个比特, 我们将 5 比特存入一个字节, 即一个字节的高 3 比特是无效位, 低 5 比特存储密钥差分一列的值。因此, 一个密钥差分需要 16 个字节存储, 如表 2-2 所示。

表 2-1　最窄点密钥状态差分汉明重量不大于 2 时, 7 轮的轮函数的一条最优差分路径, 轮函数中 *S* 层的总权重为 19, 第 4 轮是最窄点。

Table 2-1　An Optimal Differential Trails of 7 Rounds is Based on Hamming Weight of the Narrowest Point Not More Than 2. Total Weight of the S layer is 19 and the Narrowest Point is the 4-round.

轮数	子密钥差分	*S* 层输入差分	*S* 层输出差分	权重
0	0x2000400010000000	0x0000060005000000	0x0000020008000000	4
1	0x0000200000008000	0x0000000000000000	0x0000000000000000	0
2	0x0000000080000000	0x0000000080000000	0x0000000010000000	3
3	0x0000000040000000	0x0000000050000000	0x0000000010000000	3
4	**0x0000000020000000**	**0x0000000030000000**	**0x0000000090000000**	**2**
5	0x0000000010000000	0x0000000000080000	0x0000000000010000	3
6	0x1000000000000000	0x1000000000010000	0x6000000000060000	4

表 2-2　最窄点密钥状态差分汉明重量不大于 2 时, 7 轮密钥编排的一条最优差分路径, 密钥编排过程中 *SK* 层所产生的总权重为 3, 第 4 轮是最窄点。

Table 2-2　An Optimal Differential Trails of 7 Rounds is Based on Hamming Weight of the Narrowest Point Not More Than 2. Total Weight of the SK layer is 3 and the Narrowest Point is the 4-round.

轮数	*SK* 层输入差分	*SK* 层输出差分	权重
0	0x02000000040000000100000010000000	0x02000000040000000100000010000000	0
1	0x00000000020000001000000008000000	0x00000000020000001000000001000000	3
2	0x00000000000000000800000010000000	0x00000000000000000800000010000000	0
3	0x00000000000000000400000000000000	0x00000000000000000400000000000000	0
4	**0x00000000000000000200000000000000**	0x00000000000000000200000000000000	0
5	0x00000000000000000100000000000000	0x00000000000000000100000000000000	0
6	0x01000000000000001000000000000000		

表 3-1　最窄点密钥状态差分汉明重量不大于 5 时, 7 轮的轮函数的一条最优差分路径, 轮函数中 *S* 层所产生的总权重为 19, 第 1 轮是最窄点。

Table 3-1　An Optimal Differential Trails of 7 Rounds is Based on Hamming Weight of the Narrowest Point Not More Than 5. Total Weight of the S layer is 19 and the Narrowest Point is the 1-round.

轮数	子密钥差分	*S* 层输入差分	*S* 层输出差分	权重
0	0x4000100020006000	0x06000000b0000000	0x0200000010000000	4
1	**0x2000000010000000**	**0x0000000000000000**	**0x0000000000000000**	**0**
2	0x0000800000000000	0x0000800000000000	0x0000100000000000	3
3	0x0000400000000000	0x0000500000000000	0x0000100000000000	3
4	0x0000200000000000	0x0000300000000000	0x0000900000000000	2
5	0x0000100000000000	0x0000000800000000	0x0000000100000000	3
6	0x0000000000001000	0x0000000100001000	0x0000000600006000	4

表 3-2　最窄点密钥状态差分汉明重量不大于 5 时, 7 轮密钥编排的一条最优差分路径, 密钥编排过程中 *SK* 层所产生的总权重为 2, 第 1 轮是最窄点。

Table 3-2　An Optimal Differential Trails of 7 Rounds is Based on Hamming Weight of the Narrowest Point Not More Than 5. Total Weight of the SK layer is 2 and the Narrowest Point is the 1-round.

轮数	*SK* 层输入差分	*SK* 层输出差分	权重
0	0x0400000001000000020000006000000	0x0400000001000000020000002000000	2
1	**0x0200000010000000010000000000000**	0x0200000010000000010000000000000	0
2	0x0000000008000000100000000000000	0x0000000008000000100000000000000	0
3	0x0000000004000000000000000000000	0x0000000004000000000000000000000	0
4	0x0000000002000000000000000000000	0x0000000002000000000000000000000	0
5	0x0000000001000000000000000000000	0x0000000001000000000000000000000	0
6	0x0000000010000000000000001000000		

表 4-1　最窄点密钥状态差分汉明重量不大于 5 时, 8 轮的轮函数的一条最优差分路径, 轮函数中 *S* 层所产生的总权重为 27, 第 4 轮是最窄点。

Table 4-1　An Optimal Differential Trails of 8 Rounds is Based on Hamming Weight of the Narrowest Point Not More Than 2. Total Weight of the S layer is 27 and the Narrowest Point is the 4-round.

轮数	子密钥差分	*S* 层输入差分	*S* 层输出差分	权重
0	0x2000400010000000	0x0000060005000000	0x0000020008000000	4
1	0x0000200000008000	0x0000000000000000	0x0000000000000000	0
2	0x0000000080000000	0x0000000080000000	0x0000000010000000	3
3	0x0000000080000000	0x0000000050000000	0x0000000010000000	3
4	**0x0000000020000000**	**0x0000000030000000**	**0x0000000090000000**	**2**
5	0x0000000010000000	0x0000000000080000	0x0000000000010000	3
6	0x1000000000000000	0x1000000000010000	0x6000000000060000	4
7	0x0000000090000000	0x0000400090200006	0x0000300060d00002	8

表 4-2　最窄点密钥状态差分汉明重量不大于 5 时, 8 轮密钥编排的一条最优差分路径, 密钥编排过程中 *SK* 层所产生的总权重为 3, 第 4 轮是最窄点。

Table 2-2　An Optimal Differential Trails of 78Rounds is Based on Hamming Weight of the Narrowest Point Not More Than 5. Total Weight of the SK layer is 3 and the Narrowest Point is the 4-round.

轮数	*SK* 层输入差分	*SK* 层输出差分	权重
0	0x0200000004000000010000010000000	0x0200000004000000010000010000000	0
1	0x0000000002000000100000008000000	0x0000000002000000100000001000000	3
2	0x0000000000000000080000010000000	0x0000000000000000080000010000000	0
3	0x0000000000000000080000010000000	0x0000000000000000080000010000000	0
4	**0x0000000000000000020000000000000**	0x0000000000000000020000000000000	0
5	0x0000000000000000010000000000000	0x0000000000000000010000000000000	0
6	0x0100000000000000100000000000000	0x0100000000000000100000000000000	0
7	0x1000000000000000090000000000000		

表 5-1　最窄点密钥状态差分汉明重量不大于 2 时, 9 轮的轮函数的一条最优差分路径, 轮函数中 *S* 层所产生的总权重为 39, 第 3 轮是最窄点。

Table 5-1　An Optimal Differential Trails of 9 Rounds is Based on Hamming Weight of the Narrowest Point Not More Than 2. Total Weight of the S layer is 39 and the Narrowest Point is the 3-round.

轮数	子密钥差分	*S* 层输入差分	*S* 层输出差分	权重
0	0x2000400010000000	0x0000060005000000	0x0000020008000000	4
1	0x0000200000008000	0x0000000000000000	0x0000000000000000	0
2	0x0000000080000000	0x0000000080000000	0x0000000010000000	3
3	**0x0000000040000000**	**0x0000000050000000**	**0x0000000040000000**	**2**
4	0x0000000020000000	0x0000000020004000	0x00000000d0007000	4
5	0x0000000010000000	0x40000000000a5000	0x3000000000068000	7
6	0x1000000000000000	0x000000000020000e	0x0000000000600002	5
7	0x0000000090000000	0x0000000092000060	0x0000000018000020	8
8	0x9000000040008000	0x9000000050000200	0x6000000040000d00	6

表 5-2　最窄点密钥状态差分汉明重量不大于 2 时, 9 轮密钥编排的一条最优差分路径, 密钥编排过程中 *SK* 层所产生的总权重为 3, 第 3 轮是最窄点。

Table 5-2　An Optimal Differential Trails of 9 Rounds is Based on Hamming Weight of the Narrowest Point Not More Than 2. Total Weight of the SK layer is 3 and the Narrowest Point is the 3-round.

轮数	*SK* 层输入差分	*SK* 层输出差分	权重
0	0x02000000040000000100000010000000	0x02000000040000000100000010000000	0
1	0x00000000020000001000000008000000	0x00000000020000001000000001000000	3
2	0x00000000000000000800000010000000	0x00000000000000000800000010000000	0
3	**0x00000000000000000400000000000000**	0x00000000000000000400000000000000	0
4	0x00000000000000000200000000000000	0x00000000000000000200000000000000	0
5	0x00000000000000000100000000000000	0x00000000000000000100000000000000	0
6	0x01000000000000001000000000000000	0x01000000000000001000000000000000	0
7	0x10000000000000000900000000000000	0x10000000000000000900000000000000	0
8	0x09000000000000001400000008000000		