

# 自动化搜索 ARX 密码差分特征的方法

刘正斌<sup>1,2</sup>

(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 100093;  
2. 中国科学院大学, 北京 100049)

**摘要:** 要解决 ARX 密码算法差分特征的自动化搜索问题, 关键是要解决搜索过程中模加差分的快速计算。首先, 提出了相关差分分布表的概念, 通过查找相关差分分布表, 可以有效地计算模加的差分以及差分概率; 其次, 利用相关差分分布表, 将 Matsui 算法扩展到 ARX 密码, 提出了自动化搜索 ARX 密码差分特征的算法; 最后, 利用提出的搜索算法, 搜索 SPECK 算法的差分特征, 得到了 SPECK32、SPECK48 和 SPECK64 的最优差分特征。

**关键词:** 分组密码; ARX; 差分特征; 自动化搜索; SPECK

**中图分类号:** TP393

**文献标识码:** A

**doi:** 10.11959/j.issn.2096-109x.2016.00050

## Automatic search algorithm for differential characteristics in ARX ciphers

LIU Zheng-bin<sup>1,2</sup>

(1. The State Key Lab of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China;  
2. University of Chinese Academy of Sciences, Beijing 100049, China)

**Abstract:** How to compute the differential of modular addition efficiently is critical to automatic search for differential characteristic in ARX ciphers. To solve this problem, firstly, the concept of collaterated difference distribution table (CDDT) was proposed. By looking up the CDDT, it was very efficient to compute the differential probability of modular addition. Secondly, extending Matsui's algorithm to ARX ciphers and using CDDT, an automatic search algorithm was proposed, and the algorithm could give the differential characteristic with highest probability in ARX ciphers. Finally, the proposed algorithm was applied to the ARX cipher SPECK, and got the best differential characteristics for SPECK32, SPECK48 and SPECK64.

**Key words:** block cipher, ARX, differential characteristic, automatic search, SPECK

## 1 引言

许多分组密码算法采用 ARX 结构——模加 (addition)、循环移位 (rotation)、异或 (XOR), 这些密码算法称为 ARX 密码算法。这种设计的

优点在于算法设计简单、执行效率高, 并且非常适于软硬件实现。到目前为止, 已经提出了许多 ARX 密码算法, 比较著名的算法有分组密码 FEAL<sup>[1]</sup>、TEA\XTEA<sup>[2]</sup>、RC5<sup>[3]</sup>、HIGHT<sup>[4]</sup>和 SPECK<sup>[5]</sup>; 序列密码 Salsa20<sup>[6]</sup>; 散列函数 MD4\MD5<sup>[7]</sup>、SHA-1

收稿日期: 2016-03-17; 修回日期: 2016-04-27。通信作者: 刘正斌, liuzhengbin@iie.ac.cn

基金项目: 国家自然科学基金资助项目 (No.61379142); 国家重点基础研究发展计划 (“973”计划) 基金资助项目 (No.2013CB834203)

**Foundation Items:** The National Natural Science Foundation of China (No.61379142), The National Basic Research Program of China (973 Program)(No.2013CB834203)

以及 SHA-3 中的 Skein<sup>[8]</sup>和 Blake<sup>[9]</sup>等。

差分分析<sup>[10]</sup>是分组密码中最有效的分析方法,对于现代分组密码的设计,抵抗差分分析是一个基本的设计准则。对于基于 S 盒的分组密码,目前,已经存在多种自动化搜索差分特征的算法,利用这些算法可以评估分组密码抵抗差分分析的安全性<sup>[11~18]</sup>。由于基于 S 盒的分组密码通常使用 8 bit 或者 4 bit 的 S 盒,可以很容易构造 S 盒的差分分布表。而对于 ARX 密码算法,由于模加是其唯一的非线性部件,要构造  $n$  bit 模加的差分分布表,需要  $2^{3n} \times 4$  B 的存储空间。由于 ARX 密码通常使用 32 bit 的模加作为非线性部件,要构造其差分分布表是不现实的。

Biryukov 等<sup>[19]</sup>提出了部分差分分布表(PDDT, partial difference distribution table)的概念,在部分差分分布表中,仅仅存储差分概率大于特定阈值的差分值。虽然无法构造  $n$  bit 模加( $n \geq 16$ )的差分分布表,但是可以非常有效地构造其部分差分分布表。利用部分差分分布表,Biryukov 等首次将 Matsui 算法<sup>[16]</sup>扩展到了 ARX 密码差分特征的搜索,提出了一种阈值搜索算法,并用该算法搜索 ARX 密码算法 TEA/XTEA、SPECK 和 SIMON 的差分特征,得到一些改进的差分特征<sup>[19~20]</sup>。但是,Biryukov 等的算法无法保证得到最优的差分特征。

本文提出了一种通用的自动化搜索 ARX 密码差分特征的算法,该算法也是基于 Matsui 的分支定界算法。主要思想是首先将  $n$  bit 模加的计算分块成小比特字,如 4 bit 字或者 8 bit 字;然后计算小比特字的部分和;最后再将这些部分和组合起来计算整个  $n$  bit 模加。对于 4 bit 字或者 8 bit 字,很容易构造其部分和的真值表。在基于 S 盒的分组密码中,各个 S 盒之间是相互独立的,然而,由于进位的影响,部分和的真值表是相互关联的,本文将这种相互关联的真值表称为相关 S 盒。

类似地,可以构造相关差分分布表(CDDT, correlated difference distribution table),相关差分分布表其实就是这些部分和的差分分布表。通过查找相关差分分布表,可以很容易地计算模加中给定输入差分对应的输出差分以及相应的差分概率。基于 CDDT 提出了一种自动化搜索 ARX 密

码差分特征的算法。将此算法应用到 SPECK 算法的 3 个实例 SPECK32、SPECK48 和 SPECK64 中,得到了 SPECK32、SPECK48 和 SPECK64 的最优差分特征。由于此算法能够遍历 SPECK32 和 SPECK48 的明文差分,所以,能够得到其抗差分分析的可证明的安全界。对于 SPECK64,尽管本文没有遍历所有可能的明文差分,但是其结果依然是目前最优的。

## 2 相关 S 盒与相关差分分布表

**定义 1** 设  $x, y \in \mathbb{F}_2^n$ , 记  $x \boxplus y = x + y \bmod 2^n$ ,  $x \boxplus y$  的进位向量记为  $\text{carry}(x, y) = (c_{n-1}, \dots, c_1, c_0)$ , 则

$$x \boxplus y = x \oplus y \oplus \text{carry}(x, y) \quad (1)$$

其中,  $\text{carry}(x, y)$  可按如下方式迭代计算:  $c_0 = 0$ ,  $c_{i+1} = (x_i \wedge y_i) \oplus (x_i \wedge c_i) \oplus (y_i \wedge c_i)$ ,  $0 \leq i \leq n-2$ 。

**定义 2** 设  $x, y, z, x', y', z' \in \mathbb{F}_2^n$ ,  $z = x \boxplus y$ ,  $z' = x' \boxplus y'$ 。令  $\alpha, \beta, \gamma$  分别表示  $x, y, z$  的异或差分, 即  $\alpha = x \oplus x', \beta = y \oplus y', \gamma = z \oplus z'$ 。将  $n$  比特模加  $z = x \boxplus y$  的差分定义为由输入差分  $\alpha, \beta$  和输出差分  $\gamma$  构成的三元组  $(\alpha, \beta \mapsto \gamma)$ , 则  $n$  比特模加的差分概率定义为

$$P(\alpha, \beta \mapsto \gamma) = P_{x,y} [(x \boxplus y) \oplus ((x \oplus \alpha) \boxplus (y \oplus \beta)) = \gamma] \quad (2)$$

**定义 3** 设  $x, y, z \in \mathbb{F}_2^n$ ,  $z = x \boxplus y$ 。假设  $m, t \in \mathbb{Z}$ , 且满足  $n = m \times t$ , 则  $z = x \boxplus y$  可以分块成  $m$  个部分和, 其中, 每一个部分和对应  $t$  bit 的加法。遍历 2 个  $t$  bit 加数的所有可能取值, 可以构造  $t$  bit 部分和的真值表, 这样, 通过依次查找这些部分和的真值表就可以计算  $z = x \boxplus y$  (由于进位的影响, 只能从最低比特对应的真值表开始依次查找), 这些部分和的真值表称为相关 S 盒。

当然, 要计算  $n$  bit 模加, 没有必要通过查找相关 S 盒来实现, 因为计算机软件可以快速实现模加的计算。但是, 要计算  $n$  bit 模加的差分却非常耗时, 因为给定 2 个加数的差分, 需要遍历和差分的所有可能取值才能确定所有可能的和差分。对于  $n$  bit 的字, 需要遍历  $2^n$  种可能取值, 由于密码算法中通常使用 32 bit 的字, 因此, 这是无法实现的。下面给出相关差分分布表的定义, 利用相关差分分布表可以非常有效地计算模加的差分, 因此, 利用相关差分分布表可以有效地搜索 ARX 密码的差分特征, 就像基于 S 盒的密码中搜索差分特征一样。

**定义4** 设  $x, y, z \in \mathbb{F}_2^n$ ,  $z = x \oplus y$ 。假设  $m, t \in \mathbb{Z}$ , 且满足  $n = m \times t$ 。令  $\alpha, \beta, \gamma$  分别表示  $x, y, z$  的异或差分,  $\gamma = (x \oplus y) \oplus ((x \oplus \alpha) \oplus (y \oplus \beta))$ , 则  $\gamma$  的计算可以划分成  $m$  块, 每一块包含  $t$  bit。对于每一块可以构造相应的差分分布表, 由于加法运算中进位的影响, 2 个相邻的差分分布表是相关联的, 这样的差分分布表称为相关差分分布表。

利用相关差分分布表计算模加差分概率的过程如下: 设  $x, y, z, x', y' \in \mathbb{F}_2^n$ ,  $z = x \oplus y$ , 令  $\alpha, \beta, \gamma$  分别表示  $x, y, z$  的异或差分, 即  $\alpha = x \oplus x', \beta = y \oplus y'$ , 则

$$\begin{aligned} \gamma &= (x \oplus y) \oplus ((x \oplus \alpha) \oplus (y \oplus \beta)) \\ &= \alpha \oplus \beta \oplus \text{carry}(x, y) \oplus \text{carry}(x', y') \end{aligned} \quad (3)$$

其中,  $\text{carry}(x, y) = (c_{n-1}, \dots, c_1, c_0)$  表示  $x \oplus y$  的进位向量,  $\text{carry}(x', y') = (c'_{n-1}, \dots, c'_1, c'_0)$  表示  $x' \oplus y'$  的进位向量,  $c_0 = 0, c'_0 = 0$ 。

假设  $m, t \in \mathbb{Z}$ , 且满足  $n = m \times t$ , 将  $\gamma$  划分成  $m$  块, 每一块包含  $t$  bit, 这样, 可以构造  $m$  个相关差分分布表, 记为  $CS_0, CS_1, \dots, CS_{m-1}$ 。令  $\Delta I_k = \alpha[(k+1)t-1:kt]$  和  $\Delta O_k = \gamma[(k+1)t-1:kt]$  分别表示  $CS_k (0 \leq k \leq m-1)$  的输入和输出差分,  $d_k = c_{kt} \| c'_{kt}$  表示  $CS_{k-1}$  到  $CS_k$  的进位标记,  $d_0 = 0$ , 其中 “ $\|$ ” 表示比特串的级联。

令

$$X_k = x[(k+1)t-1:kt], Y_k = y[(k+1)t-1:kt] \quad (4)$$

$$X'_k = x'[(k+1)t-1:kt], Y'_k = y'[(k+1)t-1:kt] \quad (5)$$

$$\Gamma_k = \gamma[(k+1)t-1:kt] \quad (6)$$

$\delta_k$  定义为  $(X_k \oplus Y_k \oplus c_{kt}) \oplus (X'_k \oplus Y'_k \oplus c'_{kt}) = \Gamma_k$ , 则  $n$  比特模加的差分概率可以按照全概率公式进行如下计算。

1) 当  $m=1$  时

$$P(\alpha, \beta \mapsto \gamma) = P(\delta_0) \quad (7)$$

2) 当  $m=2$  时

$$P(\alpha, \beta \mapsto \gamma) = \sum_{g_1 \in \mathbb{F}_2^t} P(\delta_1 | d_1 = g_1) P(\delta_0, d_1 = g_1) \quad (8)$$

3) 当  $m \geq 3$  时

$$\begin{aligned} P(\alpha, \beta \mapsto \gamma) &= \sum_{g_{m-1} \in \mathbb{F}_2^t} \sum_{g_t \in \mathbb{F}_2^t} P(\delta_{m-1} | d_{m-1} = g_{m-1}) \cdot \\ &\prod_{k=1}^{m-2} P(\delta_k, d_{k+1} = g_{k+1} | d_k = g_k) P(\delta_0, d_1 = g_1) \end{aligned} \quad (9)$$

下面给出一个具体实例来说明相关差分分布表的构造, 以及利用相关差分分布表计算模加的差分概率。

**例1** 设  $x, y, z, \alpha, \beta, \gamma \in \mathbb{F}_2^4$ ,  $z = x \oplus y$ ,  $\alpha, \beta, \gamma$  分别表示  $x, y, z$  的异或差分。令  $\alpha = (1010)_2$ ,  $\beta = (1101)_2$ ,  $\gamma = (0101)_2$ , 则根据模加差分概率的定义, 可以计算出  $P(\alpha, \beta \mapsto \gamma) = \frac{1}{8}$ 。

接下来, 利用相关差分分布表计算  $P(\alpha, \beta \mapsto \gamma)$ 。令  $c$  表示  $x \oplus y$  的进位向量,  $c'$  表示  $(x \oplus \alpha) \oplus (y \oplus \beta)$  的进位向量,  $c = (c_3, c_2, c_1, 0)$ ,  $c' = (c'_3, c'_2, c'_1, 0)$ 。因为  $x, y, z$  都是 4 bit 字, 所以, 可以构造 2 个 2 bit 的相关差分分布表  $CS_0$  和  $CS_1$ ,  $CS_0$  和  $CS_1$  通过进位比特  $c_2$  和  $c'_2$  相关联。因为  $c_2, c'_2 \in \mathbb{F}_2$ , 所以  $CS_0$  和  $CS_1$  各包含 4 张表。

记

$$x = (x_3, x_2, x_1, x_0), X_1 = (x_3, x_2), X_0 = (x_1, x_0) \quad (10)$$

$$y = (y_3, y_2, y_1, y_0), Y_1 = (y_3, y_2), Y_0 = (y_1, y_0) \quad (11)$$

$$z = (z_3, z_2, z_1, z_0), Z_1 = (z_3, z_2), Z_0 = (z_1, z_0) \quad (12)$$

$$\alpha = (\alpha_3, \alpha_2, \alpha_1, \alpha_0), A_1 = (\alpha_3, \alpha_2), A_0 = (\alpha_1, \alpha_0) \quad (13)$$

$$\beta = (\beta_3, \beta_2, \beta_1, \beta_0), B_1 = (\beta_3, \beta_2), B_0 = (\beta_1, \beta_0) \quad (14)$$

$$\gamma = (\gamma_3, \gamma_2, \gamma_1, \gamma_0), \Gamma_1 = (\gamma_3, \gamma_2), \Gamma_0 = (\gamma_1, \gamma_0) \quad (15)$$

那么,  $(A_0, B_0, \Gamma_0)$  就是相关差分分布表  $CS_0$  的输入和输出差分,  $(A_1, B_1, \Gamma_1)$  就是相关差分分布表  $CS_1$  的输入和输出差分, 在存储过程中将输入差分  $(A_0, B_0)$  和  $(A_1, B_1)$  分别存储成  $A_0 \| B_0$  和  $A_1 \| B_1$ 。相关差分分布表  $CS_0$  和  $CS_1$  的构造算法如下。

**算法1** 相关差分分布表  $CS_0$  和  $CS_1$  的构造算法

1) 构造  $CS_0$ , 其中,  $d$  表示模加中的进位标记, 用于选择  $CS_1$  的表。

for  $A_0, B_0, \Gamma_0 = 0$  to 3 do

for  $X_0, Y_0 = 0$  to 3 do

$Z_0 = X_0 \oplus Y_0$ ;

$Z'_0 = (X_0 \oplus A_0) \oplus (Y_0 \oplus B_0)$ ;

$c_2 = \left\lfloor \frac{X_0 + Y_0}{4} \right\rfloor$ ;

$c'_2 = \left\lfloor \frac{(X_0 \oplus A_0) + (Y_0 \oplus B_0)}{4} \right\rfloor$ ;

$d = c_2 \| c'_2$ ;

if  $Z_0 \oplus Z'_0 = \Gamma_0$  then

$CS_0^d(A_0, B_0 \mapsto \Gamma_0) = CS_0^d(A_0,$

$B_0 \mapsto \Gamma_0) + 1$ ;

end if

end for

end for

2) 构造  $CS_1$ 。

for  $A_1, B_1, \Gamma_1=0$  to 3 do

for  $X_1, Y_1=0$  to 3 do

for  $c_2, c'_2=0$  to 1 do

$Z_1=X_1 \boxplus Y_1 \boxplus c_2$ ;

$Z'_1=(X_1 \oplus A_1) \boxplus (Y_1 \oplus B_1) \boxplus c'_2$ ;

$d=c_2 \parallel c'_2$ ;

if  $Z_1 \oplus Z'_1=\Gamma_1$  then

$CS_{1,d}(A_1, B_1 \mapsto \Gamma_1)=CS_{1,d}(A_1,$

$B_1 \mapsto \Gamma_1)+1$ ;

end if

end for

end for

end for

最后,  $CS_0$  和  $CS_1$  中的每一个元素除以 16 就得到相应的差分概率, 相关差分分布表  $CS_0$  和  $CS_1$  如表 1 和表 2 所示。对于上面的例子, 通过查找相关差分分布表可以得到。

表 1

相关差分分布表  $CS_0$

| $A_0 \parallel B_0$ | $\Gamma_0$    |               |               |               |               |               |               |               |               |               |               |               |               |               |               |               |
|---------------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|
|                     | $CS_0^0$      |               |               |               | $CS_0^1$      |               |               |               | $CS_0^2$      |               |               |               | $CS_0^3$      |               |               |               |
|                     | 0             | 1             | 2             | 3             | 0             | 1             | 2             | 3             | 0             | 1             | 2             | 3             | 0             | 1             | 2             | 3             |
| 0                   | $\frac{5}{8}$ | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | $\frac{3}{8}$ | 0             | 0             | 0             |
| 1                   | 0             | $\frac{3}{8}$ | 0             | $\frac{1}{8}$ | 0             | 0             | 0             | $\frac{1}{8}$ | 0             | 0             | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ |
| 2                   | 0             | 0             | $\frac{3}{8}$ | 0             | 0             | 0             | $\frac{1}{4}$ | 0             | 0             | 0             | $\frac{1}{4}$ | 0             | 0             | 0             | $\frac{1}{8}$ | 0             |
| 3                   | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{4}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ | 0             | 0             |
| 4                   | 0             | $\frac{3}{8}$ | 0             | $\frac{1}{8}$ | 0             | 0             | 0             | $\frac{1}{8}$ | 0             | 0             | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ |
| 5                   | $\frac{3}{8}$ | 0             | $\frac{1}{8}$ | 0             | 0             | 0             | $\frac{1}{8}$ | 0             | 0             | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ | 0             |
| 6                   | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{4}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ | 0             | 0             |
| 7                   | $\frac{1}{8}$ | 0             | $\frac{1}{4}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ | 0             | 0             | 0             |
| 8                   | 0             | 0             | $\frac{3}{8}$ | 0             | 0             | 0             | $\frac{1}{4}$ | 0             | 0             | 0             | $\frac{1}{4}$ | 0             | 0             | 0             | $\frac{1}{8}$ | 0             |
| 9                   | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{4}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ | 0             | 0             |
| 10                  | $\frac{3}{8}$ | 0             | 0             | 0             | $\frac{1}{4}$ | 0             | 0             | 0             | $\frac{1}{4}$ | 0             | 0             | 0             | $\frac{1}{8}$ | 0             | 0             | 0             |
| 11                  | 0             | $\frac{1}{4}$ | 0             | 0             | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{4}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{4}$ | 0             | 0             | 0             | 0             |
| 12                  | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{4}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ | 0             | 0             |
| 13                  | $\frac{1}{8}$ | 0             | $\frac{1}{4}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{8}$ | 0             | 0             | 0             |
| 14                  | 0             | $\frac{1}{4}$ | 0             | 0             | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{4}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{4}$ | 0             | 0             | 0             | 0             |
| 15                  | $\frac{1}{4}$ | 0             | 0             | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{4}$ | 0             | $\frac{1}{8}$ | 0             | $\frac{1}{4}$ | 0             | 0             | 0             | 0             | 0             |

表 2

相关差分分布表  $CS_1$ 

| $A_0  B_0$ | $\Gamma_0$    |               |               |               |               |               |               |               |               |               |               |               |               |               |               |               |
|------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|
|            | $CS_{1,0}$    |               |               |               | $CS_{1,1}$    |               |               |               | $CS_{1,2}$    |               |               |               | $CS_{1,3}$    |               |               |               |
|            | 0             | 1             | 2             | 3             | 0             | 1             | 2             | 3             | 0             | 1             | 2             | 3             | 0             | 1             | 2             | 3             |
| 0          | 1             | 0             | 0             | 0             | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 1             | 0             | 0             | 0             |
| 1          | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ |
| 2          | 0             | 0             | 1             | 0             | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | 0             | 1             | 0             |
| 3          | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ |
| 4          | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ |
| 5          | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | 0             | 0             | 0             | 1             | 0             | 0             | 0             | 1             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             |
| 6          | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ |
| 7          | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | 0             | 1             | 0             | 0             | 0             | 1             | 0             | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             |
| 8          | 0             | 0             | 1             | 0             | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | 0             | 1             | 0             |
| 9          | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ |
| 10         | 1             | 0             | 0             | 0             | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 1             | 0             | 0             | 0             |
| 11         | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ |
| 12         | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ |
| 13         | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | 0             | 1             | 0             | 0             | 0             | 1             | 0             | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             |
| 14         | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | 0             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ |
| 15         | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             | 0             | 0             | 0             | 1             | 0             | 0             | 0             | 1             | $\frac{1}{2}$ | 0             | $\frac{1}{2}$ | 0             |

$$A_1=(\alpha_3, \alpha_2)=(10)_2, \quad A_0=(\alpha_1, \alpha_0)=(10)_2 \quad (16)$$

$$B_1=(\beta_3, \beta_2)=(11)_2, \quad B_0=(\beta_1, \beta_0)=(01)_2 \quad (17)$$

$$\Gamma_1=(\gamma_3, \gamma_2)=(01)_2, \quad \Gamma_0=(\gamma_1, \gamma_0)=(01)_2 \quad (18)$$

因此, 可以计算出

$$\begin{aligned}
 P(\alpha, \beta \rightarrow \gamma) &= \sum_{d \in \mathbb{F}_2} CS_0^d(A_0, B_0 \mapsto \Gamma_0) CS_{1,d}(A_1, B_1 \mapsto \Gamma_1) \\
 &= CS_0^0(1001 \mapsto 01) \cdot CS_{1,0}(1011 \mapsto 01) + \\
 &\quad CS_0^1(1001 \mapsto 01) \cdot CS_{1,1}(1011 \mapsto 01) + \\
 &\quad CS_0^2(1001 \mapsto 01) \cdot CS_{1,2}(1011 \mapsto 01) +
 \end{aligned}$$

$$CS_0^3(1001 \mapsto 01) CS_{1,3}(1011 \mapsto 01)$$

$$= \frac{1}{8} \times \frac{1}{2} + \frac{1}{8} \times 0 + \frac{1}{8} \times 0 + \frac{1}{8} \times \frac{1}{2}$$

$$= \frac{1}{8} \quad (19)$$

### 3 ARX 密码差分特征的自动化搜索算法

在 1994 年的欧密会上, Matsui 提出了一种自动化搜索 DES 最优差分特征的算法<sup>[16]</sup>。Matsui

算法本质上是一种深度优先的搜索算法，对于  $n$  轮差分特征，Matsui 算法执行一个递归搜索。如果已知  $i$  轮最优差分特征概率  $B_i (0 \leq i \leq n-1)$  和  $n$  轮最优差分特征概率的一个初始估计值  $\overline{B}_n$ ，Matsui 算法就能给出  $n$  轮最优差分特征概率  $B_n$ ，对轮数  $i$  进行迭代，就可以得到所有轮数的最优差分特征概率。然而 Matsui 算法仅仅适用于基于 S 盒的分组密码，并不能用于搜索 ARX 密码的差分特征，因为无法直接构造模加的差分分布表。Biryukov 等在 CT-RSA2014 提出了搜索 ARX 密码差分特征的阈值搜索算法<sup>[19]</sup>，他们提出部分差分分布表的概念，第一次将 Matsui 算法扩展到了 ARX 密码差分特征的搜索。尽管 Biryukov 等的算法可以用于搜索各种 ARX 密码的差分特征，但是他们的算法无法保证得到最优的差分特征。

本文提出了相关差分分布表的概念，利用相关差分分布表，同样将 Matsui 算法扩展到了 ARX 密码差分特征的搜索。因为本文的算法能够遍历明文差分，所以本文的算法能够给出 ARX 密码的最优差分特征。下面以 Feistel 结构为例，给出本文的算法。Feistel 结构的一轮如图 1 所示，其中 F 表示包含循环移位（rotation）、逻辑移位（shift）、异或运算（XOR）等部件的线性层。

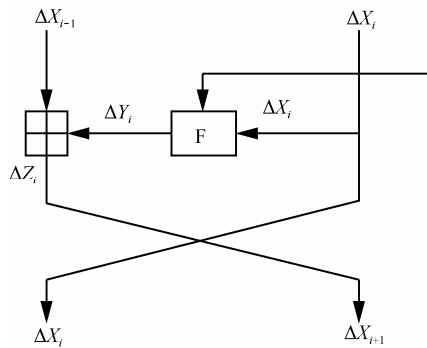


图 1 第  $i$  轮的差分传播

## 算法 2 本文算法

### Procedure Round-1:

Begin the program

For each candidate for  $\Delta X_0, \Delta X_1, \Delta Z_1$ , do the following:

$\Delta Y_1 = F(\Delta X_1)$ ;

Let  $p_1 = P(\Delta X_0, \Delta Y_1 \mapsto Z_1)$ ;

If  $p_1 B_{n-1} \geq \overline{B}_n$ , then call Procedure Round-2;

Exit the program

### Procedure Round- $i$ ( $2 \leq i \leq n-1$ ):

For each candidate for  $\Delta Z_i$ , do the following:

Let  $\Delta X_i = \Delta Z_{i-1}$  and  $\Delta Y_i = F(\Delta X_i)$ ;

Let  $p_i = P(\Delta X_{i-1}, Y_i \mapsto Z_i)$ ;

If  $p_1 \cdot p_2 \cdots p_i B_{n-i} \geq \overline{B}_n$ , then call Procedure Round- $(i+1)$ ;

Return to the upper procedure.

### Procedure Round- $n$ :

Let  $\Delta X_n = \Delta Z_{n-1}$  and  $\Delta Y_n = F(\Delta X_n)$ ;

Let  $p_n = \max_{\Delta Z_n} P(\Delta X_{n-1}, \Delta Y_n \mapsto \Delta Z_n)$ ;

If  $p_1 \cdot p_2 \cdots p_n \geq \overline{B}_n$ , then  $\overline{B}_n = p_1 \cdot p_2 \cdots p_n$ ;

Return to the upper procedure

当计算  $P(\Delta X_{i-1}, \Delta Y_i \mapsto \Delta Z_i)$  时，只需要通过查找相关差分分布表即可，跟基于 S 盒的密码算法中差分概率的计算类似。对于  $n$  bit 的模加，相关差分分布表的字长可以是整除  $n$  的任意整数，并且其字长越大，搜索算法执行效率越高。考虑到相关差分分布表的存储，本文在实验过程中采用了 8 bit 的相关差分分布表。

## 4 SPECK 算法及其差分特征

### 4.1 SPECK 算法描述

SPECK 算法是美国国家安全局（NSA, National Security Agency）在 2013 年提出的一组轻量级分组密码算法<sup>[5]</sup>，共包含 10 个算法实例，其分组长度分别为 32 bit、48 bit、64 bit、96 bit 和 128 bit，相应的算法记为 SPECK32、SPECK48、SPECK64、SPECK96 和 SPECK128。

SPECK 算法采用了类似 Threefish 算法的结构，其轮函数仅仅使用了 3 种简单的运算：模加（addition）、循环移位、异或。令  $(L_{i-1}, R_{i-1})$  表示第  $i$  轮的输入， $(L_i, R_i)$  表示第  $i$  轮的输出，第  $i$  轮的轮密钥记为  $K_i$ ，则  $(L_i, R_i)$  可以通过下式计算

$$L_i = ((L_{i-1} \ggg \alpha) \oplus R_{i-1}) \oplus K_i, \quad R_i = (R_{i-1} \lll \beta) \oplus L_i \quad (20)$$

其中，当分组长度为 32 bit 时， $\alpha=7, \beta=2$ ，对于其

他分组长度,  $\alpha=8, \beta=3$ 。SPECK 算法的轮函数如图 2 所示。

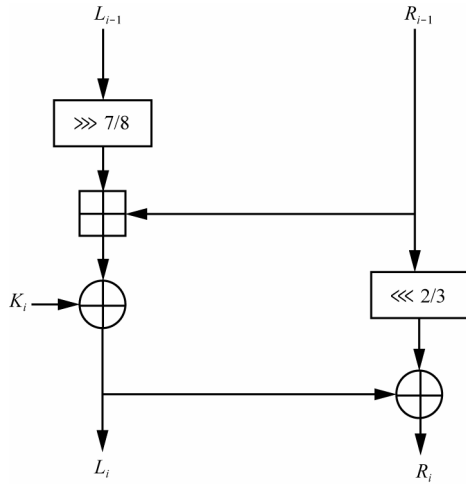


图 2 SPECK 算法的轮函数

## 4.2 SPECK 算法的差分特征

本文利用算法 2 搜索 SPECK32、SPECK48

和 SPECK64 的差分特征,所用的实验平台是 Intel Core i5 3.2 GHz, 采用 VS2010 软件平台, 得到的结果如下: 对于 SPECK32, 得到的最优差分特征包含 9 轮, 其差分概率为  $2^{-30}$ ; 对于 SPECK48, 得到的最优差分特征包含 11 轮, 其差分概率为  $2^{-45}$ ; 对于 SPECK64, 得到的最优差分特征包含 15 轮, 其差分概率为  $2^{-62}$ 。这些结果跟 Fu 等<sup>[21]</sup>在 FSE'16 上发表的结果相同, 但是本文的算法遍历了 SPECK32 和 SPECK48 的明文差分, 因此能够说明该结果就是 SPECK32 和 SPECK48 最好的差分特征, 而且本文给出了 SPECK32 和 SPECK48 抗差分分析的可证明的安全界。而对于 SPECK64, 由于受到计算能力的限制, 并没有遍历所有可能的明文差分, 尽管如此, 本文给出的结果依然是目前最优的。SPECK32、SPECK48 和 SPECK64 的最优差分特征的概率如表 3 所示, 其 9 轮、11 轮和 15 轮的最优差分特征如表 4 所示。

表 3 SPECK32、SPECK48 和 SPECK64 最优差分特征的概率

| 算法      | 轮数       |          |          |           |           |           |           |           |           |           |           |           |           |           |           |  |
|---------|----------|----------|----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|--|
|         | 2        | 3        | 4        | 5         | 6         | 7         | 8         | 9         | 10        | 11        | 12        | 13        | 14        | 15        | 16        |  |
| SPECK32 | $2^{-1}$ | $2^{-3}$ | $2^{-5}$ | $2^{-9}$  | $2^{-13}$ | $2^{-18}$ | $2^{-24}$ | $2^{-30}$ | $2^{-34}$ | —         | —         | —         | —         | —         | —         |  |
| SPECK48 | $2^{-1}$ | $2^{-3}$ | $2^{-6}$ | $2^{-10}$ | $2^{-14}$ | $2^{-19}$ | $2^{-26}$ | $2^{-33}$ | $2^{-40}$ | $2^{-45}$ | $2^{-49}$ | —         | —         | —         | —         |  |
| SPECK96 | $2^{-1}$ | $2^{-3}$ | $2^{-6}$ | $2^{-10}$ | $2^{-15}$ | $2^{-21}$ | $2^{-29}$ | $2^{-34}$ | $2^{-38}$ | $2^{-42}$ | $2^{-46}$ | $2^{-50}$ | $2^{-56}$ | $2^{-62}$ | $2^{-70}$ |  |

表 4 SPECK32、SPECK48 和 SPECK64 的 9 轮、11 轮和 15 轮最优差分特征

| 轮数 | SPECK32    |            |          | SPECK48    |            |           | SPECK64    |            |           |
|----|------------|------------|----------|------------|------------|-----------|------------|------------|-----------|
|    | $\Delta L$ | $\Delta R$ | $p$      | $\Delta L$ | $\Delta R$ | $p$       | $\Delta L$ | $\Delta R$ | $p$       |
| 0  | 56a        | 9000       |          | 1202       | 20002      |           | 924000c0   | 40104200   |           |
| 1  | 440a       | 408        | $2^{-5}$ | 10         | 100000     | $2^{-3}$  | 820200     | 1202       | $2^{-6}$  |
| 2  | 1080       | a0         | $2^{-4}$ | 0          | 800000     | $2^{-1}$  | 9000       | 10         | $2^{-4}$  |
| 3  | 83         | 203        | $2^{-4}$ | 800000     | 800004     | 1         | 80         | 0          | $2^{-2}$  |
| 4  | c          | 800        | $2^{-6}$ | 808004     | 808020     | $2^{-2}$  | 80000000   | 80000000   | 1         |
| 5  | 2000       | 0          | $2^{-3}$ | 8400a0     | 8001a4     | $2^{-4}$  | 80800000   | 80800004   | $2^{-1}$  |
| 6  | 40         | 40         | $2^{-1}$ | 608da4     | 608080     | $2^{-9}$  | 80008004   | 84008020   | $2^{-3}$  |
| 7  | 8040       | 8140       | $2^{-1}$ | 42003      | 2400       | $2^{-11}$ | 80808060   | a0848164   | $2^{-6}$  |
| 8  | 40         | 542        | $2^{-2}$ | 12020      | 20         | $2^{-5}$  | 40f24      | 4200401    | $2^{-13}$ |
| 9  | 8542       | 904a       | $2^{-4}$ | 200100     | 200000     | $2^{-3}$  | 20200008   | 1202000    | $2^{-8}$  |
| 10 | —          | —          | —        | 202001     | 202000     | $2^{-3}$  | 202001     | 202000     | $2^{-4}$  |
| 11 | —          | —          | —        | 210020     | 200021     | $2^{-4}$  | 210020     | 200021     | $2^{-2}$  |
| 12 | —          | —          | —        | —          | —          | —         | 800        | 800        | $2^{-1}$  |
| 13 | —          | —          | —        | —          | —          | —         | 808        | 4808       | $2^{-2}$  |
| 14 | —          | —          | —        | —          | —          | —         | 8004800    | 8020840    | $2^{-4}$  |
| 15 | —          | —          | —        | —          | —          | —         | 80a0808    | 481a4a08   | $2^{-6}$  |
| 概率 | $2^{-30}$  |            |          | $2^{-45}$  |            |           | $2^{-62}$  |            |           |

## 5 结束语

本文提出了一种自动化搜索 ARX 密码最优差分特征的算法,该算法是基于 Matsui 算法,通过引入相关差分分布表,解决了搜索过程中模加差分的快速计算问题,即给定模加的输入差分,通过查找相关差分分布表,可以快速计算出输出差分以及差分概率。

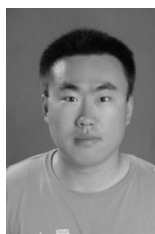
本文以 SPECK 算法中的 3 个实例 SPECK32、SPECK48 和 SPECK64 为测试平台说明该算法的有效性。使用本文提出的算法搜索 SPECK32、SPECK48 和 SPECK64 的差分特征,分别得到了 9 轮、11 轮和 15 轮的差分特征,其概率分别为  $2^{-30}$ 、 $2^{-45}$  和  $2^{-62}$ 。这些差分特征是目前能够找到的最好的差分特征,其中, SPECK32 和 SPECK48 的差分特征通过遍历明文差分得到,因此可以给出其抗差分分析的可证明的安全界。

本文提出的算法是一种通用的自动化搜索 ARX 密码差分特征的算法,尽管仅以 SPECK 算法作为测试平台,但是此算法可以用于搜索其他 ARX 密码的差分特征,作者正在研究使用本文算法搜索其他 ARX 密码的差分特征。

## 参考文献:

- [1] SHIMIZU A, MIYAGUCHI S. Fast data encipherment algorithm FEAL[C]//International Conference on Theory and Application of Cryptographic Techniques.c1987:267-278.
- [2] WHEELER D J, NEEDHAM R M. Tea, a tiny encryption algorithm[M]// Fast Software Encryption. Berlin: Springer, 2015: 363-366.
- [3] RIVEST R L. The RC5 encryption algorithm[C]//Fast Software Encryption: Second International Workshop. c1994:14-16.
- [4] HONG D, SUNG J, HONG S, et al. HIGHT: a new block cipher suitable for low-resource device[M]//Cryptographic Hardware and Embedded Systems-CHES 2006. Berlin: Springer, 2006: 46-59.
- [5] BEAULIEU R, SHORS D, SMITH J, et al. The SIMON and SPECK families of lightweight block ciphers[R]. IACR Cryptology ePrint Archive, 2013.
- [6] BERNSTEIN D J. The salsa20 family of stream ciphers[R]. New Stream Cipher Designs - The eSTREAM Finalists, 2008.
- [7] RIVEST R L. The MD4 message digest algorithm[C]//CRYPTO. c1990:303-311.
- [8] FERGUSON N, LUCKS S, SCHNEIER B, et al. The skein hash function family[R]. Submission to NIST(Round 3), 2010.
- [9] AUMASSON J P, HENZEN L, MEIER W, et al. Sha-3 proposal blake[R]. Submission to NIST(Round 3), 2010.
- [10] BIHAM E, SHAMIR A. Differential cryptanalysis of des-like cryptosystems[J]. Journal of Cryptology, 1991, 1(4): 3-72.
- [11] AOKI K, KOBAYASHI K, MORIAI S. Best differential characteristic search of FEAL[M]//Fast Software Encryption. Berlin : Springer, 1997:41-53.
- [12] BANNIER A, BODIN N, FILIOL E. Automatic search for a maximum probability differential characteristic in a substitution-permutation network[C]//International Conference on System Sciences, IEEE Computer Society. c2015: 5165-5174.
- [13] BIRYUKOV A, NIKOLIC I. Automatic search for related-key differential characteristics in byte-oriented block ciphers : application to AES, camellia, khazad and others[C]//International Conference on the Theory and Applications of Cryptographic Techniques. c2010: 322-344.
- [14] BIRYUKOV A, NIKOLIC I. Search for related-key differential characteristics in des-like ciphers[C]// International Conference on FAST Software Encryption. c2011:18-34.
- [15] BOUILLAGUET C, DERBEZ P, FOUQUE P. Automatic search of attacks on round-reduced AES and applications[R]. IACR Cryptology ePrint Archive, 2012.
- [16] MATSUI M. On correlation between the order of s-boxes and the strength of DES [M]// Advances in Cryptology — EUROCRYPT'94. Berlin: Springer, 1994:366-375.
- [17] MOUHAN N, WANG Q, GU D, et al. Differential and linear cryptanalysis using mixed-integer linear programming[C]//7th International Conference on Information Security and Cryptology. c2011.
- [18] SUN S W, HU L, WANG P, et al. Automatic security evaluation and (related-key) differential characteristic search: application to simon, present, lblock, DES(L) and other bit-oriented block ciphers[C]//The 20th International Conference on the Theory and Application of Cryptology and Information Security. c2014.
- [19] BIRYUKOV A, VELICHKOV V. Automatic search for differential trails in ARX ciphers[M]//Topics in Cryptology – CT-RSA 2014. Berlin: Springer, 2014:227-250.
- [20] BIRYUKOV A, ROY A, VELICHKOV V. Differential analysis of block ciphers SIMON and SPECK[M]//Fast Software Encryption. Berlin: Springer Heidelberg, 2014:546-570.
- [21] FU K, WANG M Q, GUO Y, et al. MILP-based automatic search algorithms for differential and linear trails for speck[C]//Fast Software Encryption, 23rd International Workshop. c2016.

## 作者简介:



刘正斌(1985-),男,山东青岛人,中国科学院信息工程研究所博士生,主要研究方向为密码学。