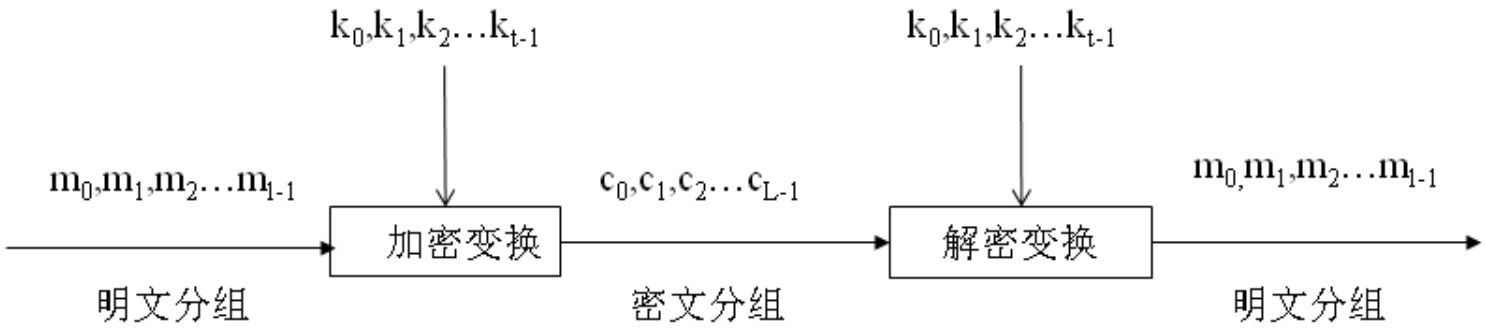
应用密码学（第五讲）
— 分组密码

林东岱

信息安全国家重点实验室

2022年9月

分组密码基本概念



- 分组长度：通常为64比特或128比特
- 密钥长度：64比特、128比特或256比特
- 加密特征：密文不随时间的变化而变化

什么是分组密码?

$$E(*, K) : F_2^m \rightarrow F_2^n$$

明文空间: F_2^m

密文空间: F_2^n

密钥空间: $K \in F_2^k$

什么是分组密码?

- m<n: 有数据扩展的分组密码
- m>n: 带数据压缩的分组密码
- 一般情况下: m=n, 这是相应的分组是一个置换

- 分组密码的特点：速度快
- 与流密码的区别

设计原则

- 扩散原则：

  设计的密码算法应使明文或密钥中的每一位影响密文中的许多位；扩散的目的是希望密文中的任一比特都尽可能与明文、密钥相关联，或者明文和密钥的任何位上的值的改变都会在某种程度上影响到密文比特的值，以此防止将密钥分解成若干个孤立的小部分，然后各个击破。

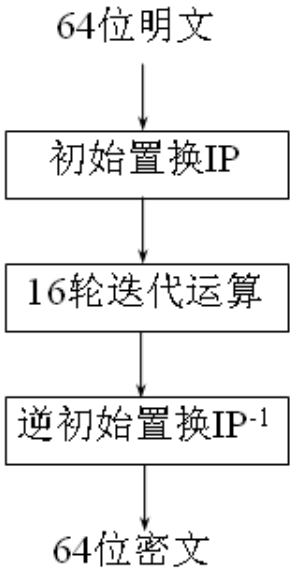
- 混淆原则：

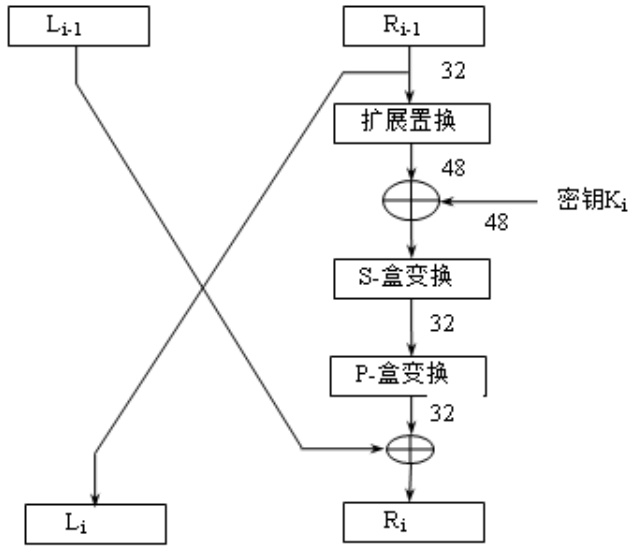
  设计的密码算法应使得密钥和明文以及密文之间的依赖关系变得尽可能复杂。使得对手即使获取了关于密文的一些统计特性，也无法推测密钥。可以使用复杂的非线性代替变换来达到较好的混淆效果。

分组密码DES

- 全称：Data Encryption Standard
- 世界上第一个加密标准
- 密钥长度：64比特（其中8位是校验位）
- 分组长度：64比特
- IBM公司1974年设计，原名Lucifer
- 1977年1月15日成为美国联邦数据加密标准
- 1981年成为金融工业标准
- 2001年11月正式被数据加密标准AES取代。

DES加密算法概述





算法的形式化描述：

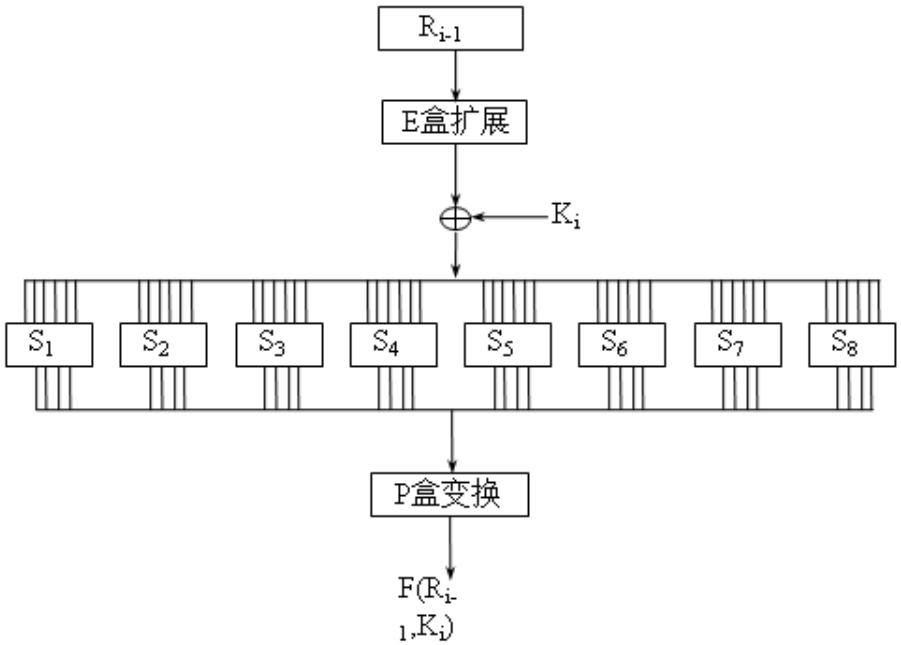
$L_i=R_{i-1}$

$R_i=L_{i-1} \oplus F(R_{i-1},K_i) \quad i=1,2,3\ldots\ldots16$

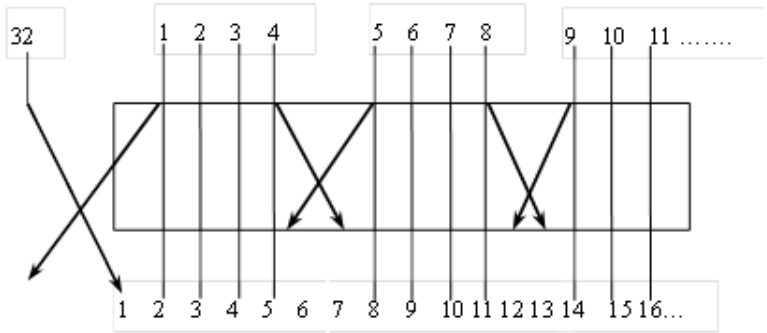
初始置换

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

加密函数 F



扩展置换



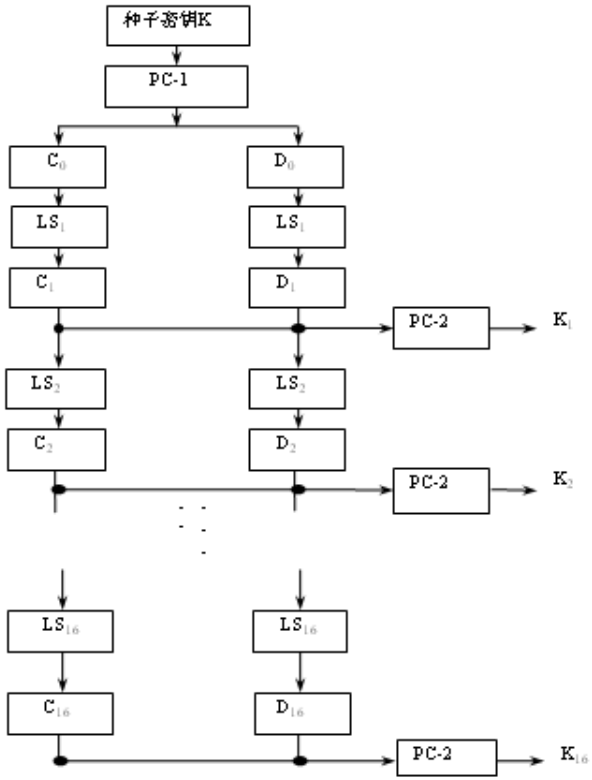
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

表 5.3 DES 的 S 盒

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	
2	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	S_1
3	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0	
4	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	
1	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	
2	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	S_2
3	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	
4	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	
1	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	
2	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	S_3
3	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7	
4	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	
1	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15	
2	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	S_4
3	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4	
4	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14	
1	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	
2	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	S_5
3	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	
4	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	
1	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11	
2	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	S_6
3	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6	
4	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13	
1	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1	
2	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	S_7
3	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2	
4	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12	
1	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7	
2	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	S_8
3	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8	
4	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11	

子密钥的生成



如果i=1,2,9,16，循环左移动1位，否则循环移动2位。

子密钥生成

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

PC-1置换

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

PC-2置换

几条S盒的设计准则

（1）S盒的每一行是整数0,1,2……15的一个置换；

（2）没有一个S盒是它输入的线性或仿射函数；

（3）改变S盒的一个输入比特至少要引起两比特的输出改变；

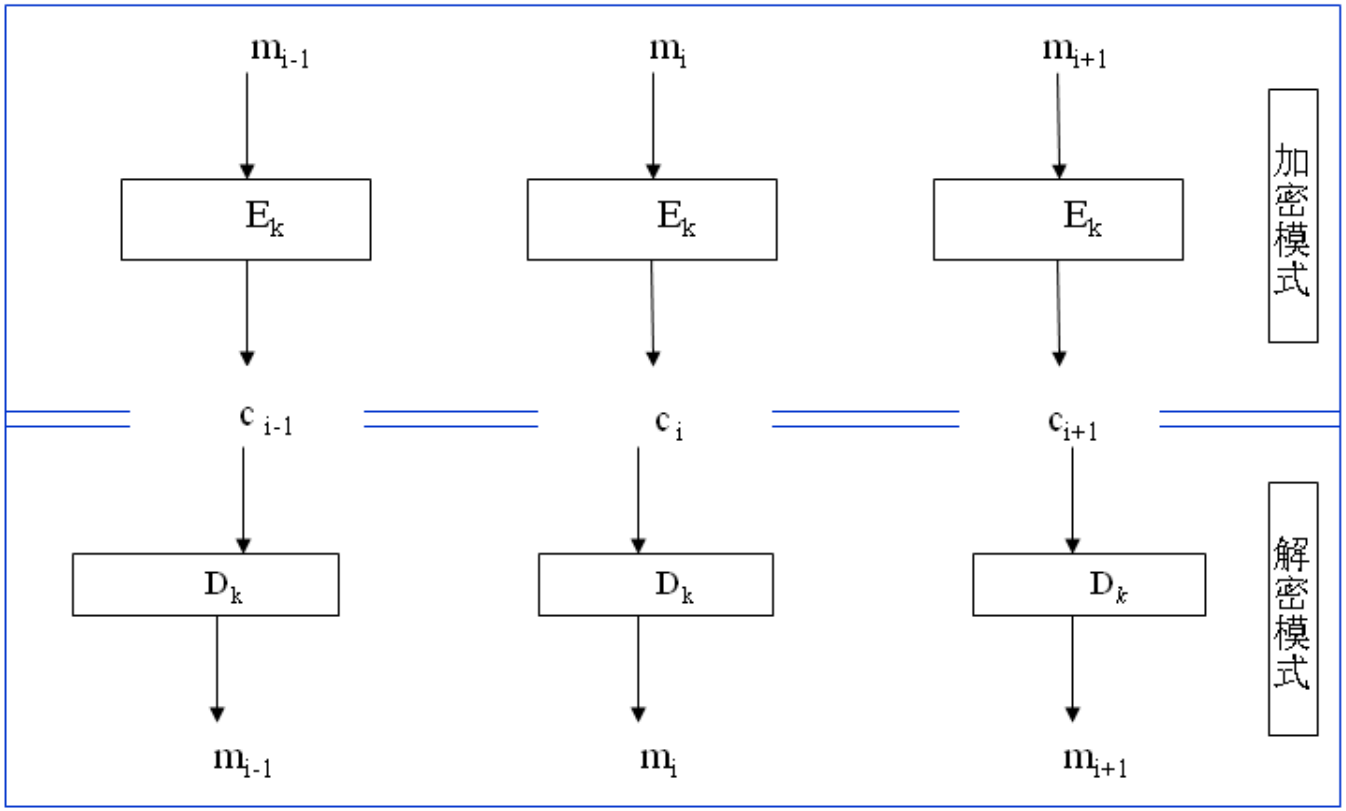
（4）对任何一个S盒和任何一个输入x(x为6位的比特串)，S(x)和S(x 001100)至少有两比特不同；

（5）对任何一个S盒，如果两个输入的前两位不同，而最后两位相同，两个输出必须不同；

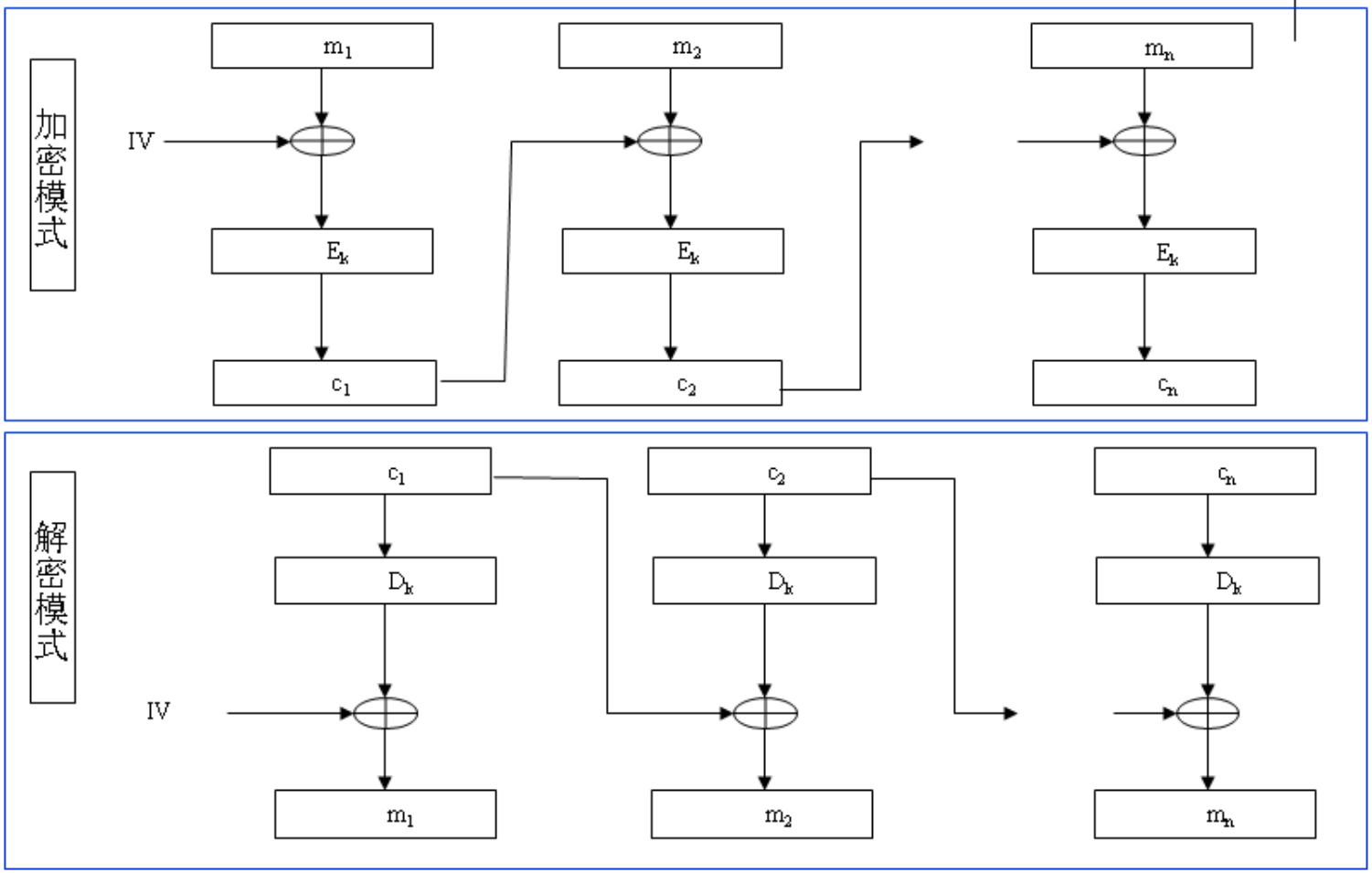
（6）对任何一个S盒，如果固定一个输入比特保持不变而使其他5比特输入变化，我们观察一个固定输出比特的值，使这个输出比特为0的输入的数目与使这个输出比特为1的输入的数目总数接近相等；

分组密码的工作模式

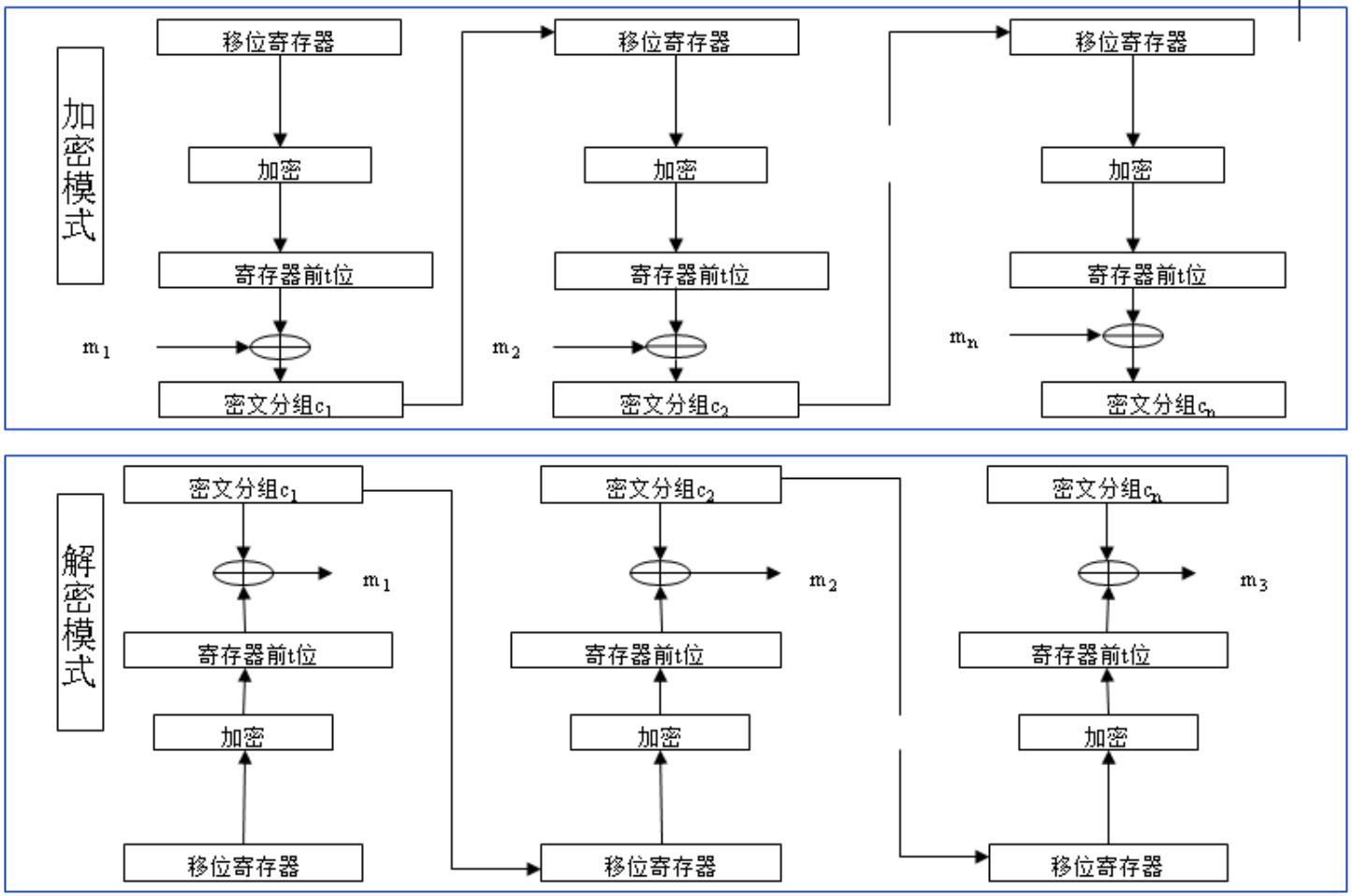
- 电子密码本模式(ECB)



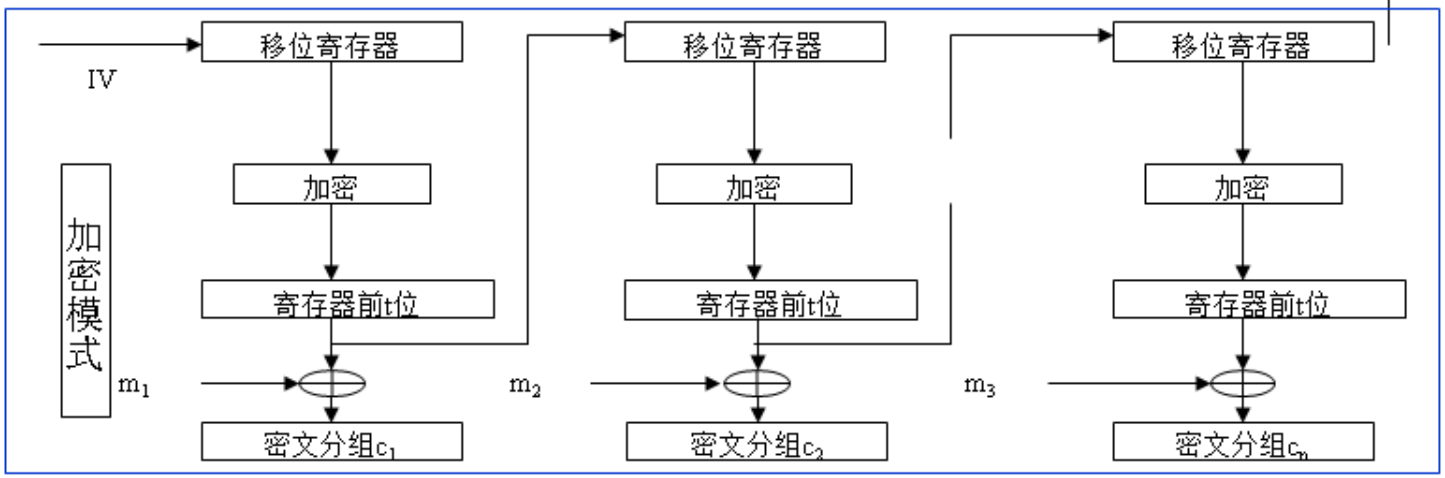
- 密码分组链接模式(CBC)

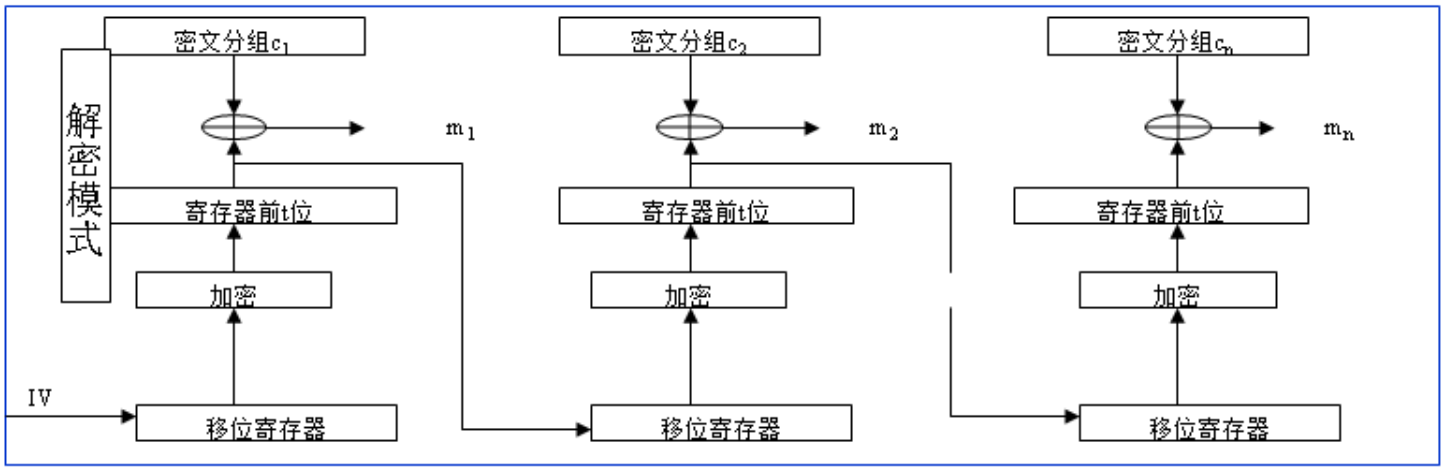


- 密码反馈模式(CFB)

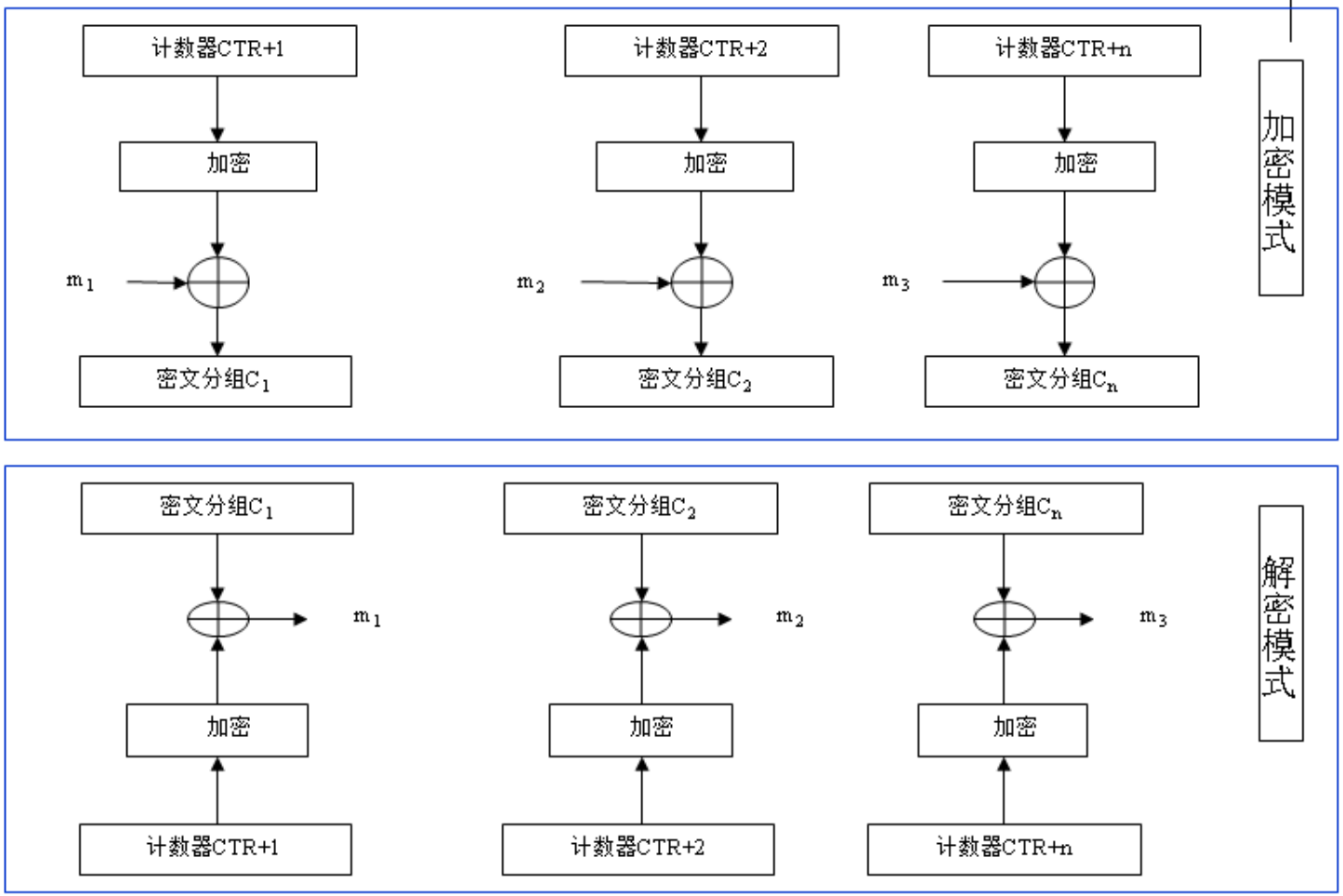

加密模式
移位寄存器
加密
寄存器前t位
m_1
密文分组c_1
移位寄存器
加密
寄存器前t位
m_2
密文分组c_2
移位寄存器
加密
寄存器前t位
m_n
密文分组c_n
解密模式
密文分组c_1
m_1
寄存器前t位
加密
移位寄存器
密文分组c_2
m_2
寄存器前t位
加密
移位寄存器
密文分组c_n
m_3
寄存器前t位
加密
移位寄存器

- 输出反馈模式(OFB)


移位寄存器
IV
加密
加密模式
寄存器前t位
m_1
密文分组c_1
移位寄存器
加密
寄存器前t位
m_2
密文分组c_2
移位寄存器
加密
寄存器前t位
m_3
密文分组c_n


密文分组c_1
解密模式
寄存器前t位
加密
IV
移位寄存器
m_1
密文分组c_2
寄存器前t位
加密
移位寄存器
m_2
密文分组c_n
寄存器前t位
加密
移位寄存器
m_n

- 记数器模式(CTR)


计数器CTR+1
加密
m_1
密文分组C_1
计数器CTR+2
加密
m_2
密文分组C_2
计数器CTR+n
加密
m_3
密文分组C_n
加密模式
密文分组C_1
m_1
加密
计数器CTR+1
密文分组C_2
m_2
加密
计数器CTR+2
密文分组C_n
m_3
加密
计数器CTR+n
解密模式

AES高级加密标准

AES(Advanced Encryption Standard)是美国新一代对称分组密码算法标准，于2001年11月26日 公布。 开发AES算法的是两位来自比利时的密码专家：Proton World International的Joan Daemen 博士和Katholieke Universiteit Leuven电子工程系(ESAT)的Vincent Rijmen博士。 在选为标准之前，AES称为Rijndael算法。它采用的是代替置换网络。每一轮由三层组成， 线性混合层确保多轮之后的高度扩散，非线性层由16个S-盒并置而成，起到混淆的作用；密钥加 层则简单地将子密钥异或到中间状态上。S-盒选取的是有限域$GF(2^8)$（$m(x) = x^8 + x^4 + x^3 + x + 1$）中的乘法逆运算， 它的差分均匀性和线性偏差都达到了最佳。

AES算法描述

AES是一迭代分组密码算法，支持长为128, 192, 256位的密钥和128位的分组 （实际上，算法的分组长度和密钥长度可以扩展到任何32位的倍数, 因此以下我们并不局限于分组长度为128）。 加密轮数依赖于分组长度和密钥长度。除最后一轮外， 每一轮变换由字节替代(ByteSub)、 行移位(ShiftRow)、列混合(MixColumn)和轮密钥加(AddRoundKey)四种不同的变换复合而成。 最后一轮不进行列混合变换。每一变换的中间结果称为算法的一个状态。

每一状态被表示成$4 \times N_b$行的矩阵，其中矩阵的列数N_b等于分组的长度除以32， 矩阵的每个元素是一个8位字节。 同样，密钥被表示成一个$4 \times N_k$行的矩阵，其中N_k等于密钥的 长度除以32。

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{3,5}$

$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$
$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$

Figure: 状态阵列($N_b = 6$)与密钥阵列($N_k = 4$)举例

AES算法的输入与输出是一个长为$4Nb$的8比特字节的序列，依次对应状态阵列中的 元素$a_{0,0}, a_{1,0}, a_{2,0}, a_{3,0}, a_{0,1}, a_{1,1}, a_{2,1}, a_{3,1}, \cdots$.加密之后，输出被以同样的顺序从状态阵列中取出。 密钥是长为$4N_k$的8比特字节的序列, 依次对应密钥阵列中的
$k_{0,0}, k_{1,0}, k_{2,0}, k_{3,0}, k_{0,1}, k_{1,1}, k_{2,1}, k_{3,1}, \cdots$.

加密轮数N_r依赖于分组的长度和密钥的长度，可用下列公式算出(见表1)：

$$N_r = \max(N_b, N_k) + 6.$$

Table: 轮数是分组和密钥长度的函数

N_r	$N_b = 4$	$N_b = 6$	$N_b = 8$
$N_k = 4$	10	12	14
$N_k = 6$	12	12	14
$N_k = 8$	14	14	14

除最后一轮外， 每一轮变换依次由字节替代(ByteSub)、 行移位(ShiftRow)、列混合(MixColumn)和轮密钥加(AddRoundKey)四种不同的变换复合而成。 最后一轮不进行列混合变换。如图2。

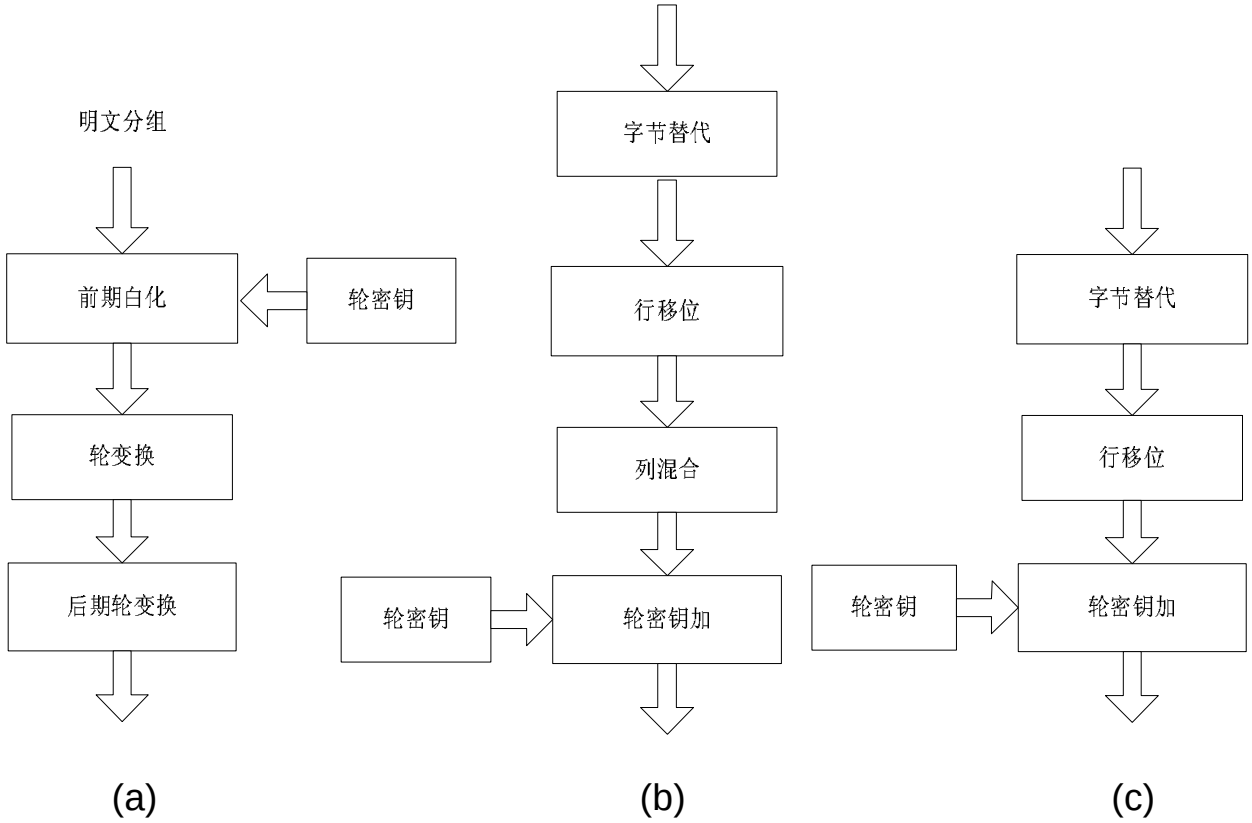


Figure: (a) AES加密流程 (b) 轮变换 (c) 末轮变换

我们知道有限域F_{2^8}中的元素可以用许多不同的方式来表示，其中一种比较常见的表示方式就是它的多项式表示.
设$m(x) = x^8 + x^4 + x^3 + x + 1$, 则$m(x)$是$F_2$上的不可约多项式，从而有$F_{2^8} \cong F_2[x]/(m(x))$. 因此$F_{2^8}$中的元素可以表示成次数小于8的$F_2$上的多项式。而域中元素的加法和乘法就是多项式模$m(x)$的 加法和乘法。

由于一个多项式完全由它的系数决定，因此我们可以在8位的字节与次数小于8的F_2上的多项式，因此也就是F_{2^8}中 的元素之间建立一个一一对应如下：

$$\phi : b_7b_6b_5b_4b_3b_2b_1b_0 \longleftrightarrow b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0,$$

其中$b_i \in F_2, 0 \le i \le 7$. 因此$F_{2^8}$中的一个元素可以看成一个字节，反之一个字节也可以看成F_{2^8}中的一个元素。 在这种对应下，我们可以对两

个字节作加法运算、乘法运算，甚至对非零字节($\neq 0x00$)做求逆运算，例如

$$(x^6+x^4+x^2+x+1)+(x^7+x+1)=x^7+x^6+x^4+x^2 \quad \text{(多项式表示)}$$
$$01010111 \oplus 10000011 = 11010100 \quad \text{(二进制表示)}$$
$$0x57 \oplus 0x83 = 0xd4 \quad \text{(十六进制表}$$

$$(x^6+x^4+x^2+x+1)\cdot(x^7+x+1)=x^7+x^6+1 \quad \text{(多项式表示)}$$
$$01010111 \otimes 10000011 = 11000001 \quad \text{(二进制表示)}$$
$$0x57 \otimes 0x83 = 0xC1 \quad \text{(十六进制表示)}$$

$$(x^7+x^6+x^5+x^4+x)^{-1}=x+1 \quad \text{(多项式表示)}$$
$$(11110010)^{-1} = 00000011 \quad \text{(二进制表示)}$$
$$(0xf2)^{-1} = 0x03 \quad \text{(十六进制表示)}$$

字节替代变换是一个非线性的字节变换，独立地作用于每一状态的字节之上(如图3)。代替表（或称 S-盒）是由下列两个变换复合而成的可逆变换：

1. 首先，对每一状态字节在F_{2^8}中求乘法逆。这里，$0x00$的乘法逆定义为$0x00$自己。
2. 然后，对每一状态字节作二元域$GF(2)$上的如下仿射变换：

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

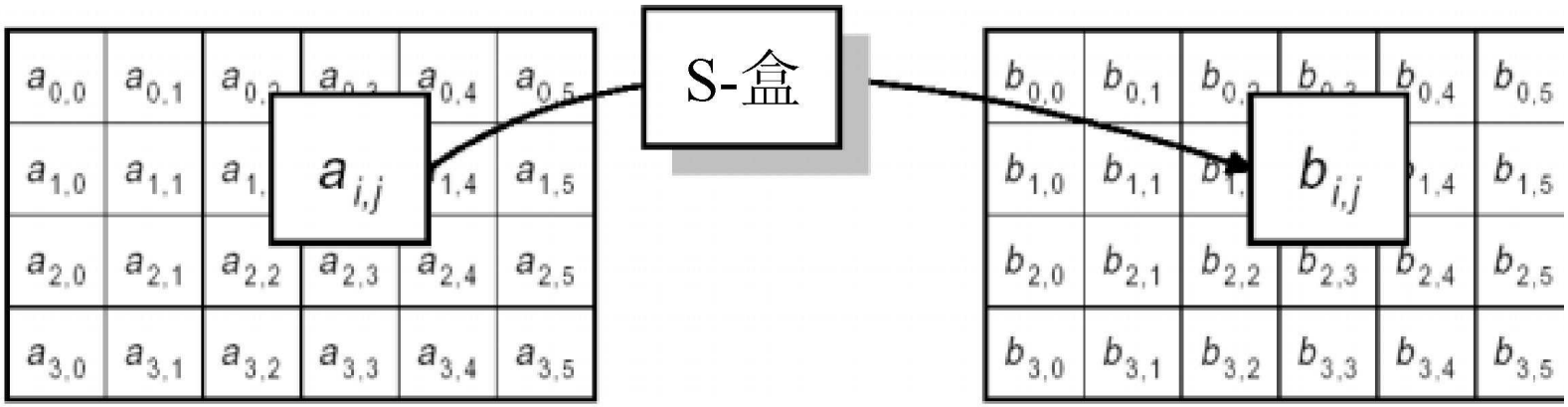


Figure: 字节替代变换对一个特定状态的作用

字节替代变换的逆变换定义为对一字节先做上述仿射变换的逆变换，然后再作有限域F_8 中的乘法逆运算。

行移位变换将状态阵列的第一行、第二行、第三行和第四行分别向左循环移位C_0字节、 C_1字节、C_2字节和C_3字节。C_0恒为0, C_1、C_2和C_3的值由分组 长度N_b按表2确定如下：

Table: 移位量与分组长度的关系

N_b	C_1	C_2	C_3
4	1	2	3
6	1	2	3
8	1	3	4

行移位变换的逆变换则将状态阵列的第二、第三和第四行分别向右移位C_1, C_2和 C_3字节。

在列混合变换中，状态阵列的每一列被看成$GF(2^8)$上的一个多项式，并模x^4+1乘上 一个固定的多项式（如图4）：

$$c(x) = 03x^3 + 01x^2 + 01x + 02,$$

即对状态阵列的每一列做如下变换：

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

用$GF(2^8)$上的多项式可表示为（如图4）：

$$b_3x^3 + b_2x^2 + b_1x + b_0 \equiv (a_3x^3 + a_2x^2 + a_1x + a_0) \otimes c(x),$$

其中$\otimes$表示$GF(2^8)$上模$x^4 + 1$的多项式乘法运算。

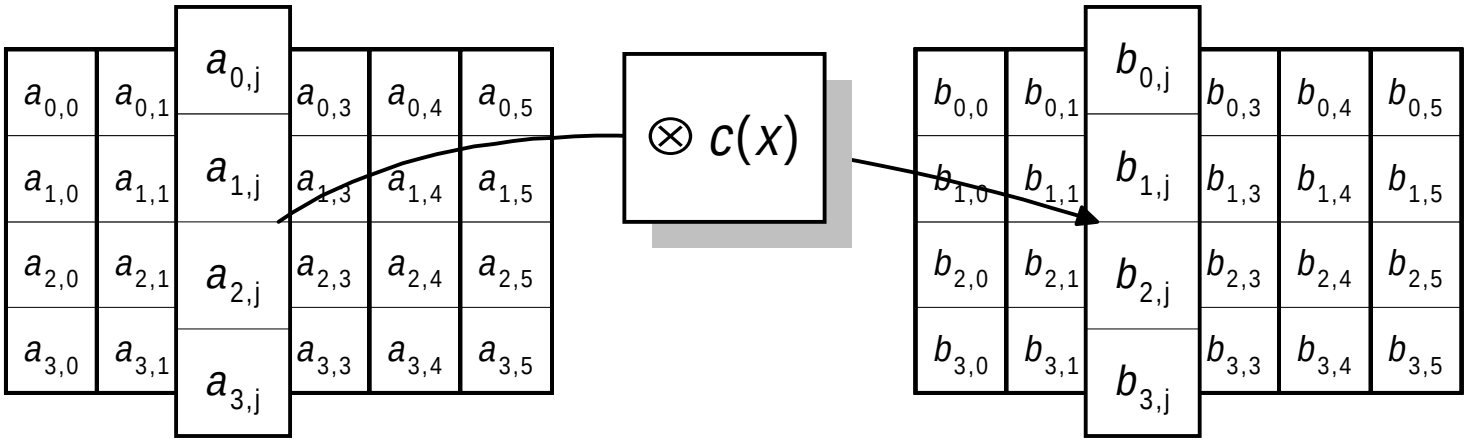


Figure: 列混合变换

事实上，$c(x)$是和x^4+1互素的多项式，从而存在多项式$d(x)$, 使

$$d(x)c(x) \equiv 01 \bmod x^4+1,$$

因此上述列混合变换是可逆的。容易算出

$$d(x) = 0Bx^3 + 0Dx^2 + 09x + 0E.$$

所以列混合变换的逆变换就是将状态阵列的每一列看成$GF(2^8)$上的一个多项式，并模x^4+1乘上 多项式$d(x)$而得到的变换，即：

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 0E & 0B & 09 & 0D \\ 09 & 0E & 0B & 09 \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 0D & 0E \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}.$$

用$GF(2^8)$上的多项式可表示为：

$$a_3x^3+a_2x^2+a_1x+a_0 \equiv (b_3x^3+b_2x^2+b_1x+b_0) \otimes d(x).$$

轮密钥是按照密钥编排规则从种子密钥得到的子密钥，轮密钥的长度和分组的长度相同。 轮密钥加变换只是将轮密钥和状态简单地进行位异或。轮密钥加变换的逆就是轮密钥加 变换自身。图5描述了分组长度为$N_b=128$时的轮密钥加变换， 其中$b_{ij}=a_{ij} \oplus k_{ij}$, $\oplus$表示字节按位的异或运算。

$$\begin{array}{|c|c|c|c|} \hline a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ \hline a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ \hline a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ \hline a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \\ \hline \end{array} \oplus \begin{array}{|c|c|c|c|} \hline k_{0,0} & k_{0,1} & k_{0,2} & k_{0,3} \\ \hline k_{1,0} & k_{1,1} & k_{1,2} & k_{1,3} \\ \hline k_{2,0} & k_{2,1} & k_{2,2} & k_{2,3} \\ \hline k_{3,0} & k_{3,1} & k_{3,2} & k_{3,3} \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ \hline b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ \hline b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ \hline b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \\ \hline \end{array}$$

Figure: 轮密钥加

每一轮的轮密钥都是从种子密钥通过密钥编排得到。密钥编排包括密钥扩展和轮密钥挑选 两步。密钥扩展首先将密钥扩展成长为$N_b(N_r+1)$的扩展密钥，然后从扩展密钥中依次选取 第一个N_b字长作为第一轮的密钥，第二个N_b字长作为第二轮的密钥，依次类推。

扩展密钥是4字节长的字的线性阵列，表示为$W[i], 0 \le i < N_b(N_r+1)$. 前$N_k$个字就是种子密钥 本身，其余的字则

由序号较小的字递归定义。递归定义的方式按$N_k \leq 6$和$N_k > 6$分成 两种情况：

第一种情况: $N_k \leq 6$

(1) $i \not\equiv 0 \bmod N_k$时, $W[i]$定义为$W[i-1]$与$W[i-N_k]$进行异或运算的结果。

(2) $i \equiv 0 \bmod N_k$时, 先将$W[i-1]$左移一个字节，然后做字节替代变换ByteSub，再先后与 $((02)^i$,00,00,00)和与$W[i-N_k]$进行异或运算，其结果定义为$W[i]$。

第二种情况: $N_k > 6$

(1) $i \not\equiv 0 \bmod N_k$且$i \not\equiv 4 \bmod N_k$时, 定义$W[i]$为$W[i-1]$与$W[i-N_k]$进行异或运算的结果。

(2) $i \not\equiv 0 \bmod N_k$但$i \equiv 4 \bmod N_k$时, 定义$W[i]$为先将$W[i-1]$做字节替代变换ByteSub， 然后再与$W[i-N_k]$进行异或运算的结果。

(3) $i \equiv 0 \bmod N_k$时, 先将$W[i-1]$左移一个字节，然后做字节替代变换ByteSub，再先后与 $((02)^i$,00,00,00)和与$W[i-N_k]$进行异或运算，其结果定义为$W[i]$。

SMS4分组密码算法

SMS4是用于WAPI（WLAN Authentication and Privacy Infrastructure）的分组密码算法，是国内官方公布的第一个商用密码算法。

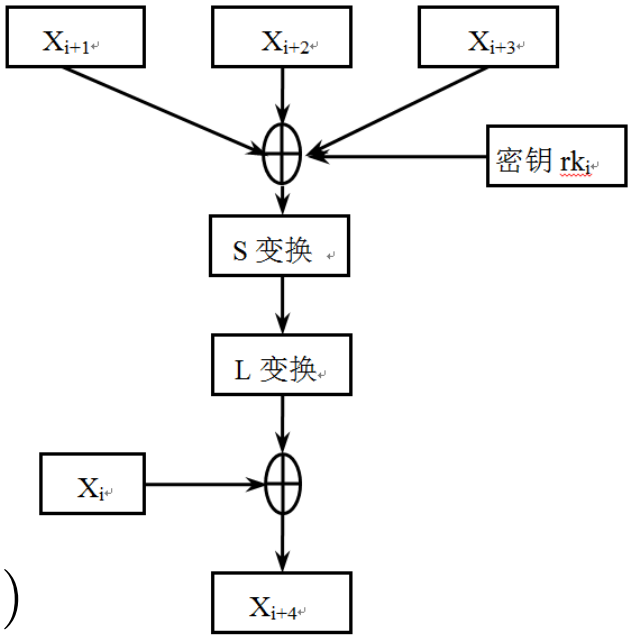
SMS4算法是一个分组算法。SMS4算法的分组长度为128比特，密钥长度为128比特。

加密算法与密钥扩展算法都采用32轮非线性迭代结构。解密算法与加密算法的结构相同， 只是轮密钥的使用顺序相反，解密轮密钥是加密轮密钥的逆序。

算法描述

SMS4算法以字为单位进行加密处理，一次迭代运算称为一轮变换，假设明文的输入为(X_0, X_1, X_2, X_3)，密文输出为(Y_0, Y_1, Y_2, Y_3)。SMS4一轮迭代当前的输入为$(X_i, X_{i+1}, X_{i+2}, X_{i+3})$，本轮的轮密钥为$rK_i$，则一轮的加密变换为：

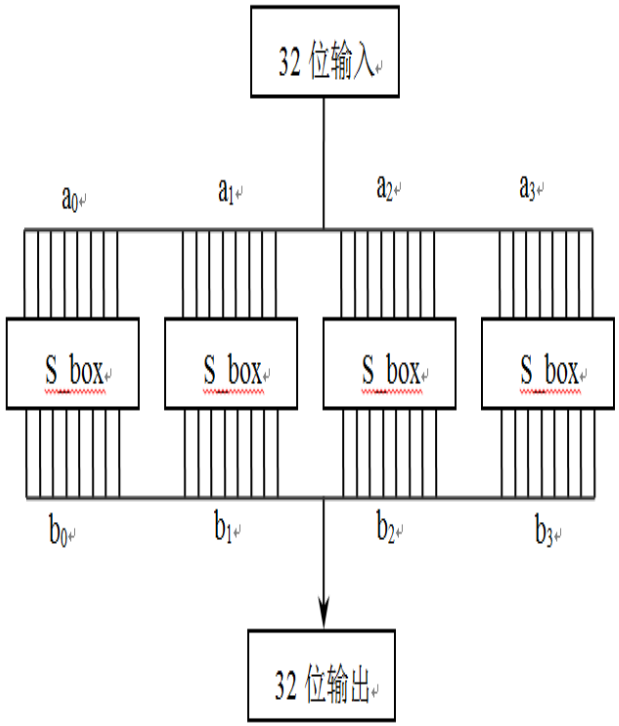
$$\begin{aligned} X_{i+4} &= F(X_i, X_{i+1}, X_{i+2}, X_{i+3}, rK_i) \\ &= X_i \oplus T(X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \oplus rK_i) \end{aligned}$$



SMS4的单轮加密

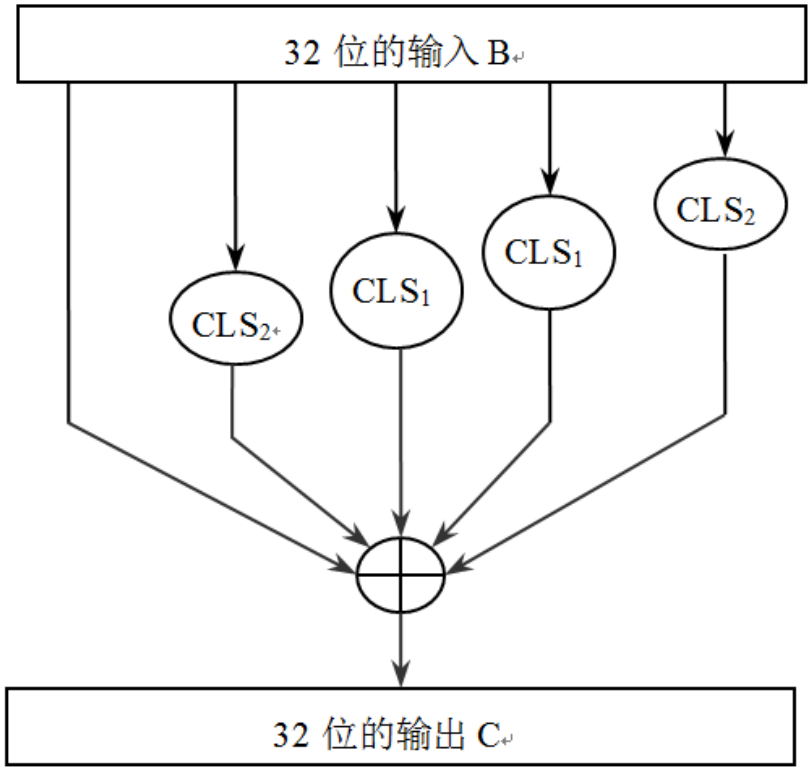
SMS4算法共需要32轮迭代，第29轮，30轮，31轮和32轮的迭代输出X_{32}, X_{33}, X_{34}和X_{35} 经过一个R变换$R(X_{32}, X_{33}, X_{34}, X_{35}) = (X_{35}, X_{34}, X_{33}, X_{32}) = (Y_0, Y_1, Y_2, Y_3)$，$Y_0, Y_1, Y_2, Y_3$就是加密的密文。其中$R$变换为反序变换：$R(A_0, A_1, A_2, A_3) = (A_3, A_2, A_1, A_0)$ ，其中A_i是32位的字。

S盒变换：SMS4的S盒输入为8位，输出为8位。SMS4将32位的输入分成4个字节，四个字节分别经过S盒。



	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	d6	90	e9	fe	cc	e1	3d	b7	16	b6	14	c2	28	fb	2c	05
1	2b	67	9a	76	2a	be	04	c3	aa	44	13	26	49	86	06	99
2	9c	42	50	f4	91	ef	98	7a	33	54	0b	43	ed	cf	ac	62
3	e4	b3	1c	a9	c9	08	e8	95	80	df	94	fa	75	8f	3f	a6
4	47	07	a7	fc	f3	73	17	ba	83	59	3c	19	e6	85	4f	a8
5	68	6b	81	b2	71	64	da	8b	f8	eb	0f	4b	70	56	9d	35
6	1e	24	0e	5e	63	58	d1	a2	25	22	7c	3b	01	21	78	87
7	d4	00	46	57	9f	d3	27	52	4c	36	02	e7	a0	c4	c8	9e
8	ea	bf	8a	d2	40	c7	38	b5	a3	f7	f2	ce	f9	61	15	a1
9	e0	ae	5d	a4	9b	34	1a	55	ad	93	32	30	f5	8c	b1	e3
a	1d	f6	e2	2e	82	66	ca	60	c0	29	23	ab	0d	53	4e	6f
b	d5	db	37	45	de	fd	8e	2f	03	ff	6a	72	6d	6c	5b	51
c	8d	1b	af	92	bb	dd	bc	7f	11	d9	5c	41	1f	10	5a	d8
d	0a	c1	31	88	a5	cd	7b	bd	2d	74	d0	12	b8	e5	b4	b0
e	89	69	97	4a	0c	96	77	7e	65	b9	f1	09	c5	6e	c6	84
f	18	f0	7d	ec	3a	dc	4d	20	79	ee	5f	3e	d7	cb	39	48

L线性变换：设输入为B，B为一个32位的字，输出为C，则L变换为：C=L(B)=B$\oplus$(B$<<<$ 2) $\oplus$(B$<<<$ 10) $\oplus$(B$<<<$ 18) $\oplus$(B$<<<$ 24),$<<<$表示32比特的字循环左移.



密钥扩展

SMS4算法的解密与加密变换结构相同，只是使用轮密钥的顺序不同，如果加密的时候使用的密钥顺序为：$(rK_0, rK_1, rK_2, \cdots, rK_{31})$，则解密时使用的顺序为$(rK_{31}, rK_{30}, rK_{29}, \cdots, rK_0)$。 SMS4的加密密钥为128位，但每轮迭代的密钥为32位，由于算法需要迭代32轮，所以共需要32个子密钥， 则需要从128位的用户密钥扩展出32个子密钥。其密钥扩展算法如下：

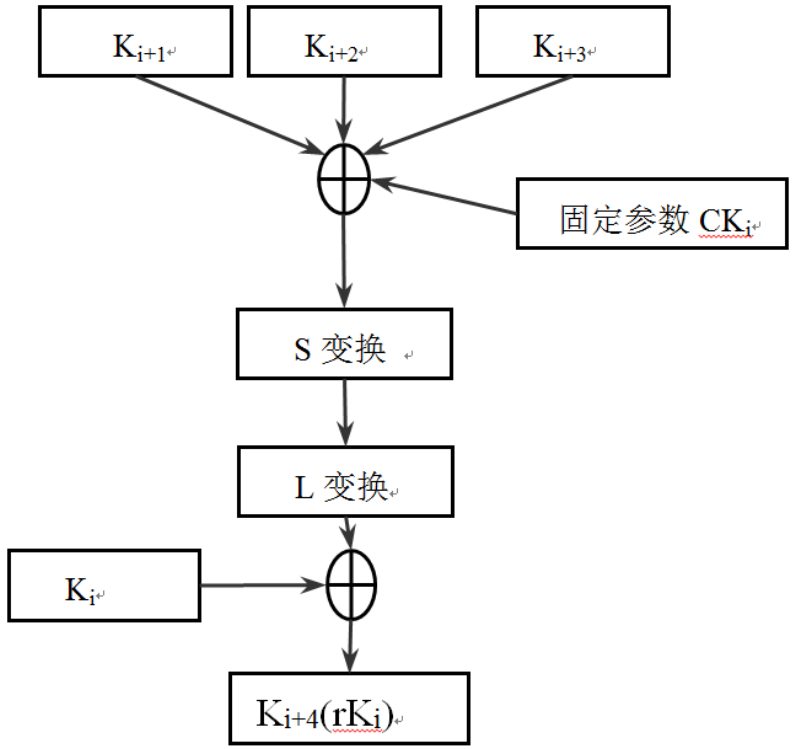
首先，(K0,K1,K2,K3)=(MK0 $\oplus$ FK0, MK1 $\oplus$ FK1, MK2 $\oplus$ FK2, MK3 $\oplus$ FK3)，其中，FKi为系统参数。

然后，
对i=0,1,2,$\cdots$,31，rKi=Ki+4=Ki $\oplus$ T'(Ki+1 $\oplus$ Ki+2 $\oplus$ Ki+3 $\oplus$ CKi)，其中T'变换与加密算法中的迭代变换基本相同， 只是线性变换L变为：L(B)=B $\oplus$ (B$<<<$ 13) $\oplus$ (B$<<<$ 23)，其中CKi(i=0,1,2,$\cdots$,31)为固定参数.

设加密密钥为$MK = (MK_0, MK_1, MK_2, MK_3)$, MK_i为32位的字，i=0,1,2,3；则轮密钥的生成算法为:

SMS4 的系统参数

FK_0	A3B1BAC6
FK_1	56AA3350
FK_2	677D9197
FK_3	B27022DC



SMS4 算法轮密钥的生成

固定参数 CK_i

00070e15	1c232a31	383f464d	545b6269
70777e85	8c939aa1	a8afb6bd	c4cbd2d9
e0e7eef5	fc030a11	181f262d	343b4249
50575e65	6c737a81	888f969d	a4abb2b9
c0c7ced5	dce3eaf1	f8ff060d	141b2229
30373e45	4c535a61	686f767d	848b9299
a0a7aeb5	bcc3cad1	d8dfe6ed	f4fb0209
10171e25	2c333a41	484f565d	646b7279

密码分析技术

5.7 分组密码分析技术

密码分析和密码设计是相互对立、相互依存的。伴随着对任何一种密码的分析，分析者千方百计从该密码算法中寻找漏洞和缺陷，进而进行攻击。对现代分组密码的分析更是如此，如 DES 自从诞生以来，对它的分析工作就没停止过。归纳起来，分组密码的分析方法主要有以下几种类型：

（1）穷举攻击
（2）线性分析方法
（3）差分分析方法
（4）相关密钥密码分析方法
（5）中间相遇攻击

在本节中我们典型的以一个 SPN 加密网络为基础，介绍两种典型的分析方法：线性分析方法和差分分析方法。

5.7.1 代换－置换网络

在本小节将介绍密码设计和分析中的基本的代换－置换(SPN)加密网络，如图 6.40 所示。图 6.40 所示加密算法，算法的输入为 16 比特的数据块，并且重复四次相同的操作处理数据块。每一轮包括（1）S-box 置换（2）比特变换（3）密钥混合。这个基本结构称为 Feistel 网络，最初在 1973 年提出，且其中的操作与 DES 中的操作相似，也与其它的像 Rijndael 算法等现代加密算法相似。虽然如此，我们引入这样一个稍微有点简单的结构，通过对这样一个加密算法的分析来分析更复杂密码算法的安全性。

（1）S-box 置换

在给出的加密算法中，将 16 比特的数据块分成 4 个 4 比特的子块。每个子块形成一个 4×4 的 S-box（一种置换装置，输入 4 比特输出也为 4 比特）的输入。S-box 可以很容易的用一个由输入的 4 个比特进行编号的、包含 16 个 4 比特数据的查找表格实现（类似 DES 的 S 盒查找表）。S-box 的最基本的性质是其非线性映射，S-box 的输入不能通过对输入的线性变换而得到。

在如图 5.40 加密算法中，对所有的 S-box 可以使用相同的非线性映射，也可是使用不同的映射（DES 每一轮中的 S-box 是不相同，但每一轮重复利用这些 S-box）。是否所有的 S-box 都使用不同的非线性映射还是只使用一个映射，对线性密码分析和差分密码分析来说是等价的。我们在密码分析中使用的映射如表 6.14 所示，是从 DES 的 S-box 中选取的(是第一个 S-box 的第一行)。在该表中，这些十六进制的符号表示图 5.40 的 S-box 的输入与输出。

表 5.14 S 置换

input	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
output	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

（2）P 置换

每一轮的 P 置换只是简单将比特进行替换或者变换比特的位置。图 5.40 中的 P 变换如表 6.15 所示（其中的数字表示比特的位置，1 表示最左边的比特，16 表示最右边的比特）。可以简单的描述 P 置换如下：第 j 个 S-box 的第 i 个输出与第 i 个 S-box 的第 j 个输入相关联。由于在加密的最后一轮使用 P 置换是没有用处的，因此我们的加密算中在最后一轮没有使用 P 置换。

表 5.15 P 置换

input	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
output	1	5	9	13	2	6	10	14	3	7	11	15	4	8	12	16

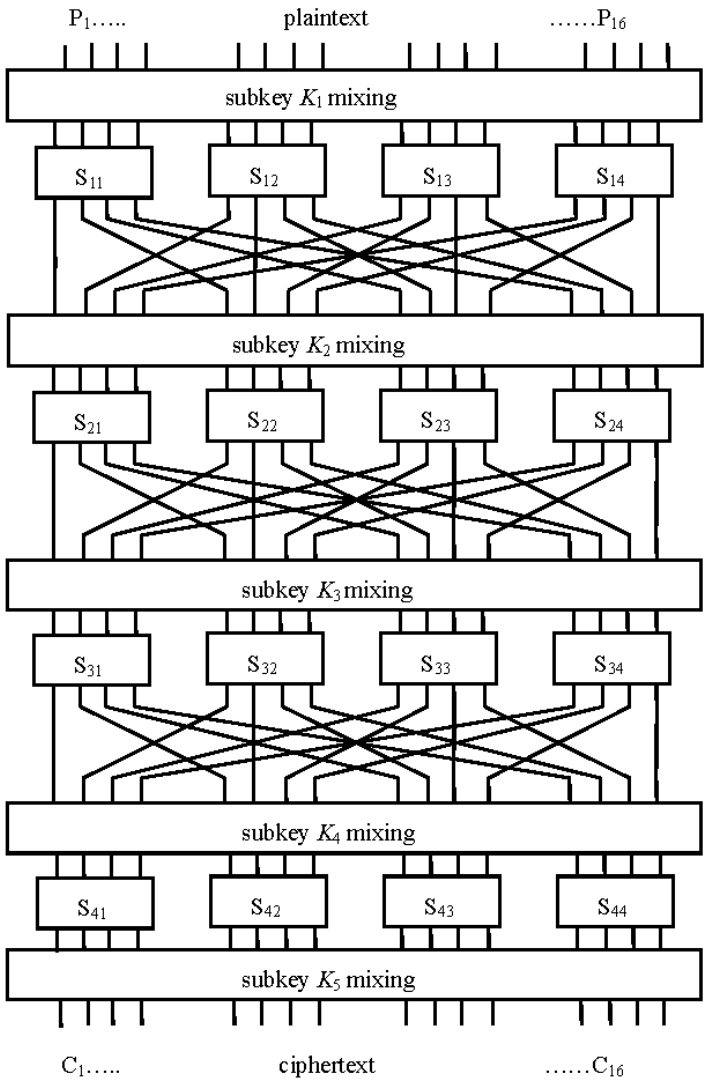


图 5.40 基本的 SPN 加密网络

线性密码分析

（3）密钥混合

密钥混合是将每一轮比特输入和子密钥进行简单的 OR 运算。最后一轮也需要一个子密钥进行密钥混合运算，如果没有子密钥混合运算，则分析者可以对最后一轮的 S-box 做逆运算，从而忽略最后一轮的加密作用。一般地，一个密码算法中，每一轮的子密钥都是通过加密主密钥变换产生的。在我们的加密算法中，假设所有的子密钥比特都是独立产生的而且没有相互联系。

（4）解密

解密时，数据以实际上以与加密相反的方向通过 SPN 加密网络。因此，解密的形式和图 6.40 所示的加密网络的形式相似。但是，解密过程中 S-box 的解密映射应该是加密过程中 S-box 的加密映射的逆映射（例如：输入变成输出，输出变成输入）。也就是说，为了使得一个 SPN 网络既可用于加密又可用于解密，所有的 S-box 必须都是双射的，且对于相同的输入和输出是一个一对一的映射。同时，为了保证加密网络可以正确的解密，子密钥必须以与加密相反的次序使用，并且子密钥的使用必须与 P 置换一致，如果与图 6.40 中的代换－置换网络相似，必须注意到在最后一轮没有 P 置换，以保证加密和解密的 SPN 网络在结构上的一致（如果在最后一轮有一个 P 置换，则在解密的第一轮的 S 置换之前需要一个 P 置换）。

5.7.2 线性密码分析

在这一部分中，我们以前面的 SPN 网络为基础，介绍利用线性密码分析的方法攻击加密算法的方法。

1 基本的分析概述

线性分析的分析者利用了包含明文、密文和子密钥的线性表达式发生的较大可能性（在该例子

中，我们将利用倒数第 2 轮的输出作为密文)。线性分析一种已知明文攻击：假设攻击者已经知道了一系列的明文和相对应的密文。在许多情况下，假设攻击者知道一系列随机的明文和其相应的密文是合理的。

线性分析最基本的思想就是用一个线性表达式来近似表示加密算法的一部分，该线性表达式是关于模 2 的操作（例如用⊕表示 XOR 操作)。表达式具有如下的形式：

$$X_{i_1} \oplus X_{i_2} \oplus \ldots \oplus X_{i_u} \oplus Y_{j_1} \oplus Y_{j_2} \oplus \ldots \oplus Y_{j_v} = 0 \qquad (5.1)$$

其中：X_i表示输入$X = [X_1, X_2, \ldots]$的第i个比特，Y_j表示输出$Y = [Y_1, Y_2, \ldots]$的第j个比特。该表达式是u比特的输入与v比特的输出进行XOR操作的结果。

线性密码分析的方法就是测定上述形式的线性表达式发生的可能性的大小。如果一个密码算法使得等式(5.1)成立的可能性非常大或者非常小，这说明该密码算法的随机性比较差。一般情况下，假如我们随即选择u+v个比特值，并且将它们代入等式（5.1）,等式成立的可能性应该为1/2。在线性密码分析中，一个线性表达式成立的可能性与1/2之间的差值定义为偏移量（或偏差）。一个线性表达式成立的可能性与1/2的距离越大，密码分析者利用线性密码分析就越有效果。在下面的叙述中，我们将一个表达式成立的可能性与1/2的差值称为线性可能性偏移量，简称偏移量或偏差。因此，如果对于随机选择的明文找到相应的密文，使得上述表达式成立的可能性为P_L，则线性可能性偏移量是$P_L - 1/2$。线性可能性偏移量的绝对值$|P_L - 1/2|$越大，线性密码分析对于知道较少明文情况下的攻击越有效。

有多种方法来设计线性分析攻击。我们将利用Matsui的模2运算法则。这样安排线性近似表达式的各个部分：将明文用表达式（5.1）中的X表示，加密算法最后一轮的输入作为表达式（5.1）中的Y（也就是倒数第2轮的输出）。这样明文比特和最后一轮的输入都是随机的。

可以将子密钥的和作为表达式的右边，对表达式进行改写。当等式（5.1）的右边为0时，等式

表示含有这样的子密钥：这些密钥比特是固定的、未知的（因为它们由攻击所希望得到的密钥决定），满足表达式的右侧为零，并且线性表达式成立的可能性为P_L。如果涉及到的子密钥的和为0，表达式（5.1）的偏移量的符号与改写后的包含子密钥的和表达式的偏移量的符号相同；如果涉及到的子密钥的和为1，则偏移量的符号相反。

应该注意：当$P_L=1$表示线性表达式(5.1)具有较好的加密性能同时具有致命的弱点。如果$P_L=0$，则表达式（5.1）表示在加密算法中含有仿射关系，同时也表明该加密算有灾难性的弱点。对于一个模2加法系统，一个仿射函数仅仅就是一个线性函数的补充。对于线性和仿射近似，无论$P_L>1/2$还是$P_L<1/2$，同样都会受线性密码分析的影响，当我们提及线性时，通常既指线性关系也指仿射关系，对指线性关系与仿射关系不加以区分。

那么接下来的问题就是：如何构造一个具有高线性性质的表达式，又如何使用？这仅需要考虑加密算法中的非线性的元素S-box的性质。找出S-box的非线性的特性，就有可能找到S-box的输入与输出之间的近似线性表达式。因此，连接所有S-box的近似线性表达式，就可以省略加密过程的中间数据，这样我们就得到了一个具有较高偏移量且仅仅包含明文和最后一轮输入的加密算法的线性近似表达式。

2 堆积引理

在考虑构造所给加密算法例子的近似线性表达式之前，需要介绍几个基本的理论工具。考虑两个二进制变量X_1，X_2。我们通过计算简单的关系开始：$X_1 \oplus X_2 = 0$是一个线性表达式，等价与$X_1 = X_2$；$X_1 \oplus X_2 = 1$是一个仿射表达式，等价与$X_1 \neq X_2$。

假设，给出如下的概率分布：

$$\Pr(X_1=i)=\begin{cases} p_1, & i=0 \\ 1-p_1, & i=1 \end{cases}$$

$$\Pr(X_2=i)=\begin{cases} p_2, & i=0 \\ 1-p_2, & i=1 \end{cases}$$

如果两个随机变量相互独立，则

$$\Pr(X_1=i,X_2=j)=\begin{cases} p_1p_2 & ,i=0,j=0 \\ p_1(1-p_2) & ,i=0,j=1 \\ (1-p_1)p_2 & ,i=1,j=0 \\ (1-p_1)(1-p_2) & ,i=1,j=1 \end{cases}$$

而且可以等价表示为：

$$\Pr(X_1 \oplus X_2=0)=\Pr(X_1=X_2)$$
$$=\Pr(X_1=0,X_2=0)+\Pr(X_1=1,X_2=1)$$

$=p_1p_2+(1-p_1)(1-p_2)$

另一种表示方法是：令 $p_1=1/2+\varepsilon_1$ $p_2=1/2+\varepsilon_2$，ε_1、ε_2 是线性可能性偏移量，而且 $-1/2\le\varepsilon_1,\varepsilon_2\le 1/2$。因此，

$$\Pr(X_1\oplus X_2=0)=1/2+2\varepsilon_1\varepsilon_2$$

并且$X_1\oplus X_2=0$的线性可能性偏移量$\varepsilon_{1,2}$为$\varepsilon_{1,2}=1/2+2\varepsilon_1\varepsilon_2-1/2=2\varepsilon_1\varepsilon_2$。

这可以扩展到多个随机二进制变量的情况，若有随机变量$X_1\ldots\ldots X_n$且概率为 $p_1=1/2+\varepsilon_1\ldots\ldots$ $p_n=1/2+\varepsilon_n$。假设n个随机相互独立，$X_1\oplus\ldots\oplus X_n=0$ 成立的概率可以由*Piling-Up* 引理（堆积引理）得到。*Piling-Up* 引理假定几个随即变量相互独立。

***Piling-Up* 引理：**

对于n个相互独立的二进制随机变量，$X_1,X_2,\ldots\ldots X_n$，可得

$$\Pr(X_1\oplus\ldots\oplus X_n=0)=1/2+2^{n-1}\prod_{i=1}^{n}\varepsilon_i$$

或者等价表示为$\varepsilon_{1,2\ldots\ldots,n}=2^{n-1}\prod_{i=1}^{n}\varepsilon_i$，其中$\varepsilon_{1,2\ldots\ldots n}$表示$X_1\oplus\ldots\oplus X_n=0$的线性可能性偏移量。

从$\Pr(X_1\oplus\ldots\oplus X_n=0)=1/2+2^{n-1}\prod_{i=1}^{n}\varepsilon_i$可以看出，如果对于所有的i都有$p_i=0$（或者$p_i=1$），

则 $\Pr(X_1 \oplus ... \oplus X_n = 0)$ 可能为0也可能为1。但是，只要有一个p_i=1/2,那么 $\Pr(X_1 \oplus ... \oplus X_n = 0)$ =1/2。

为得到一个加密算法的线性近似表达式，将随机变量X_i作为S-box的线性近似的输入。例如，考虑有四个相互独立的随机变量 X_1，X_2，X_3 和 X_4。令 $\Pr(X_1 \oplus X_2 = 0) = \frac{1}{2} + \varepsilon_1$ 且 $\Pr(X_2 \oplus X_3 = 0) = 1/2 + \varepsilon_{2,3}$，而且 $X_1 \oplus X_3$ 可以由 $X_1 \oplus X_2$，$X_2 \oplus X_3$ 相加得到。因此，可得，

$$\Pr(X_1 \oplus X_3 = 0) = \Pr([X_1 \oplus X_2] \oplus [X_2 \oplus X_3] = 0)$$

所以我们可以通过连接多个线性表达式构成新的线性表达式。因为我们可以认为随机变量 $X_1 \oplus X_2$，$X_2 \oplus X_3$ 也是相互独立的，应用*Piling-Up* 引理可得：

$$\Pr(X_1 \oplus X_3 = 0) = 1/2 + 2\varepsilon_{1,2}\varepsilon_{2,3}$$ 相应的 $\varepsilon_{1,3} = 2\varepsilon_{1,2}\varepsilon_{2,3}$

正如我们将要看到的，表达式 $X_1 \oplus X_2$ =0，$X_2 \oplus X_3$ =0和S-box的线性近似表达式相似，表达式 $X_1 \oplus X_3$ =0和去掉比特X_2的表达式相似。当然实际的线性分析要更复杂，涉及更多的S-box线性近似表达式。

3 分析加密部件

在更详细的讨论攻击细节之前，我们首先需要知道S-box盒的线性脆弱性。如图5.41所示的S-box的输入为X=[$X_1X_2X_3X_4$]，相应的输出为Y=[$Y_1Y_2Y_3Y_4$]。所有的线性表达式都可以通过计算可能性偏移量来检测该表达式是否有用。因此，将X,Y作为S-box的输出，我们检测所有具有表达式（5.1）形式的表达式。

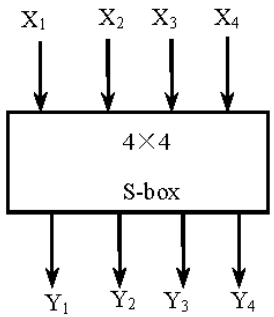


图5.41 S-box映射

例如，在上面的S-box中，考虑表达式$X_2 \oplus X_3 \oplus Y_1 \oplus Y_3 \oplus Y_4 = 0$或等价形式

$X_2 \oplus X_3 = Y_1 \oplus Y_3 \oplus Y_4$。可知，对于16种可能的输入和其相应的输出Y，有12种情况可以使得上面的等式成立，因此线性可能性偏移量是12/16-1/2=1/4。如表5.16所示。相似的，对于等式$X_1 \oplus X_4 = Y_2$其线性可能性偏移量接近于0，而等式$X_3 \oplus X_4 = Y_1 \oplus Y_4$的线性可能性偏移量是2/16-1/2=-3/8。对于最后一种情况，通过负号可知，其最佳的近似表达式是一个仿射近似表达式。

但是，攻击成功的大小取决于线性可能性偏移量的大小，正像我们将要看到的一样，仿射近似表达式与线性近似表达式的效果相同。

表5.16 S-box线性近似采样

$X1$	$X2$	$X3$	$X4$	$Y1$	$Y2$	$Y3$	$Y4$	$X2 \oplus X3$	$Y1 \oplus Y3 \oplus Y4$	$X1 \oplus X4$	$Y2$	$X3 \oplus X4$	$Y1 \oplus Y4$
0	0	0	0	1	1	1	0	0	0	0	1	0	1
0	0	0	1	0	1	0	0	0	0	1	1	1	0
0	0	1	0	1	1	0	1	1	0	0	1	1	0
0	0	1	1	0	0	0	1	1	1	1	0	0	1
0	1	0	0	0	0	1	0	1	1	0	0	0	0
0	1	0	1	1	1	1	1	1	1	1	1	1	0
0	1	1	0	1	0	1	1	0	1	0	0	1	0
0	1	1	1	1	0	0	0	0	1	1	0	0	1
1	0	0	0	0	0	1	1	0	0	1	0	0	1
1	0	0	1	1	0	1	0	0	0	0	0	1	1
1	0	1	0	0	1	1	0	1	1	1	1	1	0
1	0	1	1	1	1	0	0	1	1	0	1	0	1
1	1	0	0	0	1	0	1	1	1	1	1	0	1
1	1	0	1	1	0	0	1	1	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1	0	1	0
1	1	1	1	0	1	1	1	0	0	0	1	0	1

表5.17给出了我们使用的加密算法的S-box的所有线性近似表达式的线性可能性偏移量。其中，Input表示表达式的输入系数，而output表示表达式的输出系数，行和列交集处的值表示以此行列值代表线性表达式成立的数量减去8。因此，一个元素被16整除得到由特定输入和输出组合而成的表达式的线性可能性偏移量。将行和列的16进值看作二进制的组合时，其表示表达式中所涉及的变量。例如一个输入变量的线性近似表达式$a_1.X_1 \oplus a_2.X_2 \oplus a_3.X_3 \oplus a_4X_4$，其中，$a_i \in \{0,1\}$。”.”为二进制的“与”运算，输入行的16进制的值是$a_1a_2a_3a_4$的组合。相似的，对于一个输出变量的线性近似表达式$b_1.Y_1 \oplus b_2.Y_2 \oplus b_3.Y_3 \oplus b_4.Y_4$，其中$b_i \in \{0,1\}$，输出行的16进制的值是$b_1b_2b_3b_4$的组合。因此，线性表达式$X_3 \oplus X_4 = Y_1 \oplus Y_4$（输入行标是3，输出列标是9）的线性可能性偏移量是-6/16=-3/8，且线性表达式成立的概率是1/2-3/8=1/8。

表5.17 线性近似偏移量

		output														
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Input	0	+8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	−2	−2	0	0	−2	+6	+2	+2	0	0	+2	+2	0	0
	2	0	0	−2	−2	0	0	−2	−2	0	0	+2	+2	0	0	−6	+2
	3	0	0	0	0	0	0	0	0	+2	−6	−2	−2	+2	+2	−2	−2
	4	0	+2	0	−2	−2	−4	−2	0	0	−2	0	+2	+2	−4	+2	0
	5	0	−2	−2	0	−2	0	+4	+2	−2	0	−4	+2	0	−2	−2	0
	6	0	+2	−2	+4	+2	0	0	+2	0	−2	+2	+4	−2	0	0	−2
	7	0	−2	0	+2	+2	−4	+2	0	−2	0	+2	0	+4	+2	0	+2
	8	0	0	0	0	0	0	0	0	−2	+2	+2	−2	+2	−2	−2	−6
	9	0	0	−2	−2	0	0	−2	−2	−4	0	−2	+2	0	+4	+2	−2
	A	0	+4	−2	+2	−4	0	+2	−2	+2	+2	0	0	+2	+2	0	0
	B	0	+4	0	−4	+4	0	+4	0	0	0	0	0	0	0	0	0
	C	0	−2	+4	−2	−2	0	+2	0	+2	0	+2	+4	0	+2	0	−2
	D	0	+2	+2	0	−2	+4	0	+2	−4	−2	+2	0	+2	0	0	+2
	E	0	+2	+2	0	−2	−4	0	+2	−2	0	0	−2	−4	+2	−2	0
	F	0	−2	−4	−2	−2	0	+2	0	0	−2	+4	−2	−2	0	+2	0

线性近似表达式有着许多明显的特性。例如，任何非空输出组合等于没有比特输入组合的概率是1/2，因为对于一个双射的S-box来说，任何线性组合的输出比特的中，0和1的数量必须相等。而且，没有比特输出的线性组合等于没有比特输入的线性组合的线性可能性偏移量是1/2，反映在表中的值是+8,位于表格的左上角。因此，最上面的一行除了最左端的值以外都是0。相似的，第一列除

了最上面的值以外也都是0。同样可以得到，任何一行或者一列相加的和等于8或者－8。

4 构造加密函数的线性近似表达式

如果一个SPN网络中的S-box的线性近似信息被找出来，我们可以将数据用形如等式（5.1）关于整个加密算法的近似线性表达式进行运算。可以通过连接S-box的适当线性表达式构造整个算法的线性近似表达式。通过构造一个关于明文和倒数第二轮输入的线性表达式，就有可能恢复出最后一轮加密使用的子密钥的一个子集，以达到攻击的目的。下面通过一个例子来说明：

考虑如图6.42所示的关于S_{12}、S_{22}、S_{32}和S_{34}的线性近似表达式。实际上是建立了一个关于前3轮的表达式，而不是整个4轮加密过程。在下一小节，我们将会了解到如何通过这些表达式得到最后一轮的子密钥。

首先对于图5.42，有如下的S-box线性近似表达式：

$S_{12}: X_1 \oplus X_3 \oplus X_4 = Y_2$　概率为12/16，偏移量为＋1/4

$S_{22}: X_2 = Y_2 \oplus Y_4$　概率为4/16，偏移量为-1/4

$S_{32}: X_2 = Y_2 \oplus Y_4$　概率为4/16，偏移量为-1/4

$S_{34}: X_2 = Y_2 \oplus Y_4$　概率为4/16，偏移量为-1/4

令$U_i(V_i)$作为第i轮S-box的16比特的输入（输出），并且$U_{ij}(V_{ij})$作为第U_i块的第j比特（在图中这些比特按照从左到右的顺序，从1到16顺序标记）。类似地，令K_i表示与第i轮输出进行XOR运算的子密钥。可知，K_5表示与第四轮的输出进行XOR运算的子密钥。

因此，$U_1 = P \oplus K_1$,其中P表示16比特的明文，$\oplus$表示比特之间的OR运算。利用第一轮的线性近似表达式，我们可得

$$V_{1,6} = U_{1,5} \oplus U_{1,7} \oplus U_{1,8}$$

$$= (P_5 \oplus K_{1,5}) \oplus (P_7 \oplus K_{1,7}) \oplus (P_8 \oplus K_{1,8}) \qquad (5.2)$$

其表达式成立的概率为3/4。对于第2轮的线性近似表达式，我们有

$$V_{2,6} \oplus V_{2,8} = U_{2,6}$$

表达式成立的可能性为1/4。因为$U_{2,6} = V_{1,6} \oplus K_{2,6}$，我们可以得到如下形式的线性近似表达式：

$$V_{2,6} \oplus V_{2,8} = V_{1,6} \oplus K_{2,6}$$

表达式成立的可能性为1/4，则联合成立可能性为3/4的表达式(2)可得到如下表达式：

$$V_{2,6} \oplus V_{2,8} \oplus P_5 \oplus P_7 \oplus P_8 \oplus K_{1,5} \oplus K_{1,8} \oplus K_{2,6} = 0 \qquad (5.3)$$

由*Piling-Up* 引理可得其成立的概率为1/2+2(3/4-1/2)(1/4-1/2)=3/8(也就是，其线性可能性偏移量为-1/8)。应当注意我们假设S-box的所有线性近似表达式都是相互独立的，尽管这样做是不完全正确，但在大多数的密码分析中都工作的很好。

对于第3轮，我们得到

$V_{3,6} \oplus V_{3,8} = U_{3,6}$，且等式成立的可能性为 1/4；

$V_{3,14} \oplus V_{3,16} = U$，等式成立的概率为1/4。因此，又由$U_{3,6} = V_{2,6} \oplus K_{3,6}$，$U_{3,14} = V_{2,8} \oplus K_{3,14}$

可得：

$$V_{3,6} \oplus V_{3,8} \oplus V_{3,14} \oplus V_{3,16} \oplus V_{2,6} \oplus K_{3,6} \oplus V_{2,8} \oplus K_{3,14} = 0 \qquad (5.4)$$

且等式成立的概率为12+2(1/4-1/2)^2=5/8(也就是其线性可能性偏移量为+1/8)。同样应用*Piling-Up*引理。联合表达式(5.3)和(5.4)，构成四个S-box的线性近似表达式，

$$V_{3,6} \oplus V_{3,8} \oplus V_{3,14} \oplus V_{3,16} \oplus P_5 \oplus P_7 \oplus P_8 \oplus K_{1,5} \oplus K_{1,7} \oplus K_{1,8} \oplus K_{2,6} \oplus K_{3,6} \oplus K_{3,14} = 0$$

计算$U_{4,6} = V_{3,6} \oplus K_{4,6}$，$U_{4,8} = V_{3,16} \oplus K_{4,8}$，$U_{4,14} = V_{3,8} \oplus K_{4,14}$和$U_{4,16} = V_{3,16} \oplus K_{4,16}$，然后我们可以得到

$$U_{4,6} \oplus U_{4,8} \oplus U_{4,14} \oplus U_{4,16} \oplus P_5 \oplus P_7 \oplus P_8 \oplus \sum k = 0$$

其中，$\sum k = K_{1,5} \oplus K_{1,7} \oplus K_{1,8} \oplus K_{2,6} \oplus K_{3,6} \oplus K_{3,14} \oplus K_{4,6} \oplus K_{4,8} \oplus K_{4,14} \oplus K_{4,16}$

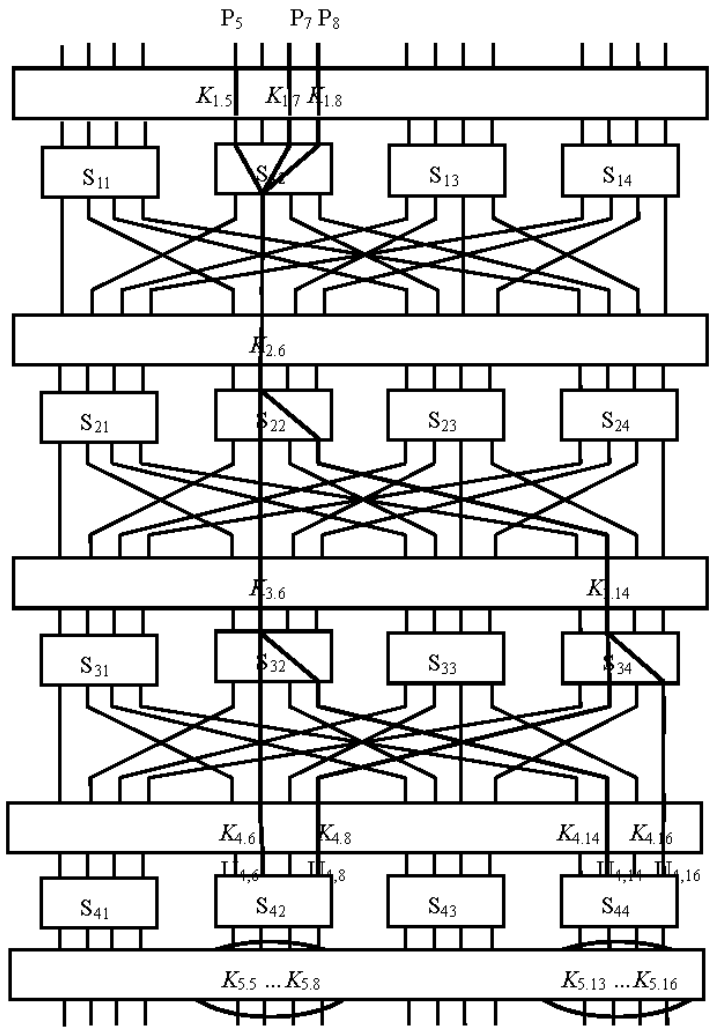

P5 P7 P8
K1.5 K1.7 K1.8
S11 S12 S13 S14
K2.6
S21 S22 S23 S24
K3.6 K3.14
S31 S32 S33 S34
K4.6 K4.8 K4.14 K4.16
U4,6 U4,8 U4,14 U4,16
S41 S42 S43 S44
K5.5 ...K5.8 K5.13 ...K5.16

图5.42 线性分析

并且$\sum k$是确定的（或者为0或者为1，取决于加密算法的密钥）。利用*Piling-Up* 引理，可以得出上述表达式成立的概率为1/2+2^3(3/4-1/2) (1/4-1/2)^3=15/32(也就是说，其线性可能性偏移量为-1/32)。

因为$\sum k$的值是确定的，我们可以得到

$$U_{4,6} \oplus U_{4,8} \oplus U_{4,14} \oplus U_{4,16} \oplus P_5 \oplus P_7 \oplus P_8 = 0 \qquad (5.5)$$

成立的概率为15/32 或者(1-15/32)=17/32（取决于$\sum k$的值是0还是1）。换句话说，我们现在有前三轮加密过程的一个线性近似表达式，其线性偏移量是1/32。下面将讨论如何利用这样一个偏移量来确定加密子密钥的某些密钥比特。

5 获取密钥比特

一旦获得了有R轮加密过程的加密算法的前R-1轮的线性近似表达式，且该表达式具有足够大的线性可能性偏移量，则恢复最后一轮的子密钥来攻击该密码算法是可行的。在我们所给的例子中，通过前3轮循环的线性近似表达式获得子密钥K_5是有可能的。我们把从最后一个子密钥中恢复出的密钥的一部分称为局部目标子密钥（或部分目标子密钥）。局部目标子密钥来自与最后一轮的S-box相关联的子密钥。

接下来的步骤包括部分的破解最后一轮的加密。特别地，对于所有可能的局部目标子密钥，相应的密文比特与其进行OR运算，运算的结果向后通过最后一轮相应的S-box进行运算。对所有的明文/密文对进行这种运算过程，并且设置一个计数器对每个局部目标子密钥进行计数。若对于输入到最后一轮S-box的（密文）比特（由对最后一轮的破解得到）和已知明文，线性近似表达式成立，则

计数器增加。如果某个局部目标子密钥对于明文/密文对的计数值与明文/密文对数目的一半相差最大，则该子密钥被假定为正确的子密钥。这样做的原因是一个正确的子密钥将使得线性近似表达式成立的概率远离1/2（至于大于1/2还是小于1/2，则取决于式线性表达式还是仿射表达式作为近似表达式最合适，且取决于线性表达式中涉及的未知的子密钥比特）。一个不正确的子密钥被认为导致一个向最后一轮S-box得随机猜测输入，因而线性表达式成立的可能性非常接近1/2。

将该理论应用于我们的例子。线性表达式(5.5)影响S-box 中S_{42}和S_{44}的输入。对于每个明文/密文对，我们尝试部分目标子密钥$[K_{5,5}...K_{5,8},K_{5,13}...K_{5,16}]$的256种可能。其中$[U_{4,5}...U_{4,8},U_{4,13}...U_{4,16}]$是通过将相应的密文与子密钥$[K_{5,5}...K_{5,8},K_{5,13}...K_{5,16}]$进行OR运算，然后将运算结果向后经过$S_{42},S_{44}$运算得到的。对每个局部目标子密钥，当表达式(6.5)成立时，我们增加计数。若一个子密钥的计数值与明文/密文的数目的一半相差最多，被假定为正确的子密钥。至于差值符号是正还是负，取决与所设计到的子密钥的和$\sum K$。当$\sum K=0$时，表达式(6.5)作为一种估计，且成立的概率小于1/2；当$\sum K=1$时，表达式(5.5)成立的概率大于1/2。

我们产生10000个已知明文/密文对，来模拟前面描述的攻击，取部分子密钥为$[K_{5,5}...K_{5,8}]=[0010]$，$[K_{5,13}...K_{5,16}]=[0100]$来模拟前面描述的攻击。正如我们所期望的，计数远离5000的符合条件的子密钥正是[2,4]（16进制表示）。表5.18给出了部分子密钥对应的偏移量计数值（完整的应该有256条记录，每个目标子密钥对应一个），从中可以确认我们找到了正确的子密钥。结合表种的数据，可以计算出线性可能性偏移量，公式如下：

$$|bias|=|count-5000|/10000$$

其中count表示对应的子密钥的计数值。

从表5.18中，可以看出偏移量最大的子密钥是$[K_{5,5}...K_{5,8},K_{5,13}...K_{5,16}]=[2,4]$，实际上，这个结果对于所有可能子密钥产生的结果也是正确的。

表5.18 线性攻击的实验数据

partial subkey [*K*5,5...*K*5,8, *K*5,13...*K*5,16]	| bias |	*partial subkey* [*K*5,5...*K*5,8, *K*5,13...*K*5,16]	| bias |
1C	0.0031	2A	0.0044
1D	0.0078	2B	0.0186
1E	0.0071	2C	0.0094
1F	0.0170	2D	0.0053
20	0.0025	2E	0.0062
21	0.0220	2F	0.0133
22	0.0211	30	0.0027
23	0.0064	31	0.0050
24	**0.0336**	32	0.0075
25	0.0106	33	0.0162
26	0.0096	34	0.0218
27	0.0074	35	0.0052
28	0.0224	36	0.0056
29	0.0054	37	0.0048

试验测定的偏移量是0.0336非常接近预期得线性可能性偏移量1/32=0.0315。注意到，尽管正确的局部子密钥具有明显的最大的偏移量，其它的较大的偏移量的出现说明对不正确的目标子密钥的

检测不等价于把一个随机的数据看作一个线性表达式（可能偏移量非常接近0）的输入。S-box的特性可能会影响不同局部目标子密钥的解密，利用*Piling-Up*定理时不严密的随机变量相互独立的假设，以及线性累积（在下一小节将会提到）等原因都会导致实验时得到的偏移量不准确性。

6 攻击的复杂度

我们把线性近似表达式中包含的S-box称为活跃S-box。在图6.42中，从第1轮到第3轮中被粗线条标出的四个S-box都是活跃的。一个线性近似表达式成立的可能性，与S-box的线性可能性偏移量的大小，活跃S-box的数目有关。一般情况下，这些活跃S-box的线性可能性偏移量越大，整个线性近似表达式的线性可能性偏移量越大。同样，活跃S-box的线性可能性偏移量越小，整个线性表达式的线性可能性偏移量越小。

令 ε 表示整个线性近似表达式成立的概率与1/2之间的偏移量。Matsui给出了攻击所需要的明文的数量与 ε^{-2} 成正比。令 N_L 表示需要的已知明文的数量，用公式 $N_L \approx 1/\varepsilon^2$ 来近似估计 N_L 是合理的。

在实际中，期望得到 ε^{-2} 几倍的明文是合理的假设。尽管严格意义上说，密码分析的复杂性需要从时间和空间（或者存储器）进行衡量，我们用攻击时所需要的明文数量来考虑密码分析的复杂性。那就是说，假设如果我们能获得密码分析所需要的 N_L 明文，那么我们有足够的时间和空间处理它，即假定分析者的计算能力和存储能力是无限的。

既然线性偏移量是由对S-box的线性近似表达式应用*Piling-Up*引理得到的，那么可以很容易推出，偏移量取决于S-box的线性可能性偏移量和活跃S-box的数目。一般地提高算法安全性抵抗线性分析的方法集中在优化S-box（例如，减小最大偏移量）和增加活跃S-box的数目。6.3节介绍的Rijndael

算法就是应用这种方法的一个极好的例子。

然而，需要知道的是，证明一个算法对线性分析是安全的，其前提是不存在高可能性的线性近似。但是，对这样的线性近似的可能性的计算，是以下两个假设为前提的：每个S-box的线性近似是相互独立的（这样才可以应用*Piling-Up*引理）；一个线性近似足够确定关于明文和最后输入数据之间的一个最佳线性近似表达式。实际情况是S-box的线性近似之间往往都不是相互独立的，这是计算线性可能性的一个重大障碍。还有，对于包含相同的明文和最后一轮输入的线性近似，具有不同种活跃S-box的组合可能比只有一种活跃S-box的线性近似具有更高的线性可能性，这个概念被称为线性累积。一个最明显的例子，许多具有较小偏移量的线性近似部件，从自身的角度看可能都是对线性分析攻击是免疫的。但当把这些部件组合在一起时，构成的线性近似表达式可能具有较大的线性可能性偏移量。但是，我们在此提出的方法，在许多密码分析中都是正确的，因为独立性假设是一个合理的近似，并且当一个近似线性部分具有较高的偏移量时，则由它支配线性累积。

差分分析

5.7.3 差分密码分析

在这一部分，我们将讨论差分分析在对SPN加密网络攻击中的应用。差分分析和线性分析在许多方面都是相似，它与线性密码分析的主要区别是差分分析包含了将两个输入的异或与两个输出的异或相比较。

1 攻击概述

差分密码分析利用了明文差分与最后一轮输入差分的某种特定情况发生的高可能性。若一个加密算法的输入表示为$X=[X_1X_2...X_n]$，输出表示为$Y=[Y_1Y_2...Y_n]$。令算法的两个输入为X'与X''，相应的输出为Y'和Y''。则输入差分为$\Delta X=X'\oplus X''$，其中$\oplus$表示比特组之间的XOR运算，因此$\Delta X=[\Delta X_1\Delta X_2...\Delta X_n]$，其中$\Delta X_i=X_i'\oplus X_i''$，$X_i'$与$X_i''$分别表示$X'$与$X''$的第i个比特。输出差分公式为$\Delta Y=Y'\oplus Y''$，且$\Delta Y=[\Delta Y_1\Delta Y_2...\Delta Y_n]$，其中$\Delta Y_i=Y_i'\oplus Y_i''$。

在理想的随机化加密算法中，给定一个输入差分ΔX使其对应的输出差分为特定的ΔY的概率是$1/2^n$，n是输入X的比特数目。差分分析利用这样一种情况：给定一个输入差分ΔX，特定输出差分为ΔY，该情况发生的概率很高（例如远大于$1/2^n$，概率记为P_D）。$(\Delta X,\Delta Y)$被作为一个差分对。

差分分析是一种选择明文攻击方法，意味着攻击者可以选择明文输入，检测相应输出以得到密

钥。对于差分分析，攻击者选择输入的数据对为X'与X''，得到一个特定的ΔX，对于这个已知的ΔX，一个特定的ΔY发生的概率很高。

在此，我们把差分结构$(\Delta X, \Delta Y)$中的明文用X表示，最后一轮的输入用Y表示。我们可以通过检测高概率的差分特征来实现。在加密算法中，前一轮的输出差分对应着下一轮相应的输入差分，一系列每一轮的输入和输出差分就构成了差分特征。利用高概率的差分特征，我们有可能利用输入到最后一轮的信息来获取最后一轮子密钥的子集。

和线性密码分析相同，为了构造高可能性的差分特征，我们观察每个单独的S-box的性质，并且用这些特征来得到完整的差分特征。特别地，我们考察S-box的输入、输出的差分，以得到一个高可能性的差分对。按照加密的轮次结合各个S-box的差分对，且非零的输出差分对应下一轮相应的非零的输入差分。这样，使得我们可以找到一个由明文差分和最后一轮输入的差分组成高可能性差分对。在差分分析中，加密所用的子密钥消失了，因为它们与所有的数据集合有关，当考虑它们对差分表达式的影响时，由其与自身进行XOR运算，可知结果为零。

2 分析加密部件

我们测试S-box的所有差分对。如图5.41所示的4×4 的S-box，其输入为$X=[X_1X_2X_3X_4]$，输出为$Y=[Y_1Y_2Y_3Y_4]$。一个 S-box 的所有差分对$(\Delta X, \Delta Y)$可以被找出，而且给定ΔX对应特定ΔY的可能性也可以得到（对于输入对(X', X'')，其输入差分 $\Delta X = X' \oplus X''$）。因为输入对的顺序是不相关的，则对一个 4×4 的 S-box，我们只需要考虑X'的值，已知ΔX值的情况下利用公式$X'' = X' \oplus \Delta X$可以计算出对应的X''的值。

表 5.19 S-box 的差分对采样

X	Y	ΔY		
		$\Delta X = 1011$	$\Delta X = 1011$	$\Delta X = 0100$
0000	1110	0010	1101	1100
0001	0100	0010	1110	1011
0010	1101	0111	0101	0110
0011	0001	0010	1011	1001
0100	0010	0101	0111	1100
0101	1111	1111	0110	1011
0110	1011	0010	1011	0110
0111	1000	1101	1111	1001
1000	0011	0010	1101	0110
1001	1010	0111	1110	0011
1010	0110	0010	0101	0110
1011	1100	0010	1011	1011
1100	0101	1101	0111	0110
1101	1001	0010	0110	0011
1110	0000	1111	1011	0110
1111	0111	0101	1111	1011

考察加密算法中的 S-box，对于每个输入对 $(X', X'' = X' \oplus \Delta X)$ 我们可以得到其对应的输出差分 ΔY。例如，对于二进制值 X 和 Y，给定输入对 $(X, X \oplus \Delta X)$ 以及其相应 ΔY。如表 5.19 所示，表格的最后三列给出了在 ΔX 的值分别为 1011(16 进制 B),1000(16 进制 8)和 0100（16 进制 4）的情

况下，对应的 ΔY 的值。从这个表格中，我们可以知道对于 $\Delta X = 1011$，$\Delta Y = 0010$ 情况的概率是 8/16，$\Delta X = 1000, \Delta Y = 1011$ 的情况的概率是 4/16，$\Delta X = 0100$，$\Delta Y = 1010$ 情况发生的次数是 0。如果 S-box 是理想化的情况，每一个差分对发生的次数是 1，也就是给定 ΔX 对应特定 ΔY 的概率是 1/16（已经被证明这样的 S-box 在数学理论上是不存在的）。

表 5.20 差分分布表

Out / In	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	2	0	0	0	2	0	2	4	0	4	2	0	0
2	0	0	0	2	0	6	2	2	0	2	0	0	0	0	2	0
3	0	0	2	0	2	0	0	0	0	4	2	0	2	0	0	4
4	0	0	0	2	0	0	6	0	0	2	0	4	2	0	0	0
5	0	4	0	0	0	2	2	0	0	0	4	0	2	0	0	2
6	0	0	0	4	0	4	0	0	0	0	0	0	2	2	2	2
7	0	0	2	2	2	0	2	0	0	2	2	0	0	0	0	4
8	0	0	0	0	0	0	2	2	0	0	0	4	0	4	2	2
9	0	2	0	0	2	0	0	4	2	0	2	2	2	0	0	0
A	0	2	2	0	0	0	0	0	6	0	0	2	0	0	4	0
B	0	0	8	0	0	2	0	2	0	0	0	0	0	2	0	2
C	0	2	0	0	2	2	2	0	0	0	0	2	0	6	0	0
D	0	4	0	0	0	0	0	4	2	0	2	0	2	0	2	0
E	0	0	2	4	2	0	0	0	6	0	0	0	0	0	2	0
F	0	2	0	0	6	0	0	0	0	4	0	2	0	0	2	0

我们可列出一个 S-box 的输入输出差分分布表格，其中 ΔX 作为行，ΔY 作为列。表格 5.19 所

表示的 S-box 的差分分布表如表 5.20 所示。表格中的数字表示对于给定的输入差分 ΔX，相应的输出差分为 ΔY 时，发生的次数。从表 5.20 中可以得知：除了特殊情况 $(\Delta X = 0, \Delta Y = 0)$ 以外表格中的最大值是 8，对应的是 $\Delta X = B, \Delta Y = 2$。因此，对于 $\Delta Y = 2$ 时，给定一个随机的输入对满足 $\Delta X = B$ 的概率是 8/16。表格中值是最小值为 0，且许多差分对的对应值为 0。在这种情况下，给定 ΔX ,对应的输出差分为 ΔY 的概率是 0。

差分分布表有许多基本的特征。首先，每一行的元素相加的和是 $2^n = 16$；类似的，每一列的元素相加的和也是 $2^n = 16$。其次，所有的元素都是偶数：这是因为对于输入 (X', X'') 和 (X'', X') 的差分值（$\Delta X = X' \oplus X'' = X'' \oplus X'$）相同。还有，对于一个一对一映射的 S-box，如果输入差分为 $\Delta X = 0$，则必须导致输出差分 $\Delta Y = 0$。因此，表格最右上角的值为 $2^n = 16$，而且第 1 行和第 1 列的其它值均为 0。最后，如果我们能构造一个理想的 S-box，该 S-box 的输入与输出之间没有明显的差分信息，差分分布表的每一项元素都是 1，且给定 ΔX 对应的特定输出差分为 ΔY 的概率应该是 $1/2^n = 1/16$。但是，将上面讨论的性质都要达到，这显然是不可能的。

在讨论联合 S-box 的差分对构成一个差分特性，而且选择一个攻击中应用的合适差分对之前。我们必须讨论密钥对 S-box 的差分性质的影响。一般假设不涉及密钥的 S-box 的输入为 X，且输出为 Y。但是，在实际的加密算法中，我们必须考虑应用到每个 S-box 的密钥。如图 5.43 所示，在这种情况下，我们令涉及密钥的 S-box 的输入为 $W = [W_1 W_2 W_3 W_4]$，则可如下表示 S-box 的输入差分表达式

$$\Delta W = [W_1' \oplus W_1'' W_2' \oplus W_2'' .. W_n' \oplus W_n'']$$

其中，$W' = [W'_1 W_2' .. W_n']$，$W'' = [W''_1 W_2'' .. W_n'']$ 表示 S-box 的两个输入。

由于 W', W'' 的加密密钥是相同的，则

$$\begin{aligned}\Delta W_i &= W_i' \oplus W_i' \\ &= (X_i' \oplus K_i) \oplus (X_i'' \oplus K_i) \\ &= X_i' \oplus X_i'' \\ &= \Delta X_i\end{aligned}$$

因为，$K_i \oplus K_i = 0$。可知，密钥比特对于 S-box 的输入差分没有影响，可以忽略密钥的影响。换句话说，涉及密钥的 S-box 与不涉及密钥的 S-box 的差分分布相同。

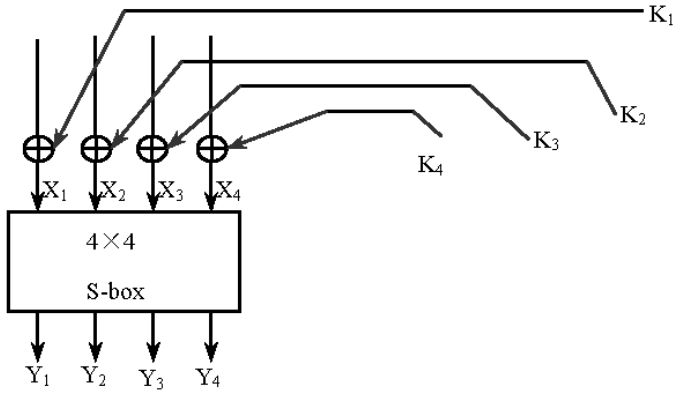


图 5.43 带有密钥的 S-box

3 构造差分特征

如果找到了一个SPN加密网络的S-box差分信息，可以利用整个加密算法的差分特征进行数据处理。构造每一轮的特定S-box的输入、输出差分构成差分特征，然后连接这些差分特征，构成整个加密算法的差分特征。这样整个加密算法的差分特征涉及了明文比特和加密算法的最后一轮的输入。通过恢复最后一轮子密钥的子集进行密码分析是有可能的。我们用一个例子来说明如何构造一个差分特性。

考察一个包含的S_{12}、S_{23}、S_{32}和S_{33}的差分特性。和线性分析的情况相同，将差分特征形象化成如图5.44的形式是非常有用的。该图表示了非零的差分通过加密网络时的影响，其中以加粗线条模式标出的S-box是活跃S-box。应该注意，这样是构造了一个前三轮加密的差分特征，而不是整个加密过程的四轮。在下面我们将会看到，这样的差分特性在从最后一个子密钥中获取部分密钥比特的过程中的作用。

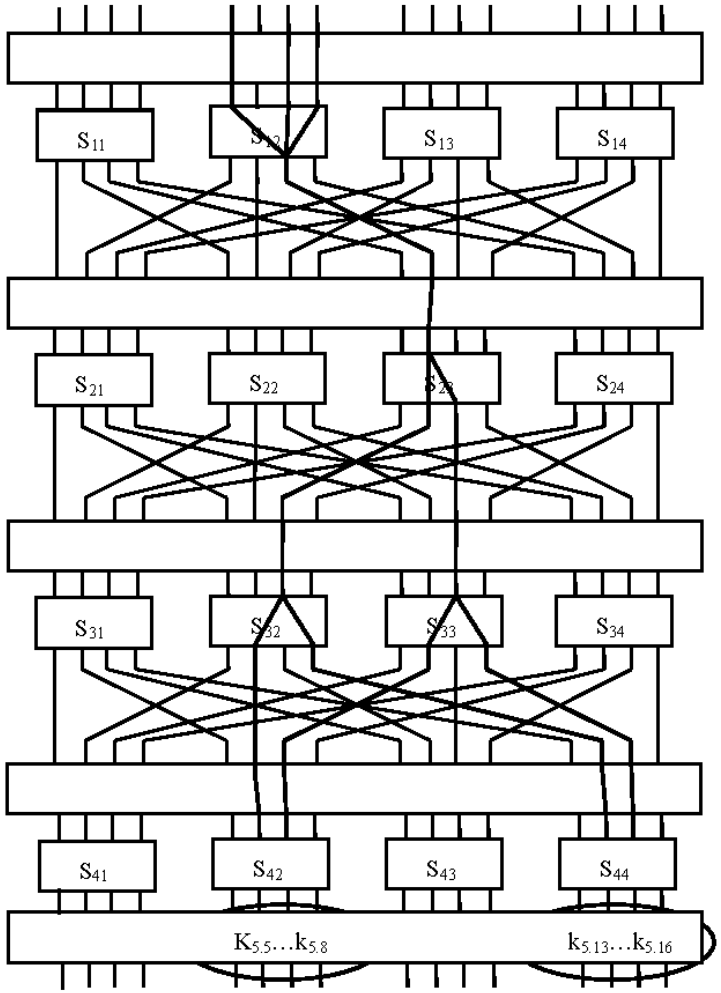


图5.44 差分分析

我们利用下面的S-box差分对：

$S_{12}: \Delta X = B \rightarrow \Delta Y = 2$ 成立的概率为8/16

$S_{23}:\Delta X=4\rightarrow\Delta Y=6$ 成立的概率为6/16

$S_{32}:\Delta X=2\rightarrow\Delta Y=2$ 成立的概率为6/16

$S_{33}:\Delta X=2\rightarrow\Delta Y=5$ 成立的概率为6/16

所有其它的S-box会有零输入差分和零输出差分。

加密算法的输入差分等于第一轮的输入差分，以下式给出：

$$\Delta P=\Delta U_1=[0000101100000000]$$

同样，和我们讨论线性分析一样，用U_i表示第i轮的S-box的输入，V_i表示第i轮的S-box的输出。因此，ΔU_i 与 ΔV_i 分别表示相应的差分。可知

$\Delta V_1=[0000001000000000]$，上面所列$S_{12}$的差分对和接下来的第1轮P置换使得

$\Delta U_2=[0000000001000000]$，可知给定明文差分 ΔP 的情况下 ΔU_2 的概率是8/16=1/2。

第2轮的差分特征：利用S_{23}的差分对

$$\Delta V_2=[0000000001100000]$$

和第2轮的P置换给出

$$\Delta U_3=[0000001000100000] \tag{5.6}$$

给定 ΔU_2 的条件下 ΔU_3 发生的概率为6/16，给定 ΔP ，ΔU_3 发生的概率为8/16×6/16=3/16。在给定

明文差分 ΔP 确定概率时，我们假设第1轮的差分特征与第2轮的差分特征是相互独立的。因此，两者同时发生的概率是两者概率的乘积。

随后，我们利用第三轮的S_{32}、S_{33}的差分以及第三轮P置换得

$\Delta V_3 = [0000010101010000]$ 以及 $\Delta U_4 = [0000011000000110]$

在给定 ΔU_3 的情况下 ΔU_4 发生的概率为 $(6/16)^2$，因此，给定明文差分 ΔP 的情况下 ΔU_4 的概率是

$8/16 \times 6/16 \times (6/16)^2 = 27/1024$，其中我们假设所有轮的S-box的差分对相互独立。

在密码分析的过程中，许多差分为 $\Delta P = [0000101100000000]$ 的明文对被加密。图5.44中所示的差分特征将以27/1024的高概率发生。我们称这样的 ΔP 对为正确对。差分特征中不发生的明文差分对为错误对。

4 获取密钥比特

如果得到了一个具有R轮加密过程的加密算法的前R-1轮差分特性，且该差分特性具有较高的概率，通过恢复最后一轮的子密钥的子集来攻击密码算法是可能的。在我们使用的加密算法的例子中，从获取子密钥K_5的比特是可能的。接下来的步骤包括穷举攻击最后一轮的加密和检测对最后的一轮输入是否有一个正确对。我们把与最后一轮加密相关联的，且对应的S-box受差分特征影响的子密钥称为局部目标子密钥（或部分目标子密钥）。对最后一轮的解密包括：将密文与局部目标子密钥进行XOR运算，将运算结果返回最后一轮的S-box进行运算，且所有可能的部分目标子密钥都要进行这种运算。

对所有可能的局部目标子密钥，依照其明文对的差分是否为 ΔP ,处理每一个密文对。每个局部

目标子密钥设置一个计数器，当最后一轮的输入的差分符合差分特性的值时，计数器的计数增加。具有最大计数的局部目标子密钥被作为子密钥的真实值。这样做的原因是，正确的局部目标子密钥会使得最后一轮的差分的概率像差分特性预计的一样，因为差分特性具有较高的概率。（如果一个错误对产生，且此时的局部目标子密钥是正确的子密钥，正确局部目标子密钥的计数不会增加。）一个不正确的子密钥被看作导致对输入到S-box的最后一轮的数据的随机猜测，因此得到的差分与预计的差分特性一致的可能性非常小。

表 5.21 差分分析的实验结果

partial subkey [*K*5,5...*K*5,8, *K*5,13...*K*5,16]	prob	*partial subkey* [*K*5,5...*K*5,8, *K*5,13...*K*5,16]	prob
1C	0.0000	2A	0.0032
1D	0.0000	2B	0.0022
1E	0.0000	2C	0.0000
1F	0.0000	2D	0.0000
20	0.0000	2E	0.0000
21	0.0136	2F	0.0000
22	0.0068	30	0.0004
23	0.0068	31	0.0000
24	**0.0244**	32	0.0004
25	0.0000	33	0.0004
26	0.0068	34	0.0000
27	0.0068	35	0.0004
28	0.0030	36	0.0000
29	0.0024	37	0.0008

在对我们所给例子进行攻击时，差分分析影响最后一轮的 S-boxS_{42} 和 S_{44}。对于每个密文对，我们将尝试密钥$[K_{5,5}...K_{5,8}, K_{5,13}...K_{5,16}]$的 256 种所有可能。对于每个局部目标子密钥值，如果由对最后一轮解密得到的输入差分符合 ΔU_4，我们增加局部目标子密钥的计数值。其中这样计算$[\Delta U_{4,5}...\Delta U_{4,8}, \Delta U_{4,13}...\Delta U_{4,16}]$的值，将密文与子密钥进行 XOR 运算，然后将运算的结果逆向经过 S_{24} 和 S_{44} 运算。对于每个局部目标子密钥来说，计数表示与正确对一致的差分发生的次数（假设局部目标子密钥是正确的）。计数值最大的作为正确的子密钥是因为我们假设正确对发生的概率最大。

并不需要处理所有的密文对。因为最后一轮的输入差分仅仅对应两个 S-box，当满足差分特性时，S_{41}、S_{43} 对应的密文差分必定为 0。因此我们可以过滤许多错误的对，若一个密文差分的 S_{41}、S_{43} 对应子块不出现 0，则不处理该密文对。在这些情况下，由于密文对不对应正确的对，所没有必要去检测$[\Delta U_{4,5}...\Delta U_{4,8}, \Delta U_{4,13}...\Delta U_{4,16}]$。

我们可以模拟攻击带有密钥的基本密码算法，例如利用随机产生的 5000 个明文/密文对（例如，含有符合明文差分为 $\Delta P = [0000101100000000]$的 10000 个密文），下面描述详细的过程。正确的部分目标子密钥应该是$[K_{5,5}...K_{5,8}, K_{5,13}...K_{5,16}] = [0010, 0100] = [2,4]_{hex}$。正如预计的一样，具有最大计数的子密钥是$[2,4]_{hex}$，证明攻击成功的获取了局部目标子密钥。表 5.21 中，给出了一部分局部目标子密钥的计数（完整的数据需要 256 项，每个部分子密钥一项），表中的数据表明了对于局部目标子密钥，利用如下公式对正确对发生的可能性的估计，

$$prob = count / 5000$$

其中，count 是局部目标子密钥的计数。

如同从表 5.21 中所看到的结果一样，子密钥可能性最大的是 $[K_{5,5}...K_{5,8}, K_{5,13}...K_{5,16}] = [2,4]_{hex}$，实际上在整个部分子密钥中，这个结果也是正确的。

在我们的例子中，我们希望那个正确的对发生的可能性是 $P_D = 27/1024 = 0.0264$，而且我们通过实验得到的正确密钥 $[2,4]$ 的概率是 $P_D = 0.0244$。注意，有时候会有由错误的局部目标子密钥产生的其它较大的计数。这表明，对于不正确目标子密钥的检测不等价于将一个随机差分与预计的差分进行对比。有许多因素使得理论的计数与我们实际所得到的计数不同，S-box 的性质对于不同的局部目标子密钥在对最后一轮解密的影响，构造差分特性时的对独立性的不严密的假设，差分包括的多种差分性质等（将在下一小节介绍）。

5 攻击的复杂度

对于差分密码分析，我们把包含在差分的特性中的、且有非零输入差分的（因此输出差分也不为零）S-box 称为活跃 S-box。一般的，活跃 S-box 的差分的可能性越大，则整个加密算法的差分特性的可能性越大。而且，活跃 S-box 的较少，则整个加密算法的差分特性的可能性越大。与线性分析一样，我们用攻击时需要的明文的数量来衡量攻击的复杂性。那就是说，如果我们能获得 N_D 数量的明文，则我们有足够的能力处理它。

一般的，确定准确的攻击需要的明文对的数量是非常复杂的。但是，从以往的经验可知，在尝

试局部目标子密钥时，可以辨别出正确对，所需要的明文对的数量N_D为

$$N_D \approx c / P_D \tag{5.7}$$

其中，P_D是一个 R-1 轮的差分特性的概率，c 是一个小的常量。假设每个 S-box 的差分发生的概率是相互独立的，则差分特性的概率可以如下公式给出：

$$P_D = \prod_{i=1}^{r} \beta_i \tag{5.8}$$

其中，r 表示活跃 S-box 的数量，β_i表示第 i 个活跃 S-box 对特定的差分对发生的概率。

证明(5.7)合理是并不困难。它简单的表明，少量正确对的产生就可以给一个正确的局部目标子密钥足够大的计数，且显然的多于对不正确的局部目标子密钥的计数。既然当大约$1/P_D$明文对被检测，就会得到一个正确对。因此，假设用几倍于$1/P_D$数量的明文，来进行密码分析是合理的。

抵抗差分分析的方法一般是优化 S-box 的特性（例如，减小每个 S-box 的差分的概率）以及寻找一种扩大活跃 S-box 的数目的结构。Rijndael 就是一种高抗差分分析的加密算法。

与线性分析一样，必须谨慎证明加密算法对差分分析的免疫性。差分特性的概率的计算是以 S-box 的近似相互独立为前提条件的，在一个真正的加密算法中，输入到 S-box 的数据是相互依赖的。因此，P_D是一个估计值。在实际应用中，这样的假设被证明是正确的。

有着相同的输入差分和输出差分的的差分特性结合在一起，所构成的差分特性的概率要远大于

只考虑一个单独的差分特性时的概率（类似线性累积的概念）。要证明对差分分析是安全的，需要证明所有独立的差分的概率都小于某个极限值，而不是仅仅证明所有的差分特性的值都小于某个极限值。然而，下面的假设是合理的，一般当一个差分特性具有较高的概率时，其决定着差分发生的概率，而且其差分特征的概率是差分发生的概率的近似。

Thank you!

(To be continued....)