东南大学网络空间安全学院

密码学与安全协议

# 第三讲 对称密码算法

黄　杰

信息安全研究中心

# 本讲内容

- 对称加密算法模型
- 经典Feistel结构
- DES加密算法
  - 工作原理
  - 加密算法的模式
  - 多重DES
- 针对DES的差分攻击
- AES加密算法

# 知识点

1、对称密码算法的安全性

2、对称密码的体系架构

3、DES密码算法

4、DES的差分攻击

5、AES算法

# 需要明确的问题

- 密码系统的三个特征
  - 密钥的数量？
  - 处理明文的方式？
  - 明文转换成密文的模型？
- 什么样的加密算法是一个好的加密算法？
- 分组密码的模型是什么？

- 问题：加密方法（算法）的安全性依赖于什么？
- 问题：选择密码系统的原则是什么？

# 现代密码学研究的历程

- 1914年之前密码学进展很少出现在公开文献上。
- 1918年，《重合指数及其在密码学中的应用》在Riverbank实验室诞生。20世纪最有影响的密码分析文章。
- 在20世纪三、四十年代，有多篇基础性文章出现在公开文献上，但内容离当时的水平相差非常远。
- 二战结束时，公开的文献几乎没有，但有个例外，香农在1949年在《贝尔系统技术杂志》发表了文章《保密系统的通信理论》，也是战时产物。
- 1949-1967，密码文献出现空白。
- 1967，David出版了《破译者》一书。

# 现代分组加密技术

- Lucifer算法（60年代末），DES（1973），Madryga算法（1984），NewDES算法（1985），FEAL算法（80‘），REDOC算法（1991），Khufu和Khafre算法（1990），LOKI（1990），IDEA（1990），Skipjack（1990），Gost，Blowfish，SAFER，3—WAY，Crab，SXAL8/MBAL，RC5。
- DES最重要的三个原因：
  - 通过验证，安全性高；
  - DES是最早成为国际标准的对称加密算法；
  - DES目前仍然被广泛的使用。

# 经典Feistel网络

- **Feistel网络设计的动机**
  - 理想分组密码：**$\{0,1\}^n \to \{0,1\}^n$**的代换
    - 完全随机的交换
    - 一个密钥即是一个可逆映射
    - 随机选择一个密钥

- **Feistel网络解决的问题**
  - 当n较小时，等价于替换变换

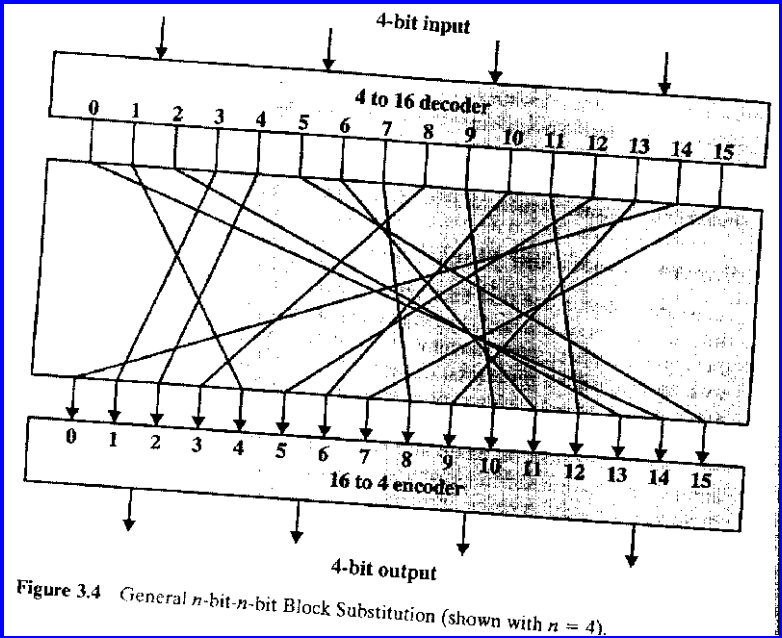



Figure 3.4 General *n*-bit-*n*-bit Block Substitution (shown with $n = 4$).

**Table 3.1** Encryption and Decryption Tables for Substition Cipher of Figure 3.4

| Plaintext | Ciphertext |
|---|---|
| 0000 | 1110 |
| 0001 | 0100 |
| 0010 | 1101 |
| 0011 | 0001 |
| 0100 | 0010 |
| 0101 | 1111 |
| 0110 | 1011 |
| 0111 | 1000 |
| 1000 | 0011 |
| 1001 | 1010 |
| 1010 | 0110 |
| 1011 | 1100 |
| 1100 | 0101 |
| 1101 | 1001 |
| 1110 | 0000 |
| 1111 | 0111 |

| Ciphertext | Plaintext |
|---|---|
| 0000 | 1110 |
| 0001 | 0011 |
| 0010 | 0100 |
| 0011 | 1000 |
| 0100 | 0001 |
| 0101 | 1100 |
| 0110 | 1010 |
| 0111 | 1111 |
| 1000 | 0111 |
| 1001 | 1101 |
| 1010 | 1001 |
| 1011 | 0110 |
| 1100 | 1011 |
| 1101 | 0010 |
| 1110 | 0000 |
| 1111 | 0101 |

  - 当n较大，且明密文之间的任意可逆变换。比如n=64。

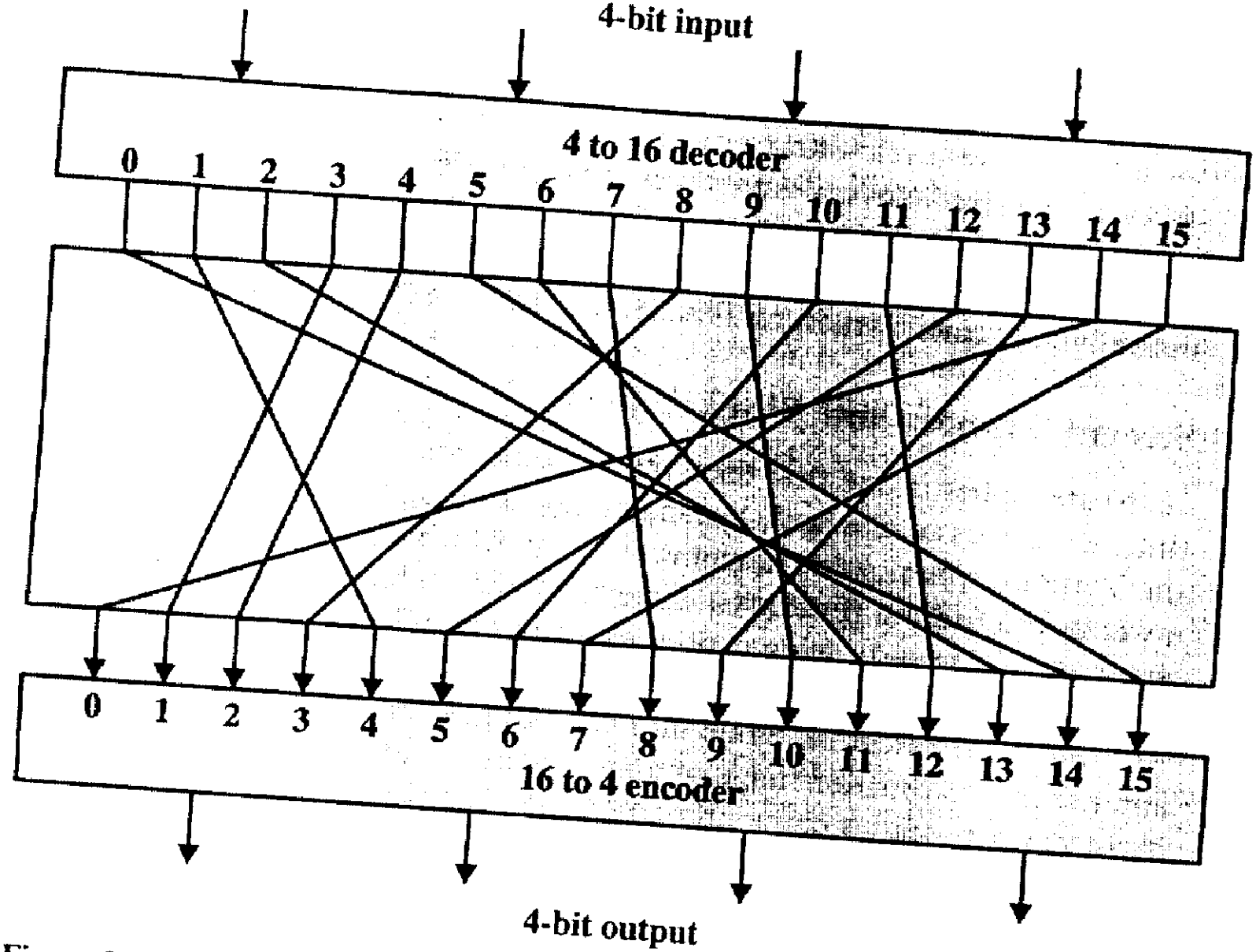
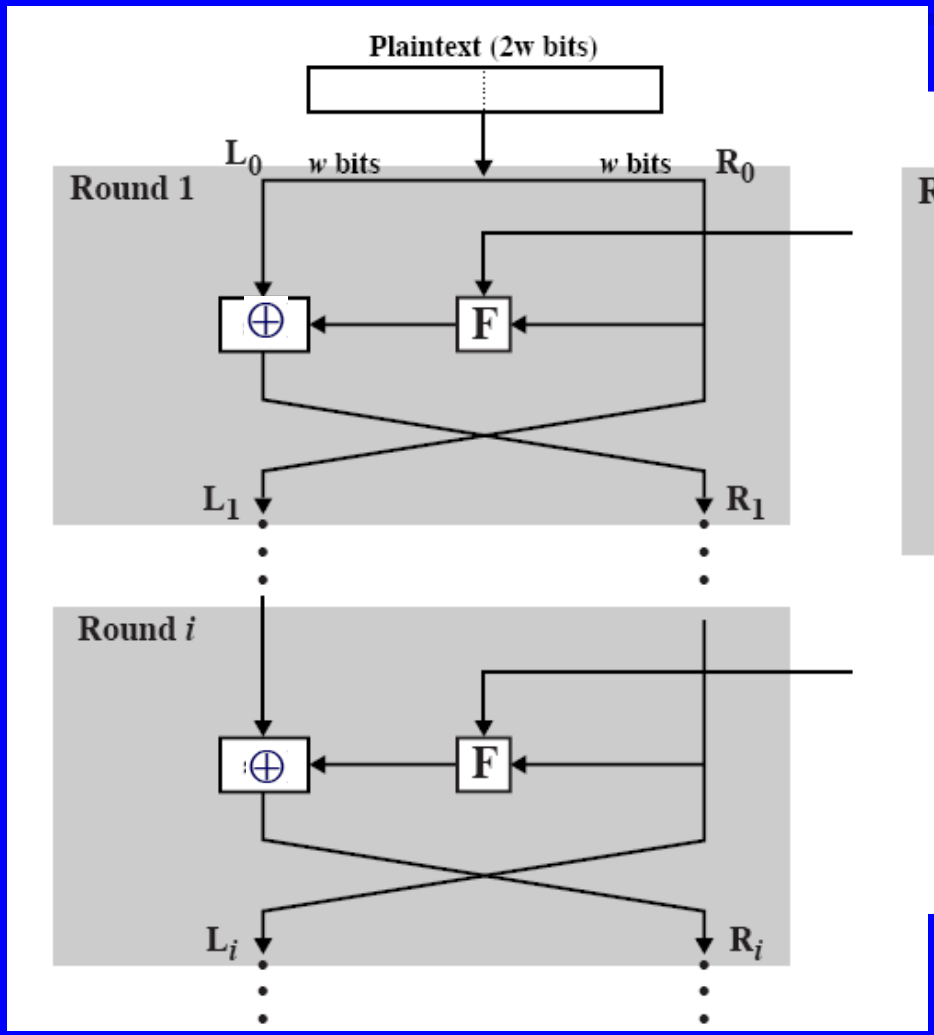
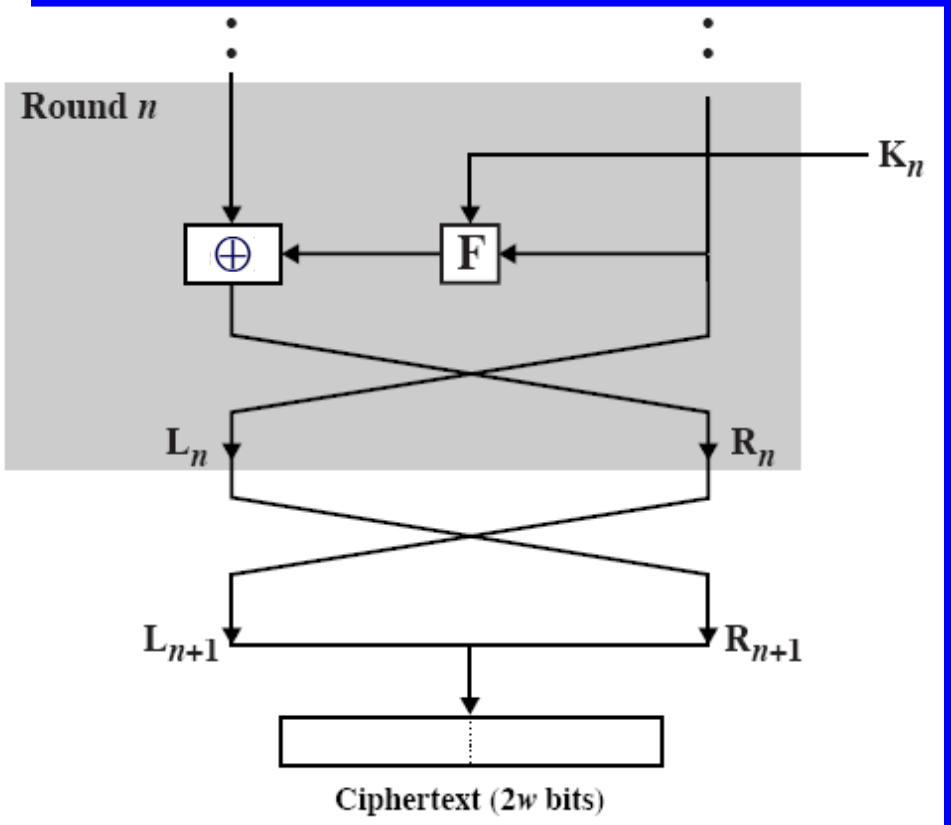



**Figure 3.4** General $n$-bit-$n$-bit Block Substitution (shown with $n = 4$).

- **问题1**：为什么把这个叫做理想分组密码？
  - 算法固定；
  - 明文和密文是一一对应的；
  - 密文的所有统计特征都是独立于所用密钥的。
- **问题2**： **Feistel网络的密钥空间**
  - 有多少个密钥：$2^n!$
  - 在**n**较小的时候是不安全的
  - 密钥的长度： $n \times 2^n$

# 经典Feistel网络



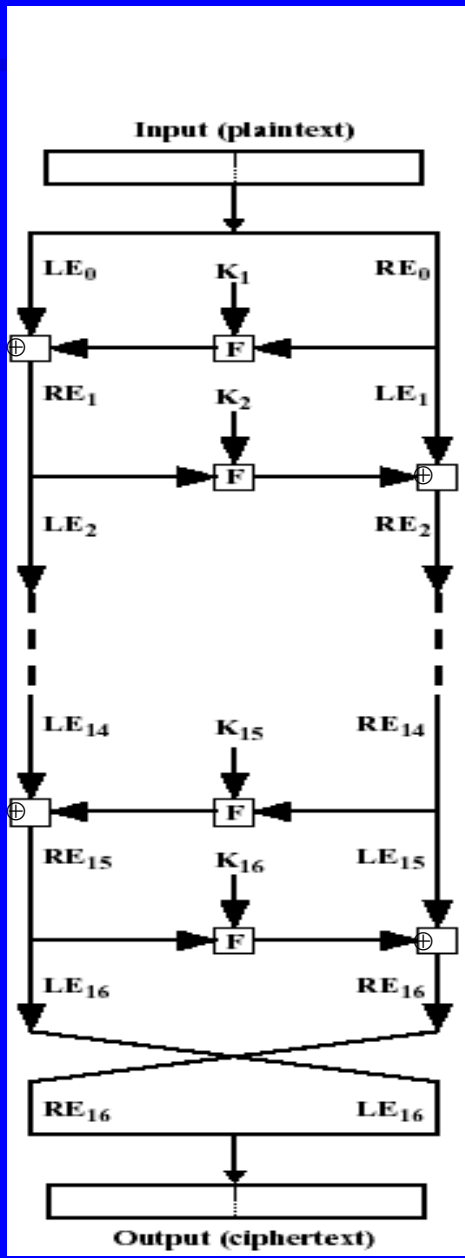



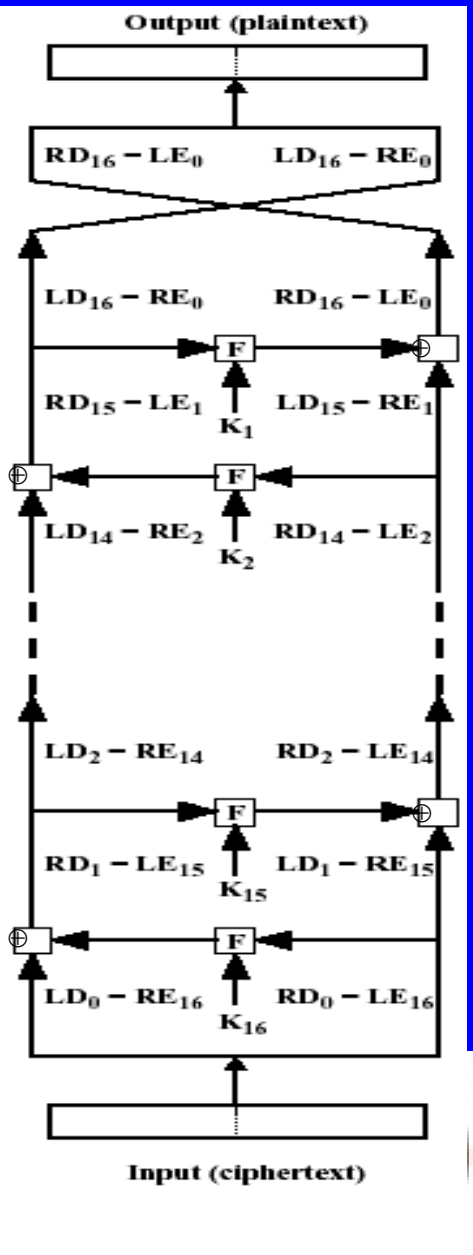

分组长度 密钥长度 迭代轮数 子密钥产生算法 轮函数

# Feistel加密与解密

- 加密: $L_i = R_{i-1}$; $R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$
- 解密: $R_{i-1} = L_i$

$$L_{i-1} = R_i \oplus F(R_{i-1}, K_i)$$
$$= R_i \oplus F(L_i, K_i)$$



# Feistel分组加密算法结构之思想

- 基本思想：用简单算法的乘积来近似表达大尺寸的替换变换
- 多个简单算法的结合得到的加密算法比任何一个部分算法都要强
- 交替使用替换变换和置换(permutation)
- 扩散(diffusion)和混淆(confusion)概念的对应

对付统计分析的方法：

（1）扩散

明文的统计特性消散在对应的密文中，这样可以让多个明文数字尽可能影响多个密文数字，等价于每个密文数字被许多明文数字影响

作用：尽可能明文和密文间统计关系复杂

（2）混淆

尽可能使密文和密钥之间的统计关系复杂，阻止攻击者发现密钥

# DES加密算法

**1973年5月15日，NBS开始公开征集标准加密算法，并公布了它的设计要求：**

**(1)算法必须提供高度的安全性**

**(2)算法必须有详细的说明，并易于理解**

**(3)算法的安全性取决于密钥，不依赖于算法**

**(4)算法适用于所有用户**

**(5)算法适用于不同应用场合**

**(6)算法必须高效、经济**

**(7)算法必须能被证实有效**

**(8)算法必须是可出口的**

# DES的产生

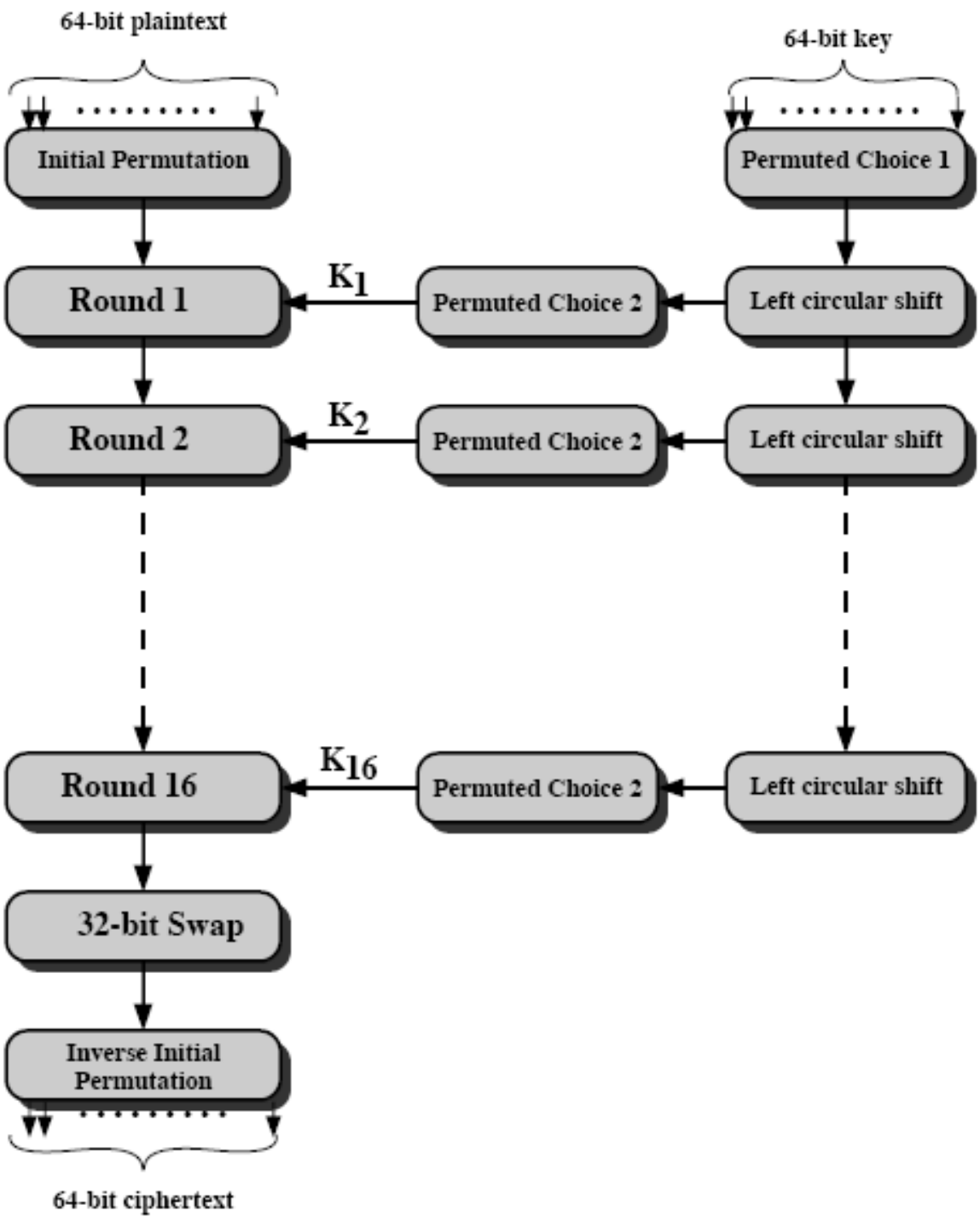
- 1974年8月27日，NBS开始第二次征集，IBM提交了算法LUCIFER，该算法由IBM的工程师在1971~1972年研制
- 1975年3月17日，NBS公开了全部细节
- 1976年,NBS指派了两个小组进行评价
- 1976年11月23日，采纳为联邦标准，批准用于非军事场合的各种政府机构
- 1977年1月15日，“数据加密标准”FIPS PUB 46发布

# DES的应用

- 1979年，美国银行协会批准使用
- 1980年，美国国家标准协会（ANSI）赞同DES作为私营使用的标准,称之为DEA（ANSI X.392）
- 1983年，国际化标准组织ISO赞同DES作为国际标准，称之为DEA-1
- 该标准规定每五年审查一次，计划十年后采用新标准
- 最近的一次评估是在1994年1月，已决定1998年12月以后，DES将不再作为联邦加密标准。
- 1999年NIST颁布标准（FIPS PUB 46-3）的新版本，规定DES只能用于遗留系统或3DES情况。

# DES的描述

- DES算法实现加密需要三个步骤：
- 第一步：初始置换（IP）
  - 对给定的64位比特的明文x，首先通过一个置换IP表来重新排列x，从而构造出64位比特的x0，x0=IP(x)=L0R0，其中L0表示x0的前32比特，R0表示x0的后32位。
- 第二步：16轮Fiestel结构迭代
  - 按照规则迭代。规则为
  - $L_i = R_{i-1}$
  - $R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$ （i=1,2,3…16）
- 第三步：末尾置换($IP^{-1}$)
  - 对L16R16利用$IP^{-1}$作逆置换，就得到了密文y。


64-bit plaintext
64-bit key
Initial Permutation
Permuted Choice 1
Round 1
K1
Permuted Choice 2
Left circular shift
Round 2
K2
Permuted Choice 2
Left circular shift
Round 16
K16
Permuted Choice 2
Left circular shift
32-bit Swap
Inverse Initial Permutation
64-bit ciphertext

# 初始置换IP和初始逆置换$IP^{-1}$

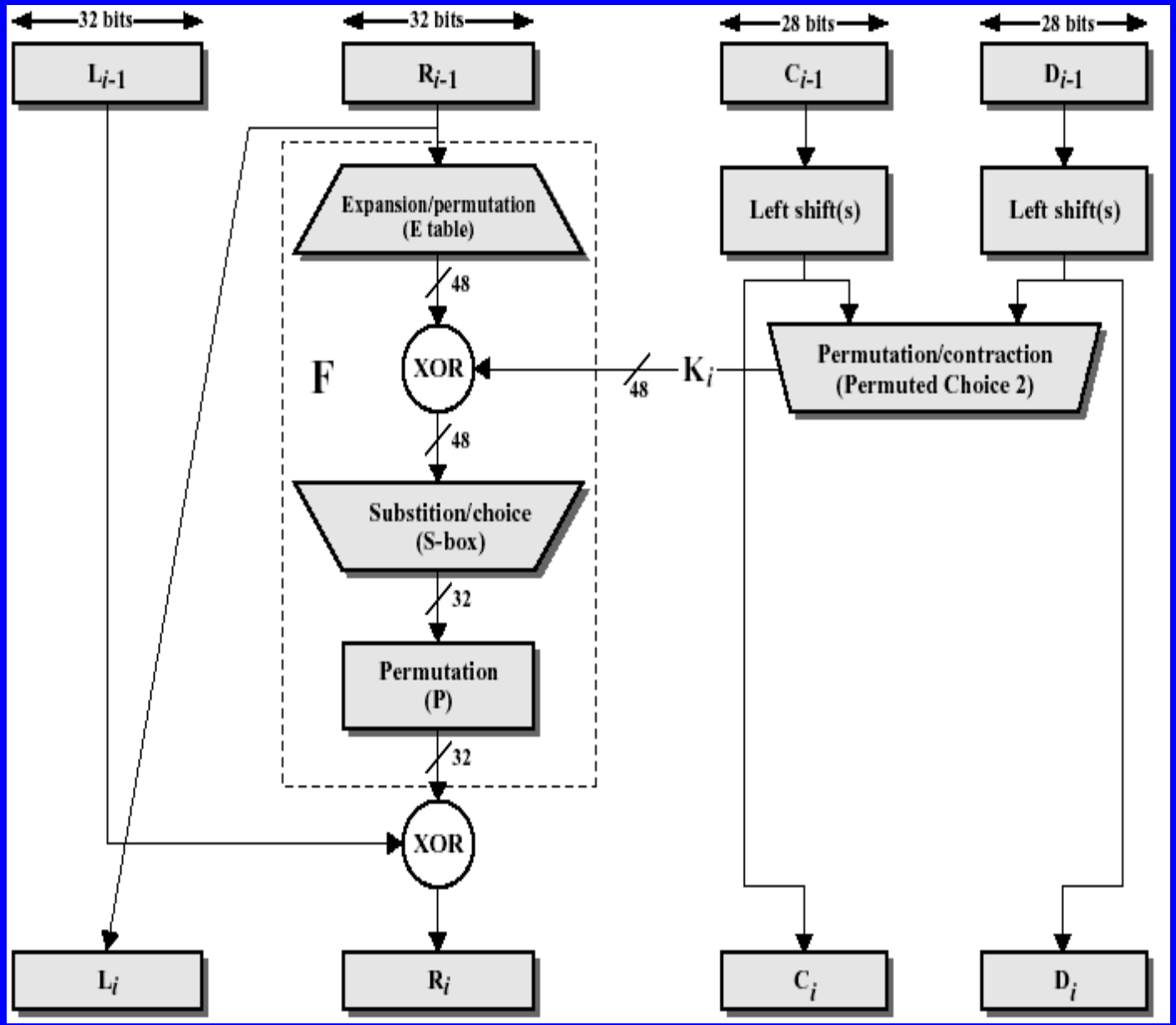
初始置换 IP

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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 |

初始逆置换 $IP^{-1}$

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

# DES的一轮迭代

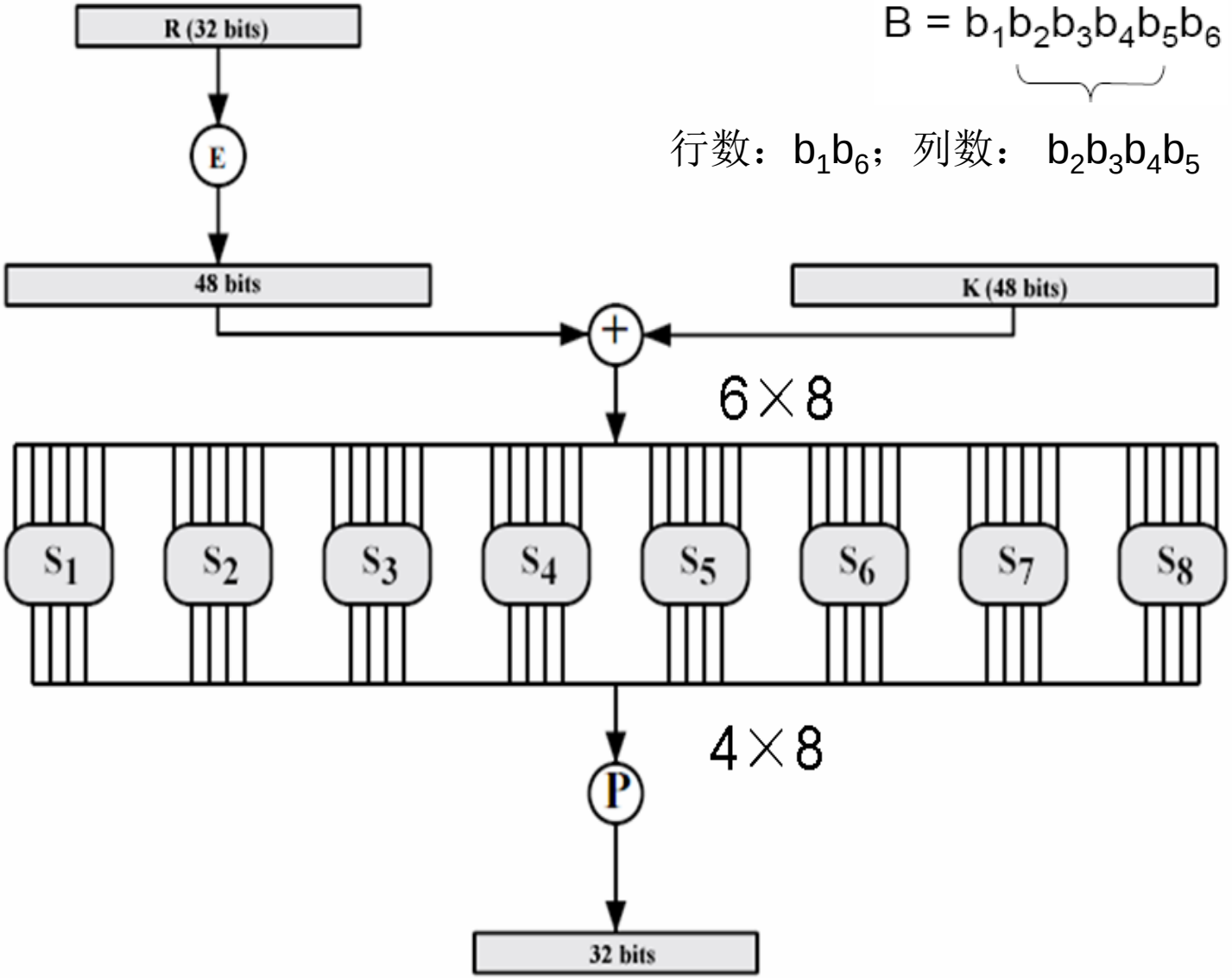

32 bits
32 bits
28 bits
28 bits
$L_{i-1}$
$R_{i-1}$
$C_{i-1}$
$D_{i-1}$
Expansion/permutation
(E table)
Left shift(s)
Left shift(s)
48
F
XOR
48
$K_i$
Permutation/contraction
(Permuted Choice 2)
48
Substition/choice
(S-box)
32
Permutation
(P)
32
XOR
$L_i$
$R_i$
$C_i$
$D_i$

# 扩充置换表（E）

- ... efgh ijkl mnop ...
- ... defghi hijklm lmnopq ...

| | | | | | | |
|---|---|---|---|---|---|---|
| 32 | | 01 | 02 | 03 | 04 | | 05 |
| 04 | | 05 | 06 | 07 | 08 | | 09 |
| 08 | | 09 | 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 | | 01 |

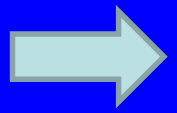
# 选择压缩运算




$B = b_1b_2b_3b_4b_5b_6$

行数：$b_1b_6$；列数：$b_2b_3b_4b_5$

# S-Box-i



| | 行\列 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_1$ | 0 | 14 | 4 | 13 | 1 | 2 | 15 | 11 | 8 | 3 | 10 | 6 | 12 | 5 | 9 | 0 | 7 |
| | 1 | 0 | 15 | 7 | 4 | 14 | 2 | 13 | 1 | 10 | 6 | 12 | 11 | 9 | 5 | 3 | 8 |
| | 2 | 4 | 1 | 14 | 8 | 13 | 6 | 2 | 11 | 15 | 12 | 9 | 7 | 3 | 10 | 5 | 0 |
| | 3 | 15 | 12 | 8 | 2 | 4 | 9 | 1 | 7 | 5 | 11 | 3 | 14 | 10 | 0 | 6 | 13 |
| $S_2$ | 0 | 15 | 1 | 8 | 14 | 6 | 11 | 3 | 4 | 9 | 7 | 2 | 13 | 12 | 0 | 5 | 10 |
| | 1 | 3 | 13 | 4 | 7 | 15 | 2 | 8 | 14 | 12 | 0 | 1 | 10 | 6 | 9 | 11 | 5 |
| | 2 | 0 | 14 | 7 | 11 | 10 | 4 | 13 | 1 | 5 | 8 | 12 | 6 | 9 | 3 | 2 | 15 |
| | 3 | 13 | 8 | 10 | 1 | 3 | 15 | 4 | 2 | 11 | 6 | 7 | 12 | 0 | 5 | 14 | 9 |
| $S_3$ | 0 | 10 | 0 | 9 | 14 | 6 | 3 | 15 | 5 | 1 | 13 | 12 | 7 | 11 | 4 | 2 | 8 |
| | 1 | 13 | 7 | 0 | 9 | 3 | 4 | 6 | 10 | 2 | 8 | 5 | 14 | 12 | 11 | 15 | 1 |
| | 2 | 13 | 6 | 4 | 9 | 8 | 15 | 3 | 0 | 11 | 1 | 2 | 12 | 5 | 10 | 14 | 7 |
| | 3 | 1 | 10 | 13 | 0 | 6 | 9 | 8 | 7 | 4 | 15 | 14 | 3 | 11 | 5 | 2 | 12 |
| $S_4$ | 0 | 7 | 13 | 14 | 3 | 0 | 6 | 9 | 10 | 1 | 2 | 8 | 5 | 11 | 12 | 4 | 15 |
| | 1 | 13 | 8 | 11 | 5 | 6 | 15 | 0 | 3 | 4 | 7 | 2 | 12 | 1 | 10 | 14 | 9 |
| | 2 | 10 | 6 | 9 | 0 | 12 | 11 | 7 | 13 | 15 | 1 | 3 | 14 | 5 | 2 | 8 | 4 |
| | 3 | 3 | 15 | 0 | 6 | 10 | 1 | 13 | 8 | 9 | 4 | 5 | 11 | 12 | 7 | 2 | 14 |

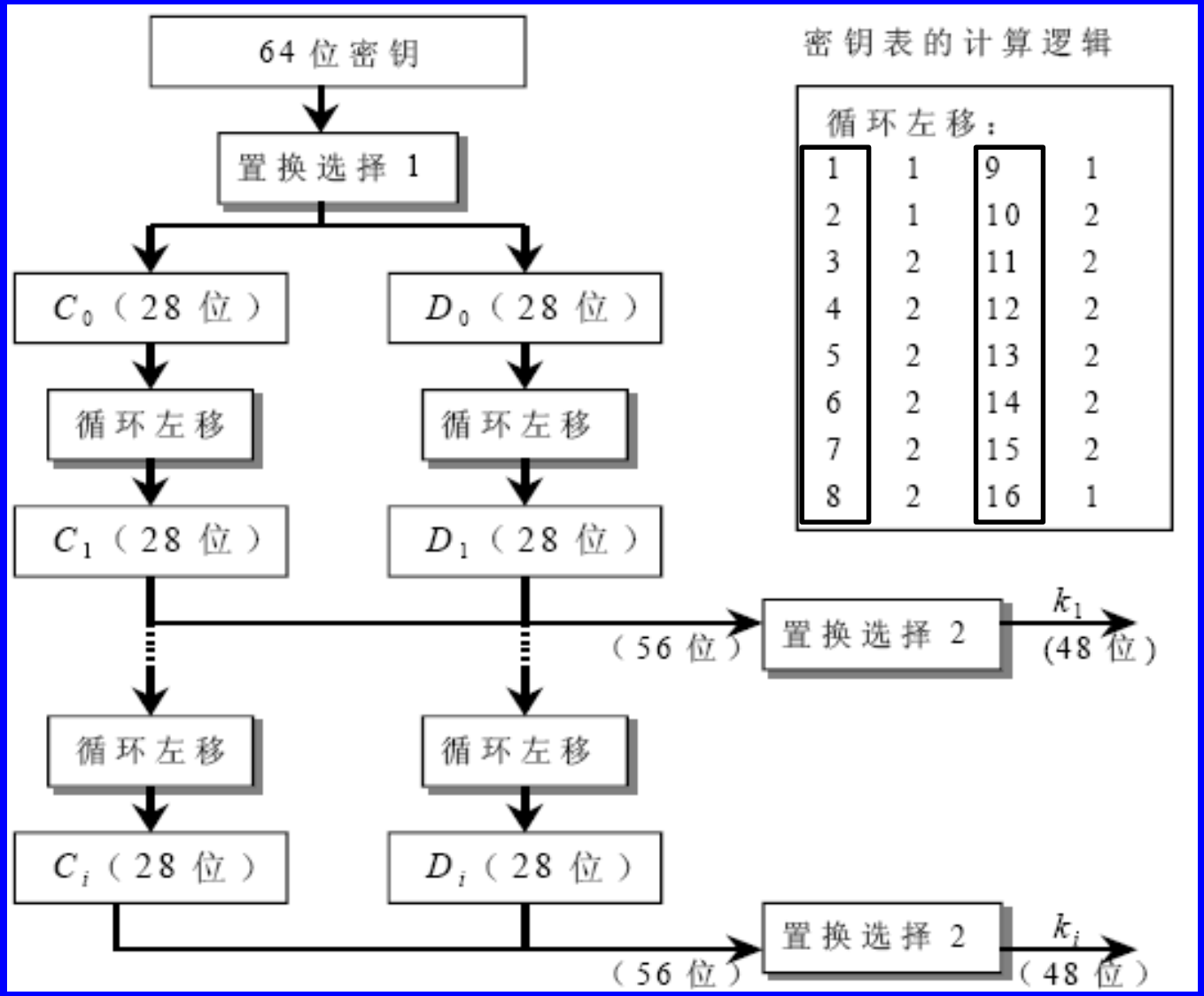
# S-Box-ii

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_3$ | 0 | 2 | 12 | 4 | 1 | 7 | 10 | 11 | 6 | 8 | 5 | 3 | 15 | 13 | 0 | 14 | 9 |
| | 1 | 14 | 11 | 2 | 12 | 4 | 7 | 13 | 1 | 5 | 0 | 15 | 10 | 3 | 9 | 8 | 6 |
| | 2 | 4 | 2 | 1 | 11 | 10 | 13 | 7 | 8 | 15 | 9 | 12 | 5 | 6 | 3 | 0 | 14 |
| | 3 | 11 | 8 | 12 | 7 | 1 | 14 | 2 | 13 | 6 | 15 | 0 | 9 | 10 | 4 | 5 | 3 |
| $S_4$ | 0 | 12 | 1 | 10 | 15 | 9 | 2 | 6 | 8 | 0 | 13 | 3 | 4 | 14 | 7 | 5 | 11 |
| | 1 | 10 | 15 | 4 | 2 | 7 | 12 | 9 | 5 | 6 | 1 | 13 | 14 | 0 | 11 | 3 | 8 |
| | 2 | 9 | 14 | 15 | 5 | 2 | 8 | 12 | 3 | 7 | 0 | 4 | 10 | 1 | 13 | 11 | 6 |
| | 3 | 4 | 3 | 2 | 12 | 9 | 5 | 15 | 10 | 11 | 14 | 1 | 7 | 6 | 0 | 8 | 13 |
| $S_7$ | 0 | 4 | 11 | 2 | 14 | 15 | 0 | 8 | 13 | 3 | 12 | 9 | 7 | 5 | 10 | 6 | 1 |
| | 1 | 13 | 0 | 11 | 7 | 4 | 9 | 1 | 10 | 14 | 3 | 5 | 12 | 2 | 15 | 8 | 6 |
| | 2 | 1 | 4 | 11 | 13 | 12 | 3 | 7 | 14 | 10 | 15 | 6 | 8 | 0 | 5 | 9 | 2 |
| | 3 | 6 | 11 | 13 | 8 | 1 | 4 | 10 | 7 | 9 | 5 | 0 | 15 | 14 | 2 | 3 | 12 |
| $S_8$ | 0 | 13 | 2 | 8 | 4 | 6 | 15 | 11 | 1 | 10 | 9 | 3 | 14 | 5 | 0 | 12 | 7 |
| | 1 | 1 | 15 | 13 | 8 | 10 | 3 | 7 | 4 | 12 | 5 | 6 | 11 | 0 | 14 | 9 | 2 |
| | 2 | 1 | 11 | 4 | 1 | 9 | 12 | 14 | 2 | 0 | 6 | 10 | 13 | 15 | 3 | 5 | 8 |
| | 3 | 2 | 1 | 14 | 7 | 4 | 10 | 8 | 13 | 15 | 12 | 9 | 0 | 3 | 5 | 6 | 11 |

# 置 换 表（P）

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 16 | 07 | 20 | 21 | 29 | 12 | 28 | 17 |
| 01 | 15 | 23 | 26 | 05 | 18 | 31 | 10 |
| 02 | 08 | 24 | 14 | 32 | 27 | 03 | 09 |
| 19 | 13 | 30 | 06 | 22 | 11 | 04 | 25 |

# 子密钥的产生


64 位密钥
置换选择 1
$C_0$（28 位）
$D_0$（28 位）
循环左移
循环左移
$C_1$（28 位）
$D_1$（28 位）
（56 位）
置换选择 2
$k_1$
(48 位)
循环左移
循环左移
$C_i$（28 位）
$D_i$（28 位）
（56 位）
置换选择 2
$k_i$
（48 位）
密钥表的计算逻辑
循环左移：
1 1 9 1
2 1 10 2
3 2 11 2
4 2 12 2
5 2 13 2
6 2 14 2
7 2 15 2
8 2 16 1

# 输入密钥

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

# 置换选择1和置换选择2

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

# 对DES的讨论

- DES的雪崩效应
- F函数(S-Box)设计原理
- 密钥长度的争论
- DES的破译
- 弱密钥与半弱密钥

# DES的雪崩效应

- 雪崩效应：明文或密钥的微小改变对密文产生很大的影响。

# DES的雪崩效应

| (a) Change in Plaintext | | (b) Change in Key | |
|---|---|---|---|
| Round | Number of bits that differ | Round | Number of bits that differ |
| 0 | 1 | 0 | 0 |
| 1 | 6 | 1 | 2 |
| 2 | 21 | 2 | 14 |
| 3 | 35 | 3 | 28 |
| 4 | 39 | 4 | 32 |
| 5 | 34 | 5 | 30 |
| 6 | 32 | 6 | 32 |
| 7 | 31 | 7 | 35 |
| 8 | 29 | 8 | 34 |
| 9 | 42 | 9 | 40 |
| 10 | 44 | 10 | 38 |
| 11 | 32 | 11 | 31 |
| 12 | 30 | 12 | 33 |
| 13 | 30 | 13 | 28 |
| 14 | 26 | 14 | 26 |
| 15 | 29 | 15 | 34 |
| 16 | 34 | 16 | 35 |

# 密钥长度

- 关于DES算法的另一个最有争议的问题：56比特的密钥长度不足以抵御穷举式攻击，因为密钥量只有$2^{56}$约$10^{17}$个。

原因：早在1977年，Diffie和Hellman已建议制造一个每秒能测试100万个密钥的VLSI芯片。每秒测试100万个密钥的机器大约需要一天就可以搜索整个密钥空间。他们估计制造这样的机器大约需要2000万美元。

# 密钥长度

- 密钥长度多少才合适呢？
- 加密系统怎样才是安全的呢？

- 1997年1月28日，美国的RSA数据安全公司在RSA安全年会上公布了一项“秘密密钥挑战”竞赛，其中包括悬赏1万美元破译密钥长度为56比特的DES。美国克罗拉多洲的程序员Verser从1997年2月18日起，用了96天时间，在Internet上数万名志愿者的协同工作下，成功地找到了DES的密钥，赢得了悬赏的1万美元。
- 1998年7月电子前沿基金会（EFF）使用一台25万美圆的电脑在56小时内破译了56比特密钥的DES。
- 1999年1月RSA数据安全会议期间，电子前沿基金会用22小时15分钟就宣告破解了一个DES的密钥。

# DES的安全性分析

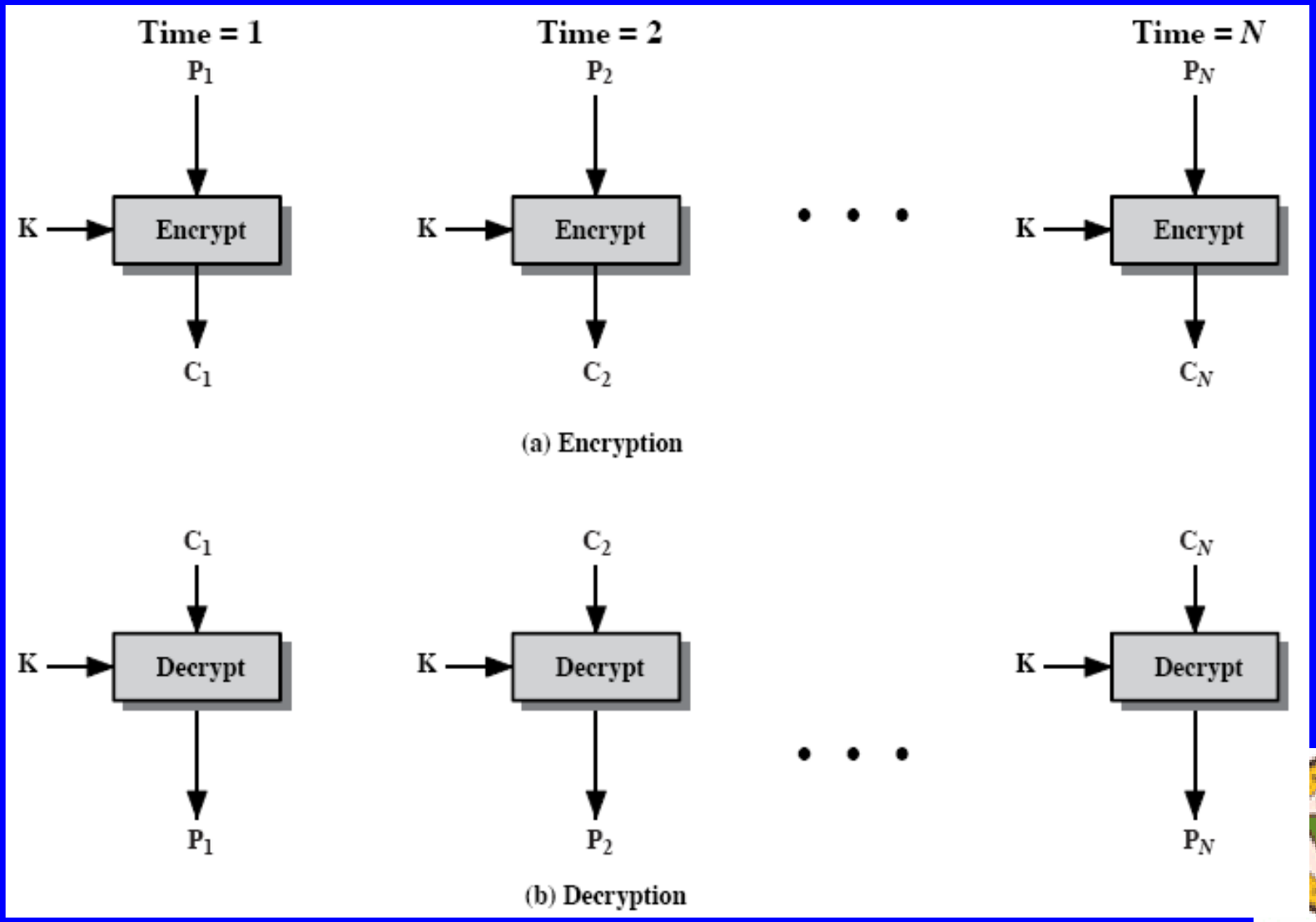
- S盒的输入数组改变1位，输出数组至少改变2位。如：输入：011001 行：01 列：1100，输出：1001。
- 当输入的某1位固定，而其他各位改变时，S盒输出数据的1的个数和0的个数相差很小。否则若干次变换后，输出数组全0或1。
- 雪崩效应
- 16轮迭代是一个饱和状态。次数太多或太少，都会影响雪崩的效果。

# 分组密码的操作模式

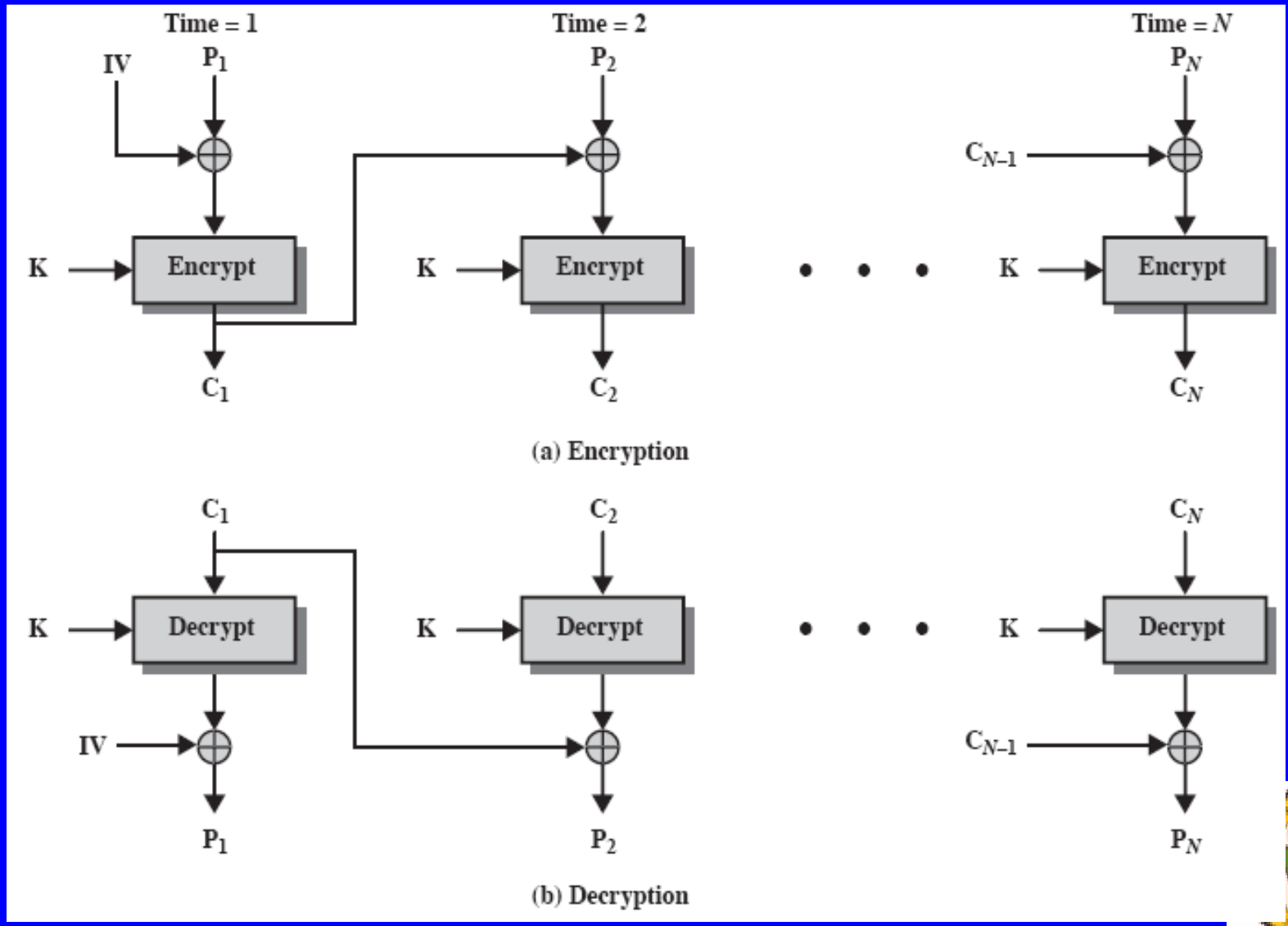
- 电子密码本ECB (electronic codebook mode)
- 密码分组链接CBC (cipher block chaining)
- 密码反馈CFB (cipher feedback)
- 输出反馈OFB (output feedback)
- 计数器模式(CTR)

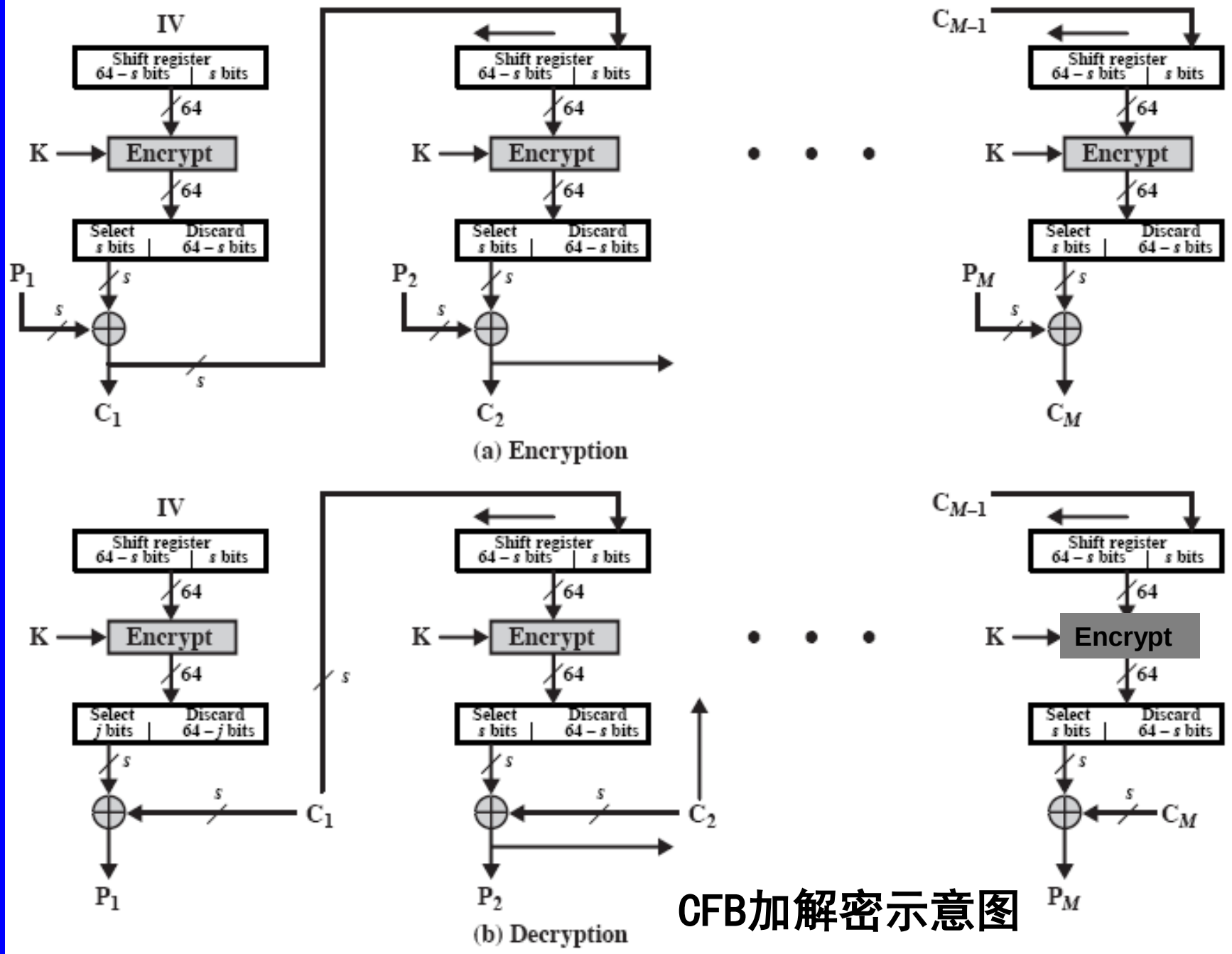
# 电子密码本（ECB）

$C_i = E_K(P_i) \Leftrightarrow P_i = D_K(C_i)$

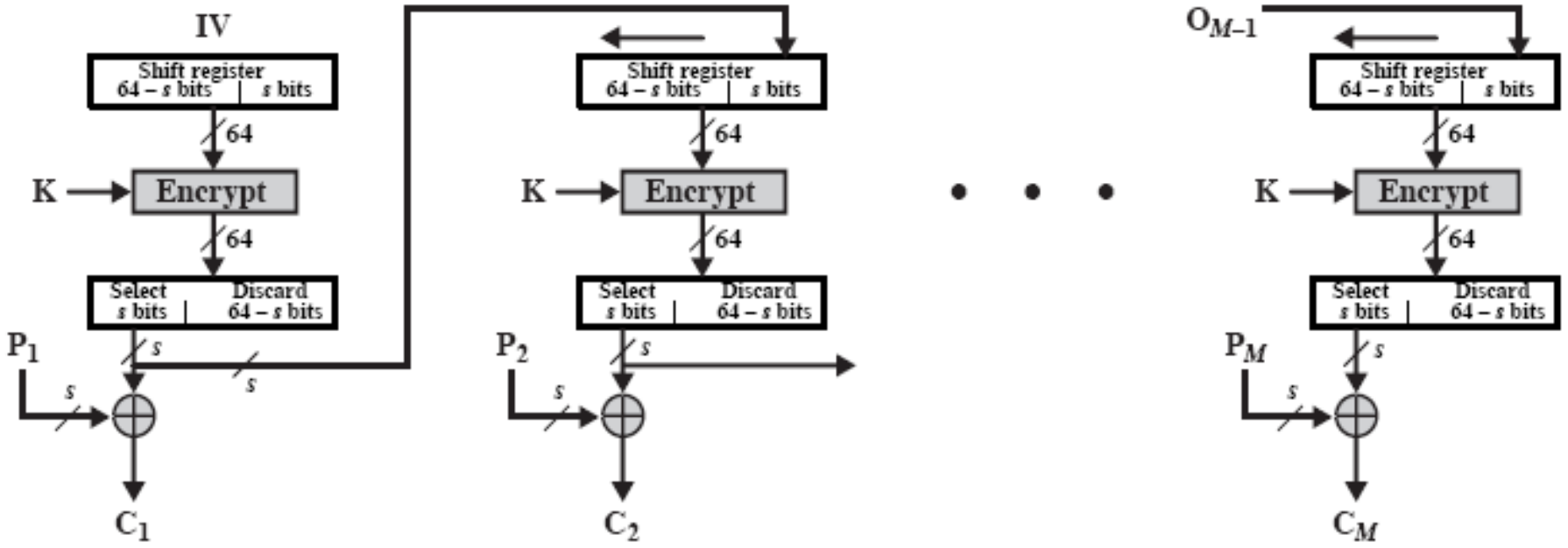


# 密码分组链接CBC

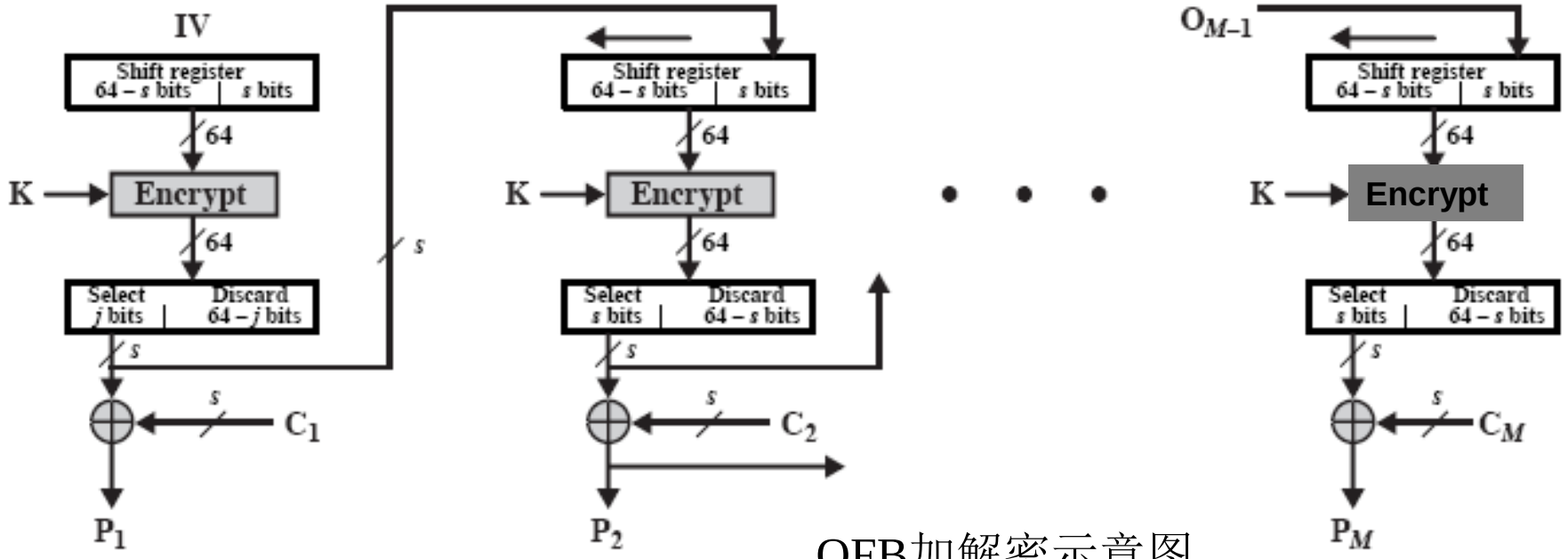

Time = 1
Time = 2
Time = N
IV
$P_1$
$P_2$
$P_N$
$C_{N-1}$
K
Encrypt
$C_1$
$C_2$
$C_N$
(a) Encryption
Decrypt
(b) Decryption


IV
Shift register
64 – s bits
s bits
64
K
Encrypt
64
Select
s bits
Discard
64 – s bits
s
P1
C1
P2
C2
CM–1
PM
CM
(a) Encryption
Select
j bits
Discard
64 – j bits
(b) Decryption

CFB加解密示意图



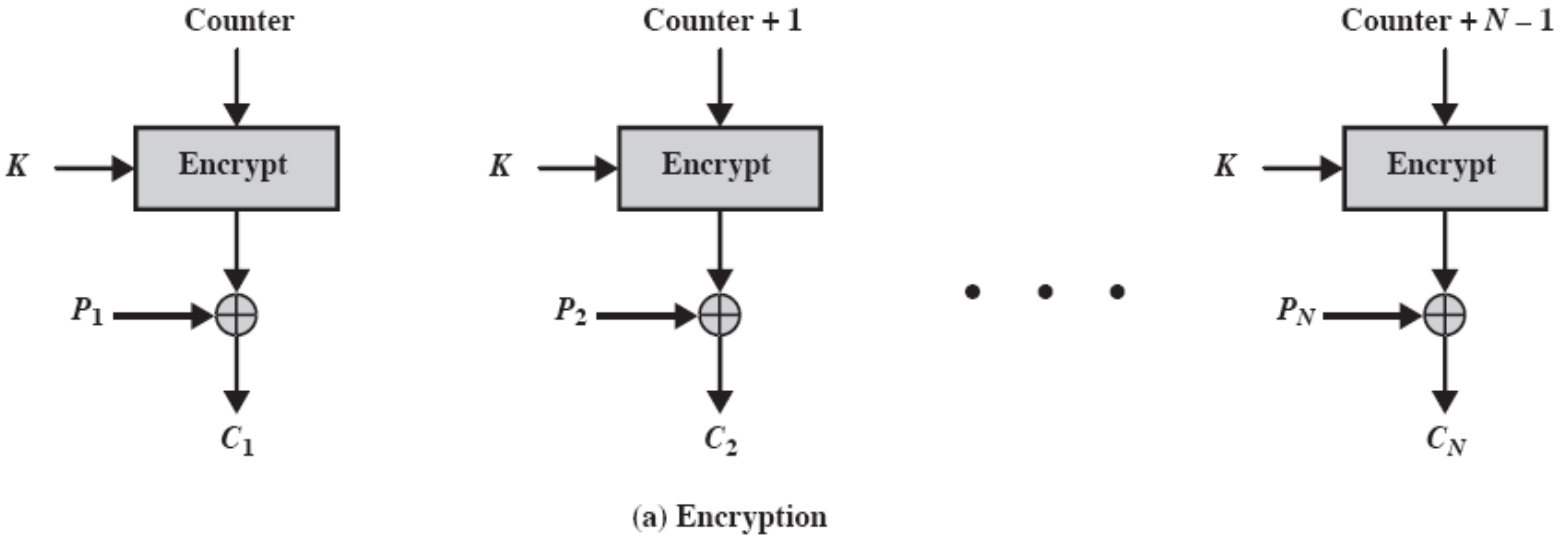

(a) Encryption



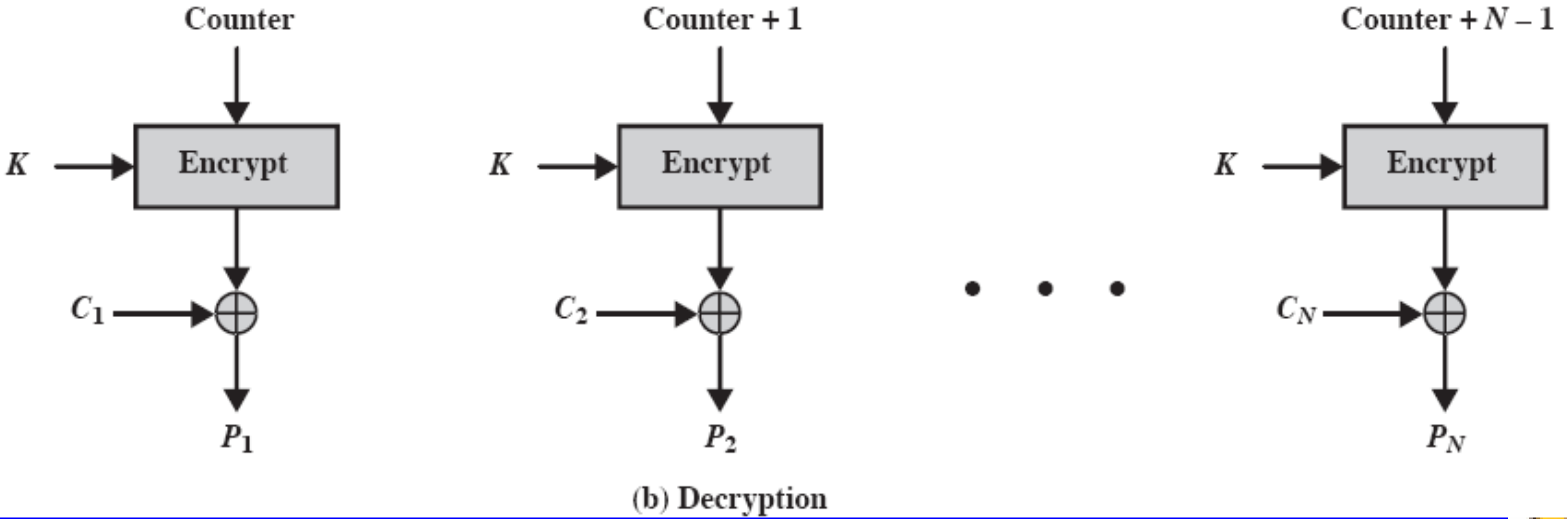

(b) Decryption

OFB加解密示意图

# 计数器CTR模式



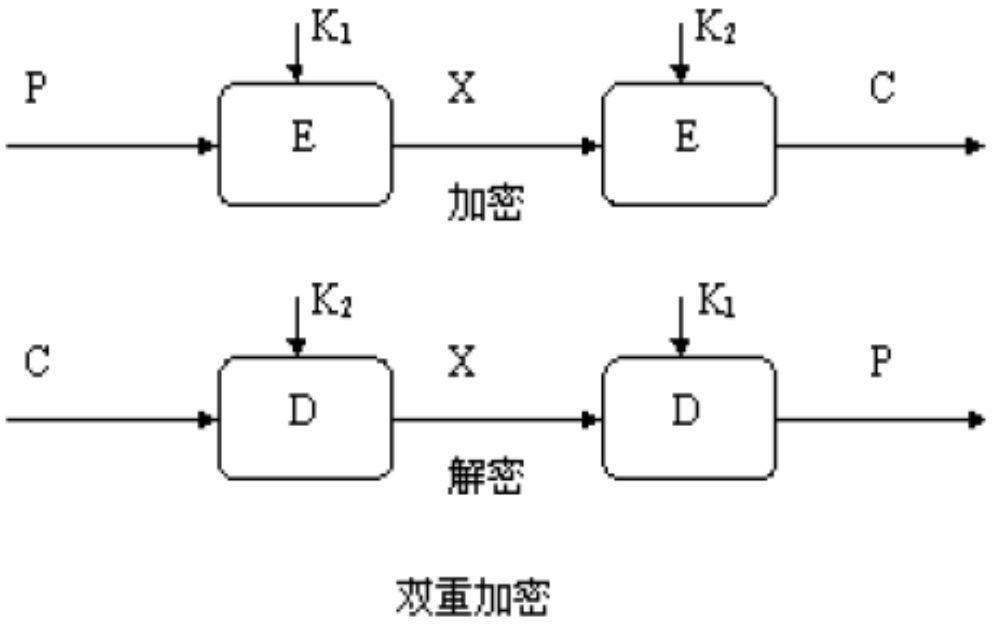

(a) Encryption




(b) Decryption

# 双重DES (Double DES)

$$C = E_{K2}(E_{K1}(P)) \Leftrightarrow P = D_{K1}(D_{K2}(C))$$



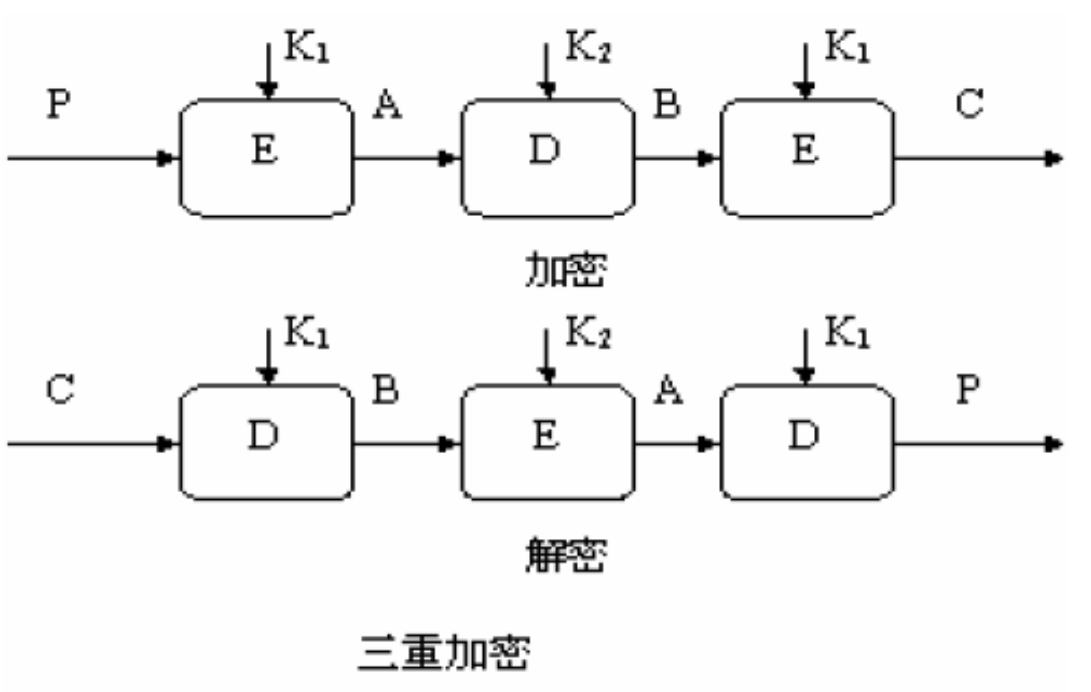

双重加密

# Triple-DES的四种模型

- DES-EEE3：三个不同密钥，顺序使用三次加密算法
- DES-EDE3：三个不同密钥，依次使用加密-解密-加密算法
- DES-EEE2：K1=K3，同上
- DES-EDE2：K1=K3，同上

# 双密钥的三重DES

$$C=E_{K1}(D_{K2}(E_{K1}(P))) \Leftrightarrow P=D_{K1}(E_{K2}(D_{K1}(C)))$$



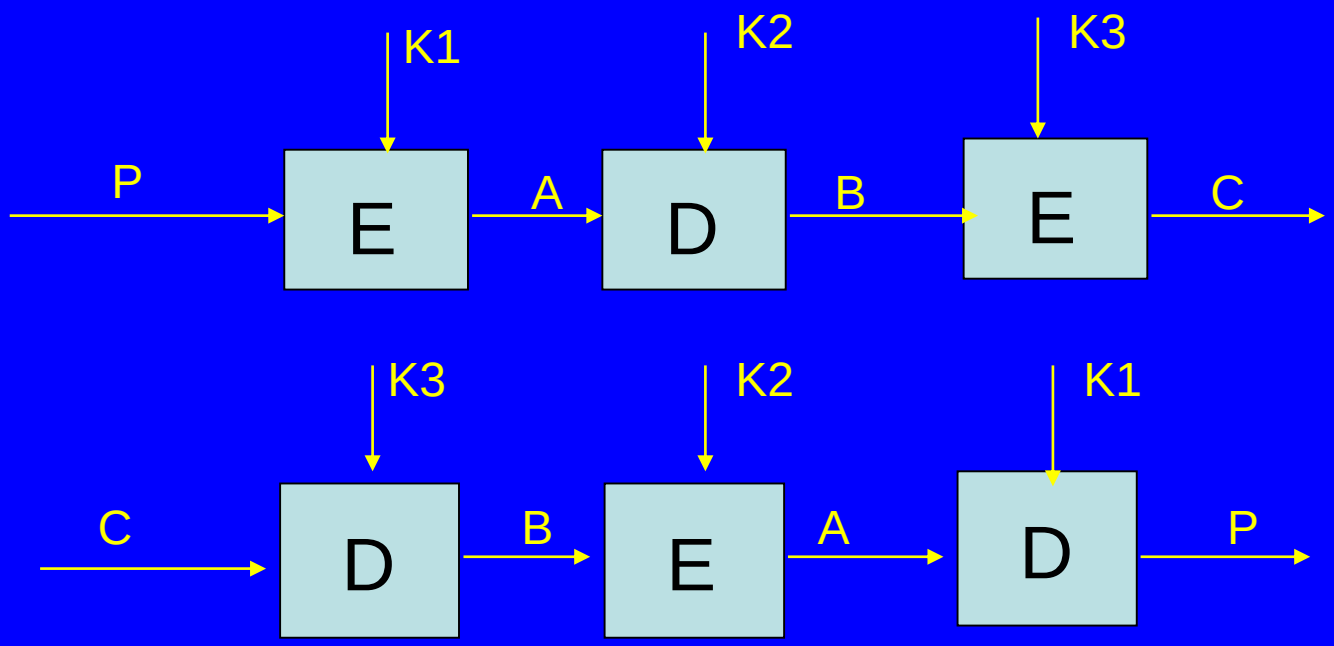
# 对双密钥的三重DES的分析

- 该模式由IBM设计，可与常规加密算法兼容
- 这种替代DES的加密较为流行并且已被采纳用于密钥管理标准（The Key Manager Standards ANSX9.17和ISO8732).
- 交替使用$K_1$和$K_2$可以抵抗中间相遇攻击.如果$C=E_{K2}(E_{K1}(E_{K1}(P)))$，只需要$2^{56+2}$次加密
- 到目前为止，还没有人给出攻击三重DES的有效方法。对其密钥空间中密钥进行蛮干搜索，那么由于空间太大为$2^{112}=5\times10^{33}$，这实际上是不可行的。若用差分攻击的方法，相对于单一DES来说复杂性以指数形式增长，要超过$10^{52}$。

# 三密钥的三重DES

加密：$C=E_{K3}(D_{K2}(E_{K1}(P)))$

解密：$P=D_{K1}(E_{K2}(D_{K3}(C)))$

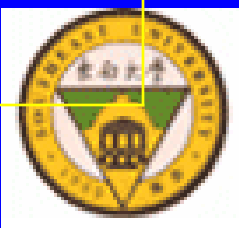


# 三密钥的三重DES分析

- 密钥的有效长度为168位
- 与DES的兼容性可以通过令$K_3=K_2$或$K_1=K_2$得到
- 许多基于Internet的应用里用到：PGP和S/MIME

# 几种对称密码算法性能的比较

| 算法 | 每轮所用的时钟周期数 | 轮数 | 加密一字节所用的时钟周期数 |
|---|---|---|---|
| Blowfish | 9 | 16 | 18 |
| RC5 | 12 | 16 | 23 |
| DES | 18 | 16 | 45 |
| IDEA | 50 | 8 | 50 |
| 3DES | 18 | 48 | 108 |



# 问 题

1、加密算法的安全性依赖于什么？

2、选择加密算法的原则

# 差分密码分析原理

- 概念
  - 差分分析（**differential cryptanalysis)**方法是一种选择明文攻击。该方法的基本思想是：通过分析一对特选的明文对的差相应密文对的差的影响来提取密钥信息。这种攻击方法主要适用于攻击迭代密码。

- 概念
  - 对分组长度为n的r轮迭代密码，将两个n比特串x与x’的差分定义为：

$$\Delta x = x \oplus x'$$

其中⊕表示模2运算

如果给定一对n长的明文m和m’，那么在密钥的控制下，第i轮迭代所产生的中间密文差为：

$$\Delta c(i) = c(i) \oplus c'(i) \quad 0 \le i \le r$$

- **S盒的差分表**

| Input $x'$ | Output $y'$ 0 1 | 2 3 | 4 5 6 | 7 8 | 9 | A B | C D | E F |
|---|---|---|---|---|---|---|---|---|
| 00 | 64 0 | 0 0 | 0 0 0 | 0 0 | 0 | 0 0 | 0 0 | 0 0 |
| 01 | 0 0 | 0 6 | 0 2 4 | 4 0 | 10 | 12 4 | 10 6 | 2 4 |
| 02 | 0 0 | 0 8 | 0 4 4 | 4 0 | 6 | 8 6 | 12 6 | 4 2 |
| 03 | 14 4 | 2 2 | 10 6 4 | 2 6 | 4 | 4 0 | 2 2 | 2 0 |
| ⋮ | | | | | | | | |
| 0C | 0 0 | 0 8 | 0 6 6 | 0 0 | 6 | 6 4 | 6 6 | 14 2 |
| ⋮ | | | | | | | | |
| 34 | 0 8 | 16 6 | 2 0 0 | 12 6 | 0 | 0 0 | 0 8 | 0 6 |
| ⋮ | | | | | | | | |
| 3E | 4 8 | 2 2 | 2 4 4 | 14 4 | 2 | 0 2 | 0 8 | 4 4 |
| 3F | 4 4 | 4 2 | 4 0 2 | 4 4 | 2 | 4 8 | 8 6 | 2 2 |

0 →

6 3 →

$=2^6-1$

出现的次数 ←

因为，ki是第i轮迭代的子密钥，所以

$$c(i) = f(c(i-1), ki)$$

$$\Delta c(i) = f(c(i-1), ki) - f(c'(i-1), ki)$$

这里，每轮迭代所用的子密钥ki与密文统计独立，且可以认为它服从均匀分布。

# 差分密码分析原理

当i = 0时

$c(0) = m, c'(0) = m'$,

$\Delta c(0) = \Delta m = m \oplus m'$

当i = r 时

$\Delta c = \Delta c(r)$

例如：在DES中，令$m= L_0R_0$，$m'= L_0'R_0'$

如果$R_0 = R_0'$，则明文差为：

$$\Delta m = L_0R_0 \oplus L_0' R_0' = (L_0^*, 0)$$

其中，$L_0^* = L_0 \oplus L_0'$

如果DES只有一轮迭代，而不是16论迭代，那么根据DES轮函数可以计算出

$L_1 = R_0$

$L_1' = R_0'$

$L_1 \oplus L_1' = 0$

$R_1 = L_0 \oplus f(R_0, k1)$

$R_1' = L_0' \oplus f(R_0', k1)$

$R_1 \oplus R_1' = L_0^*$

所以

$L_1 R_1 \oplus L_1' R_1' = (0, L_0^*)$

说明：

- 这个差分一轮的概率是1。这表明，随机的选择一对输入差为 $(L_0^*, 0)$的明文，经过DES的一轮迭代后的密文差为 $(0, L_0^*)$的概率为1。

例如：在DES中，令$m= L_0R_0$，$m'= L_0'R_0'$

如果$R_0 \oplus R_0' = (60000000)$。用16进制表示32位

则明文差为：

$$\partial_0 = \Delta m = m \oplus m' = (L_0 \oplus L_0', R_0 \oplus R_0')$$

$$= (L_0^*, 60000000)$$

$L_1 = R_0, L_1' = R_0' \rightarrow L_1 \oplus L_1' = 60000000$

$R_1 = L_0 \oplus f(R_0,k1), R_1' = L_0' \oplus f(R_0',k1)$

$\rightarrow R_1 \oplus R_1' = (L_0 \oplus L_0') \oplus f(R_0,k_1) \oplus f(R_0',k1)$

$= (L_0 \oplus L_0') \oplus P(S(E(R_0) \oplus k1)) \oplus P(S(E(R_0') \oplus k1))$

$= L_0^* \oplus P\{S[E(R_0) \oplus k1] \oplus S[E(R_0') \oplus k1]\}$

因为 $R_0$和$R_0'$ 经过相同的扩展运算后分别于K1模2相加，所以选择压缩运算S盒的输入差为：

$E(R_0) \oplus K1 \oplus E(R_0') \oplus K1$

$= E(R_0) \oplus E(R_0')$

$= E(R_0 \oplus R_0') = E(60000000)$

=(001100,000000,000000,000000,000000,000000,000000,000000)

因为S2盒至S8盒的输入差都是000000，其输出差都是0000的概率为1。问：

1、而S1盒的输入差为001100，其对应多少对输入?

2、其输出差有多少种取值？

$2^6=64$

$2^4=16$

这样的一轮迭代的特征是$\Omega = \partial_0\partial_1$。这说明，随机地选择一对输入差为（$L_0^*$，60000000）的明文，经过一轮迭代后的输出差为

$$\partial 1 = （60000000，\ L_0^* \oplus 00808200）$$

的概率是14/64，而得到其他形式的密文差是相当随机且具有的概率都很小。

简单地说就是：某些输入差对应的输出差具有较大的概率，这就是我们需要的信息

例如：

(000000) ⊕(001100) = (001100)

(000001) ⊕(001101) = (001100)

S(000000) ⊕S(001100) = 0101

S(000001) ⊕S(001101) = 1101

S1盒的输入差为001100，其输出差为1110的概率是14/64，于是，选择S的输出差为E0000000的概率为14/64，再经过置换运算P，它的输出差为00808200，故：

$$\partial 1 = （60000000，\ L_0^* \oplus 00808200）$$

$$L_0^* = L_0 \oplus L_0'$$

正是由于输入差和输出差的分布表的不均匀性，才导致了差分分析的可能性

# 差分的步骤

- 对r轮迭代密码的差分攻击步骤为：
  - 找出一个 (r-1) 论特征$\partial_0\partial_1 \ldots\ldots \partial_{r-1}$，使其概率达到 最大
  - 均匀随机地选择明文m，并计算出m'，使 $m \oplus m' = \partial_0$，再找出m和m'在实际密钥加密下所得的密文c(r)和c'(r)。
  - 如果最后一轮的子密钥kr（或kr的部分比特）有2^m个可能的值，我们设置2^m相应的计数器，用每一个可能的密钥解密c(r)和c'(r)。如果$c(r-1) \oplus c'(r-1) = \partial_{r-1}$，则相应的计数器加1.

# 差分的步骤

- 对r轮迭代密码的差分攻击步骤为：
  - 重复第2,3步，直到有一个或几个计算器的值明显高于其他计数器的值，输出它们所对应的子密钥（或部分比特）

# AES加密算法

- 3DES的优点和缺点
  - 优点：
    - 密钥长度168位，可以抗穷举攻击；
    - 底层加密算法采用DES，具有很强的抗密码分析能力
  - 缺点：
    - 轮的数量多，计算速度慢
    - 分组长度为64位，效率低
    - 难以用软件有效的实现算法

# AES的背景

- 1997年4月15日，NIST发起征集高级加密标准AES的活动，活动目的是确定一个非保密的、可以公开技术细节的、全球免费使用的分组密码算法，作为新的数据加密标准。
- 1997年9月12日，美国联邦登记处公布了正式征集AES候选算法的通告。作为进入AES候选过程的一个条件，开发者承诺放弃被选中算法的知识产权。
- 对AES的基本要求是：
  - 至少与三重DES一样安全
  - 比3DES快
  - 数据分组长度为128比特
  - 密钥长度为128/192/256比特
  - 软件和硬件平台上都适用

# AES的背景

- 1998年8月12日，在首届AES会议上指定了15个候选算法。
- 1999年3月22日第二次AES会议上，将候选名单减少为5 个，这5 个算法是RC6， Rijndael SERPENT，Twofish和MARS。
- 2000年4月13日，第三次AES会议上，对这5个候选算法的各种分析结果进行了讨论。
- 2000年10月2日，NIST宣布了获胜者—Rijndael算法，2001 年11 月出版了最终标准FIPS PUB197。
- NIST于2002年5月26日制定了新的高级加密标准（AES）规范。

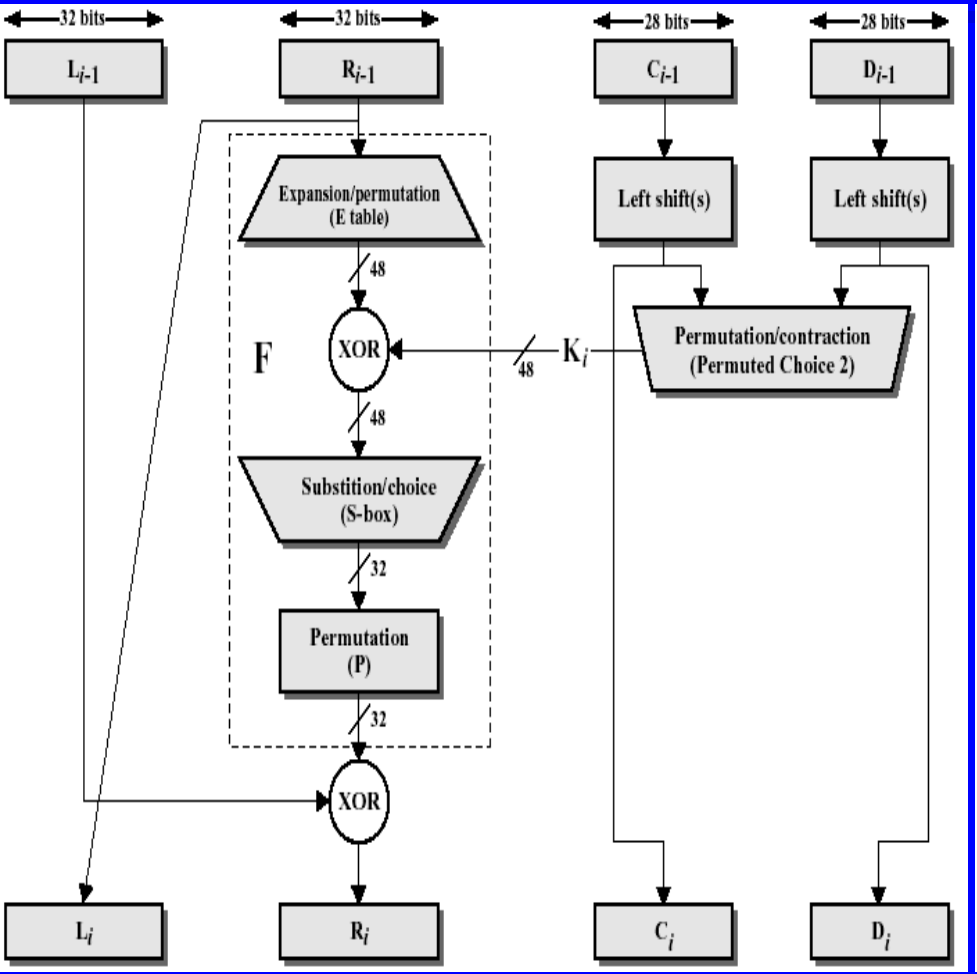
# Rijndael简介

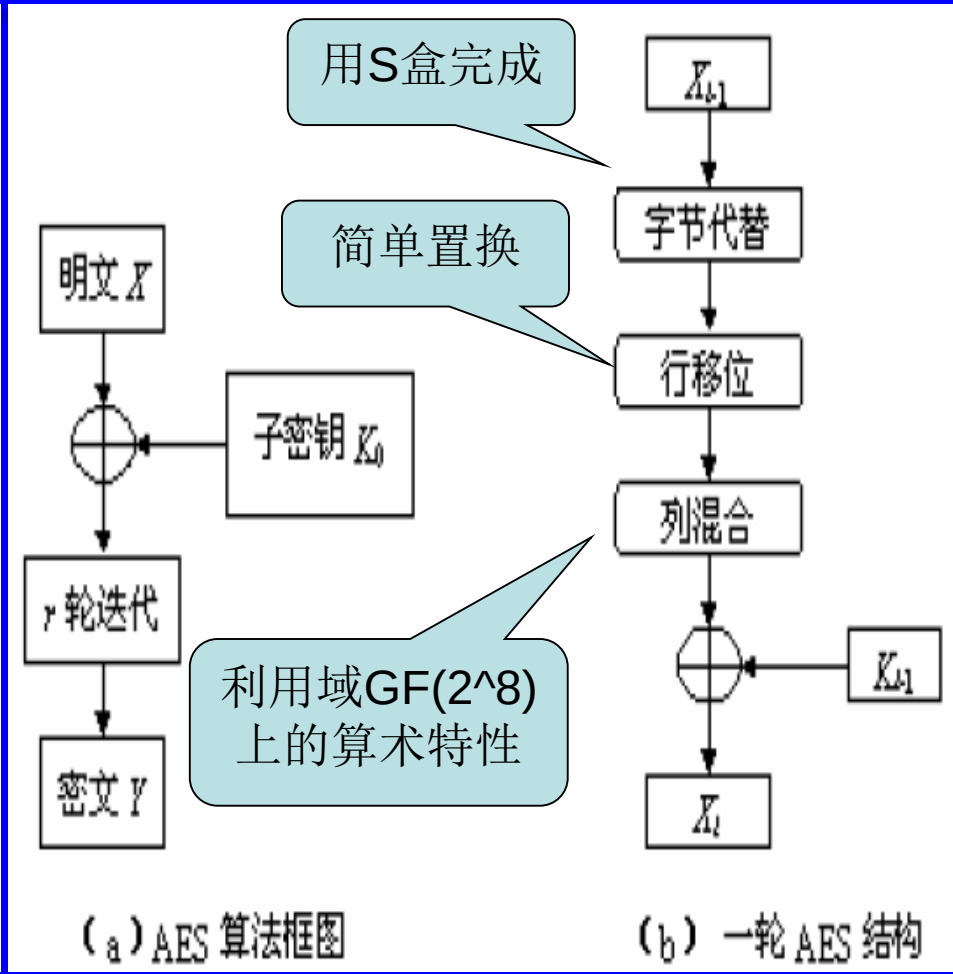
- 不属于Feistel结构
- 加密、解密相似但不对称
- 支持128位数据块大小
- 支持128/192/256(/32=Nk)密钥长度
- 有较好的数学理论作为基础
- 结构简单、速度快

# AES参数

| Key size (words/bytes/bits) | 4/16/128 | 6/24/192 | 8/32/256 |
|---|---|---|---|
| Plaintext block size (words/bytes/bits) | 4/16/128 | 4/16/128 | 4/16/128 |
| Number of rounds | 10 | 12 | 14 |
| Round key size (words/bytes/bits) | 4/16/128 | 4/16/128 | 4/16/128 |
| Expanded key size (words/bytes) | 44/176 | 52/208 | 60/240 |

# AES算法结构（SPN结构）

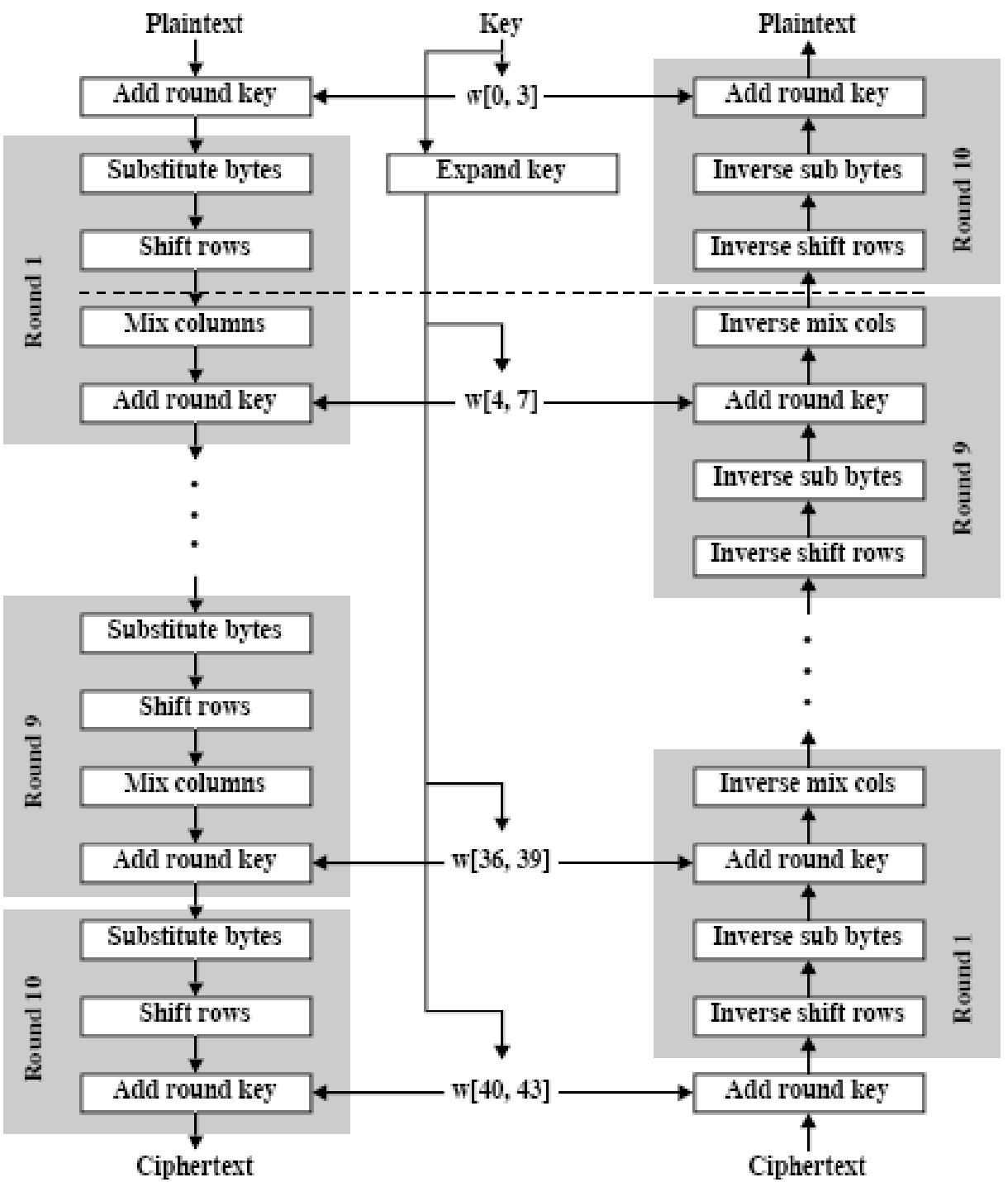


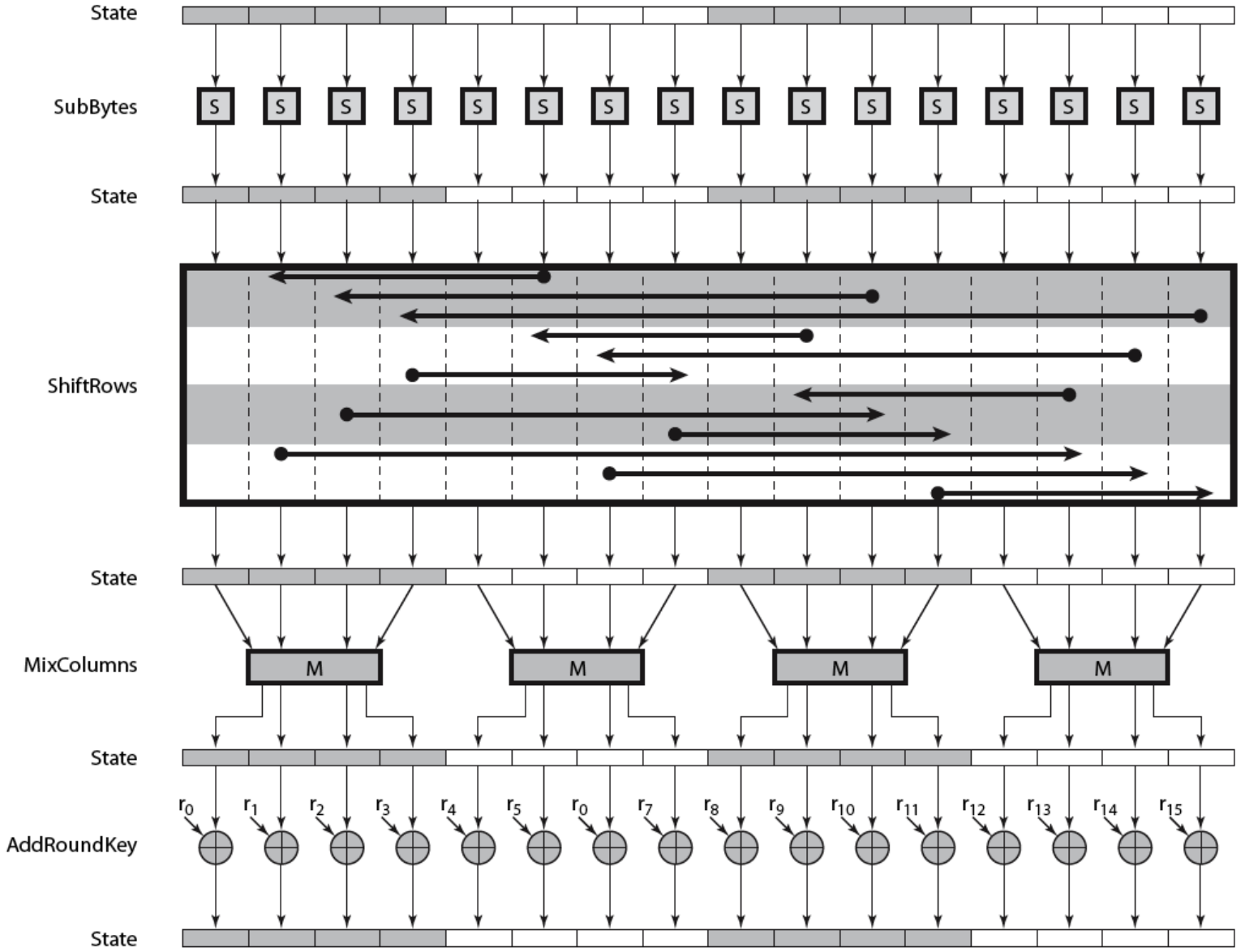

DES算法

AES算法

# AES-128加解密过程



# AES 一轮加密过程


State
SubBytes
S
State
ShiftRows
State
MixColumns
M
State
AddRoundKey
r0 r1 r2 r3 r4 r5 r0 r7 r8 r9 r10 r11 r12 r13 r14 r15
State

# AES算法描述

- 假设：State表示数据，以及每一轮的中间结果，RoundKey表示每一轮对应的子密钥。算法描述如下：

```
Cipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
   byte  state[4,Nb]

   state = in

   AddRoundKey(state, w[0, Nb-1])                  // See Sec. 5.1.4

   for round = 1 step 1 to Nr-1
      SubBytes(state)                              // See Sec. 5.1.1
      ShiftRows(state)                             // See Sec. 5.1.2
      MixColumns(state)                            // See Sec. 5.1.3
      AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
   end for

   SubBytes(state)
   ShiftRows(state)
   AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])

   out = state
end
```

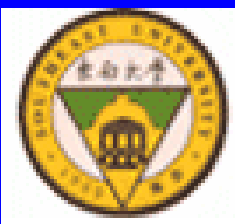
# 状态、密钥

- 状态/密钥的矩阵表示：以字节为单位的正方形矩阵

| | | | |
|---|---|---|---|
| $S_{00}$ | $S_{01}$ | $S_{02}$ | $S_{03}$ |
| $S_{04}$ | $S_{05}$ | $S_{06}$ | $S_{07}$ |
| $S_{08}$ | $S_{09}$ | $S_{10}$ | $S_{11}$ |
| $S_{12}$ | $S_{13}$ | $S_{14}$ | $S_{15}$ |

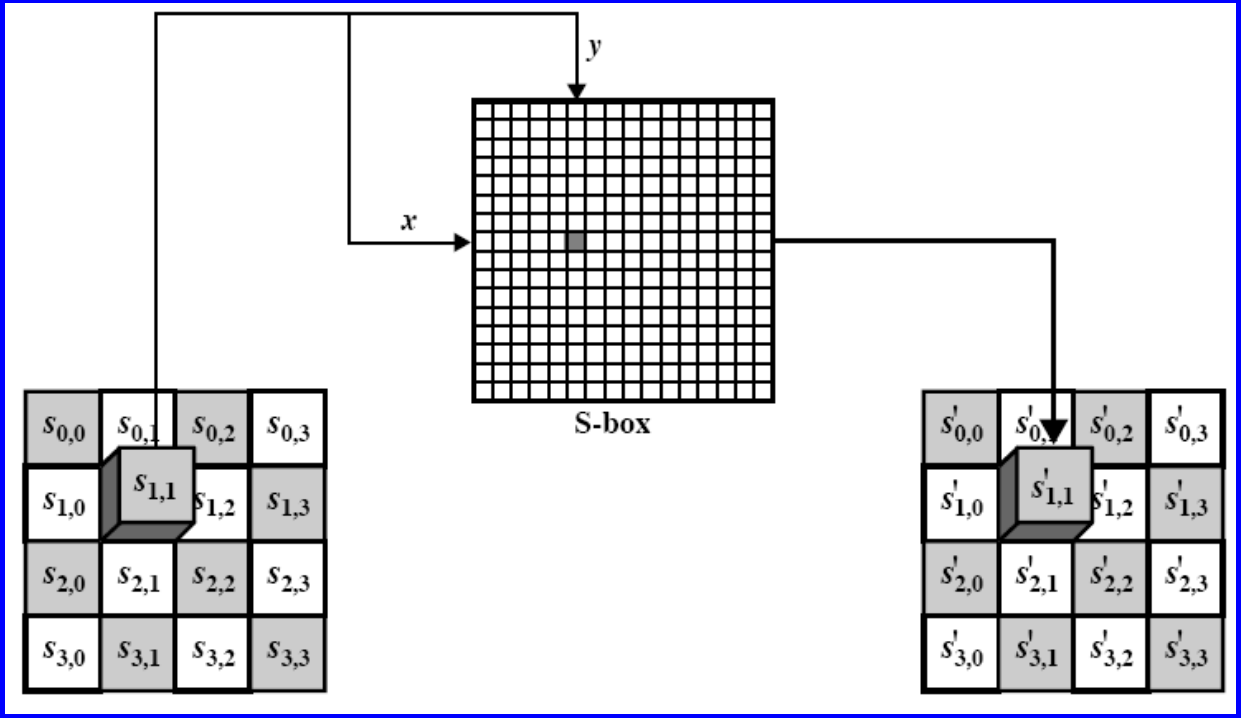
$\oplus$

| | | | |
|---|---|---|---|
| $k_{00}$ | $k_{01}$ | $k_{02}$ | $k_{03}$ |
| $k_{10}$ | $k_{11}$ | $k_{12}$ | $k_{13}$ |
| $k_{20}$ | $k_{21}$ | $k_{22}$ | $k_{23}$ |
| $k_{30}$ | $k_{31}$ | $k_{32}$ | $k_{33}$ |



# 字节代替(Substitute Bytes )变换

- 字节代替是一个非线性的字节代替，独立地在每个状态字节上进行运算。代替表(S-盒)是可逆的，是一个16×16的矩阵。



# AES S-Box

| | | y | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| x | 0 | 63 | 7C | 77 | 7B | F2 | 6B | 6F | C5 | 30 | 01 | 67 | 2B | FE | D7 | AB | 76 |
| | 1 | CA | 82 | C9 | 7D | FA | 59 | 47 | F0 | AD | D4 | A2 | AF | 9C | A4 | 72 | C0 |
| | 2 | B7 | FD | 93 | 26 | 36 | 3F | F7 | CC | 34 | A5 | E5 | F1 | 71 | D8 | 31 | 15 |
| | 3 | 04 | C7 | 23 | C3 | 18 | 96 | 05 | 9A | 07 | 12 | 80 | E2 | EB | 27 | B2 | 75 |
| | 4 | 09 | 83 | 2C | 1A | 1B | 6E | 5A | A0 | 52 | 3B | D6 | B3 | 29 | E3 | 2F | 84 |
| | 5 | 53 | D1 | 00 | ED | 20 | FC | B1 | 5B | 6A | CB | BE | 39 | 4A | 4C | 58 | CF |
| | 6 | D0 | EF | AA | FB | 43 | 4D | 33 | 85 | 45 | F9 | 02 | 7F | 50 | 3C | 9F | A8 |
| | 7 | 51 | A3 | 40 | 8F | 92 | 9D | 38 | F5 | BC | B6 | DA | 21 | 10 | FF | F3 | D2 |
| | 8 | CD | 0C | 13 | EC | 5F | 97 | 44 | 17 | C4 | A7 | 7E | 3D | 64 | 5D | 19 | 73 |
| | 9 | 60 | 81 | 4F | DC | 22 | 2A | 90 | 88 | 46 | EE | B8 | 14 | DE | 5E | 0B | DB |
| | A | E0 | 32 | 3A | 0A | 49 | 06 | 24 | 5C | C2 | D3 | AC | 62 | 91 | 95 | E4 | 79 |
| | B | E7 | C8 | 37 | 6D | 8D | D5 | 4E | A9 | 6C | 56 | F4 | EA | 65 | 7A | AE | 08 |
| | C | BA | 78 | 25 | 2E | 1C | A6 | B4 | C6 | E8 | DD | 74 | 1F | 4B | BD | 8B | 8A |
| | D | 70 | 3E | B5 | 66 | 48 | 03 | F6 | 0E | 61 | 35 | 57 | B9 | 86 | C1 | 1D | 9E |
| | E | E1 | F8 | 98 | 11 | 69 | D9 | 8E | 94 | 9B | 1E | 87 | E9 | CE | 55 | 28 | DF |
| | F | 8C | A1 | 89 | 0D | BF | E6 | 42 | 68 | 41 | 99 | 2D | 0F | B0 | 54 | BB | 16 |

# EXAMPLE

| | | | |
|---|---|---|---|
| EA | 04 | 65 | 85 |
| 83 | 45 | 5D | 96 |
| 5C | 33 | 98 | B0 |
| F0 | 2D | AD | C5 |

→

| | | | |
|---|---|---|---|
| 87 | F2 | 4D | 97 |
| EC | 6E | 4C | 90 |
| 4A | C3 | 46 | E7 |
| 8C | D8 | 95 | A6 |

# S-Box的构造方法

- 逐行按升序排列的字节值初始化S盒；
- 把S盒中的每个字节映射为它在有限域GF($2^8$)中的逆；
- 不可约多项式为：$m(x) = x^8 + x^4 + x^3 + x + 1$
- 把S盒中的每个字节记成($b_7$,$b_6$,$b_5$,$b_4$,$b_3$,$b_2$,$b_1$,$b_0$)。对S盒中每个字节的每位做如下的变换：

$$b_i' = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i$$

# 逆字节代替变换

$$b_i' = b_{(i+2) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus d_i$$

- 字节代换矩阵 X
- 逆字节代换矩阵 Y
- 常量c、d的矩阵表示方法为C和D
- 对某个8位的矢量B，字节代换代换结果B‘

B’＝XB⊕ C　B＝YB‘ ⊕ D

证明：B＝Y（XB⊕C）⊕ D＝ YXB⊕ YC⊕ D

# 字节代换和字节逆代换

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

字节代换

字节逆代换

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

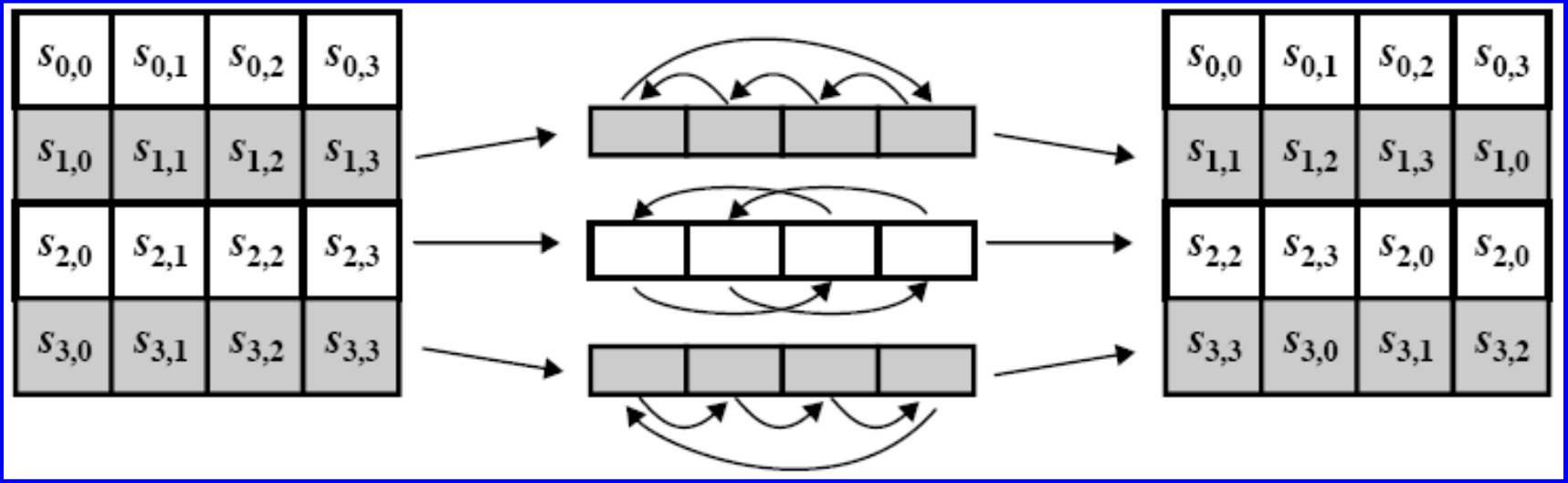
# 字节代替变换分析

- 能够防止已有的各种密码分析攻击；
- 输入位和输出位之间的相关性几乎没有；
- S盒必须是可逆的：

  即 逆S盒[S盒(a)]=a，但S盒(a)=逆S盒(a)不成立，S盒不是自逆的。

# 行移位(Shift Row)变换

- 简单的置换



# EXAMPLE

| | | | |
|---|---|---|---|
| 87 | F2 | 4D | 97 |
| EC | 6E | 4C | 90 |
| 4A | C3 | 46 | E7 |
| 8C | D8 | 95 | A6 |

→

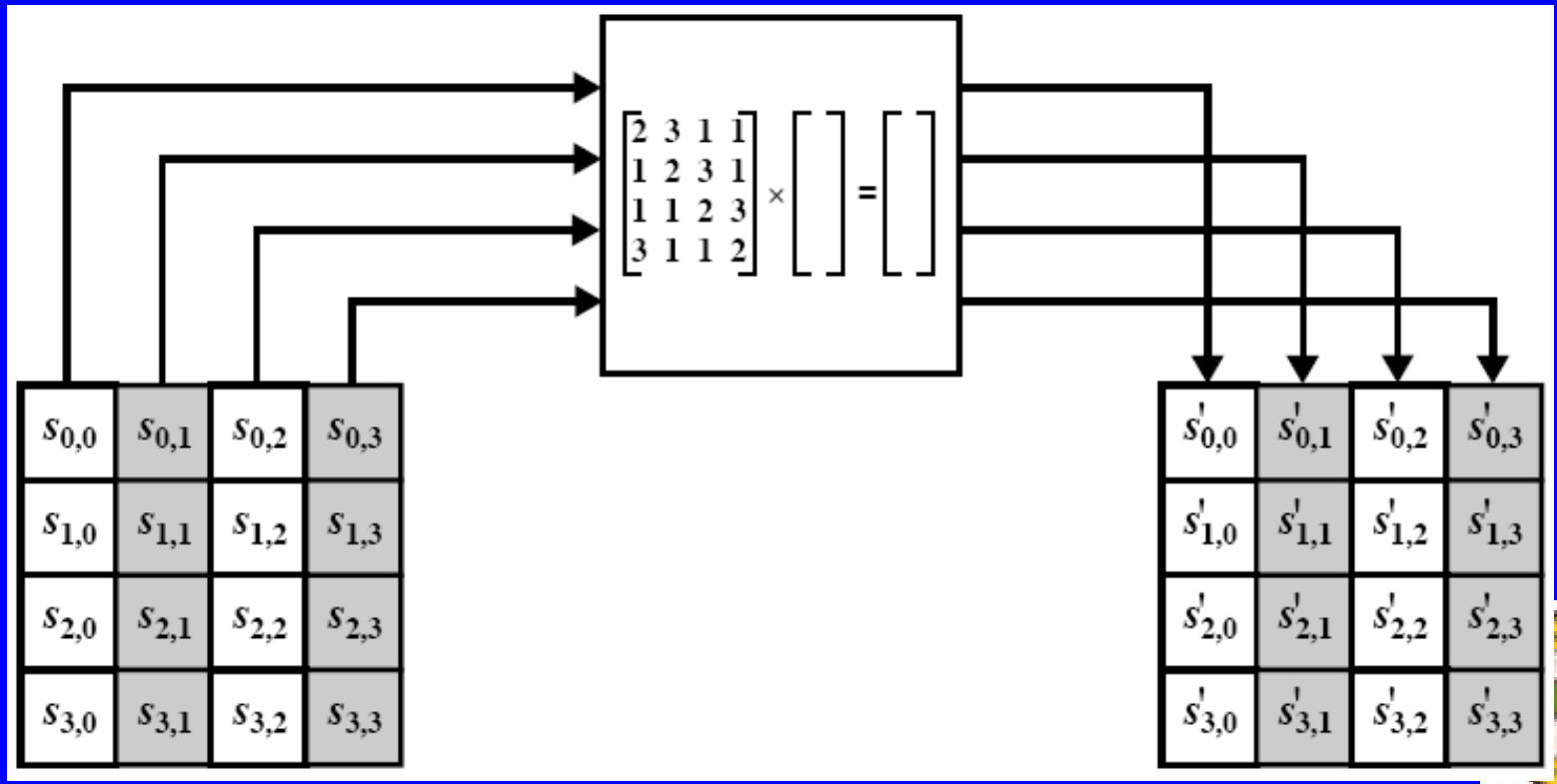
| | | | |
|---|---|---|---|
| 87 | F2 | 4D | 97 |
| 6E | 4C | 90 | EC |
| 46 | E7 | 4A | C3 |
| A6 | 8C | D8 | 95 |

# 行移位变换分析

- 行移位就是将某个字节从一列移到另一列，其线性距离是4字节的倍数。
- 确保某列中的四字节被扩展到了4个不同的列。

# 列混淆Mix Column变换

- 代替操作，将状态的列看作有限域GF($2^8$)上的4维向量并与有限域GF($2^8$)上的一个固定可逆方阵A相乘。



# 列混淆的矩阵表示

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

$$\begin{aligned} s'_{0,j} &= (2 \bullet s_{0,j}) \oplus (3 \bullet s_{1,j}) \oplus s_{2,j} \oplus s_{3,j} \\ s'_{1,j} &= s_{0,j} \oplus (2 \bullet s_{1,j}) \oplus (3 \bullet s_{2,j}) \oplus s_{3,j} \\ s'_{2,j} &= s_{0,j} \oplus s_{1,j} \oplus (2 \bullet s_{2,j}) \oplus (3 \bullet s_{3,j}) \\ s'_{3,j} &= (3 \bullet s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2 \bullet s_{3,j}) \end{aligned}$$

# 列混淆计算举例

| | | | |
|---|---|---|---|
| 87 | F2 | 4D | 97 |
| 6E | 4C | 90 | EC |
| 46 | E7 | 4A | C3 |
| A6 | 8C | D8 | 95 |

→

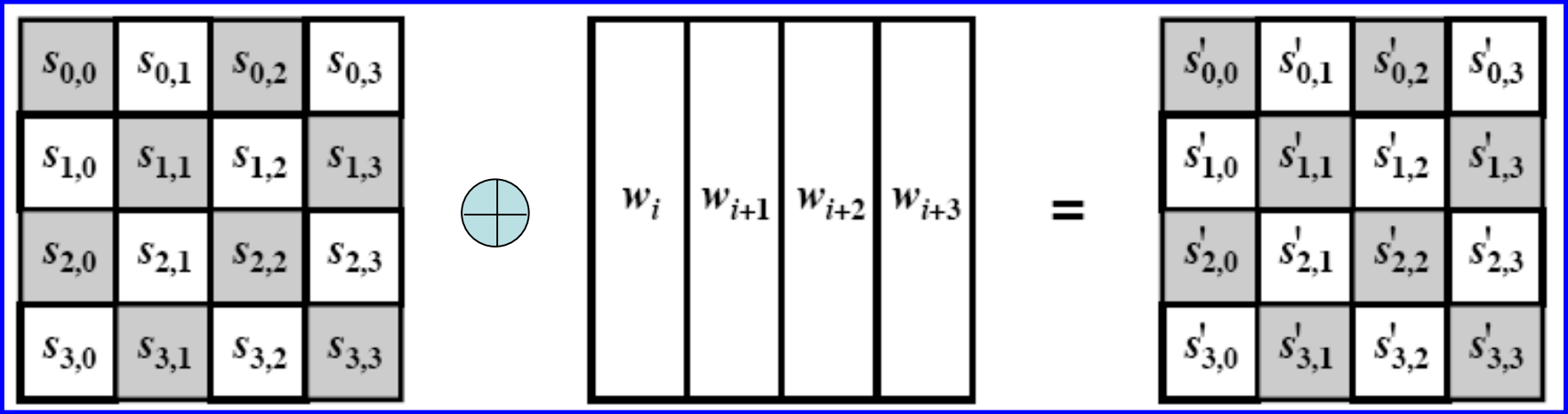
| | | | |
|---|---|---|---|
| 47 | 40 | A3 | 4C |
| 37 | D4 | 70 | 9F |
| 94 | E4 | 3A | 42 |
| ED | A5 | A6 | BC |

# 列混淆Mix Column变换评价

- 矩阵系数的选择是基于码字间有最大距离的线性编码。
- 每列的所有字节中有良好的混淆性。
- 经过几轮列混淆变换和行移位变换后，所有的输出位均与所有的输入位相关。

# 轮密钥加（Add Round Key）

- 一个简单地按位异或的操作



# AES的密钥调度

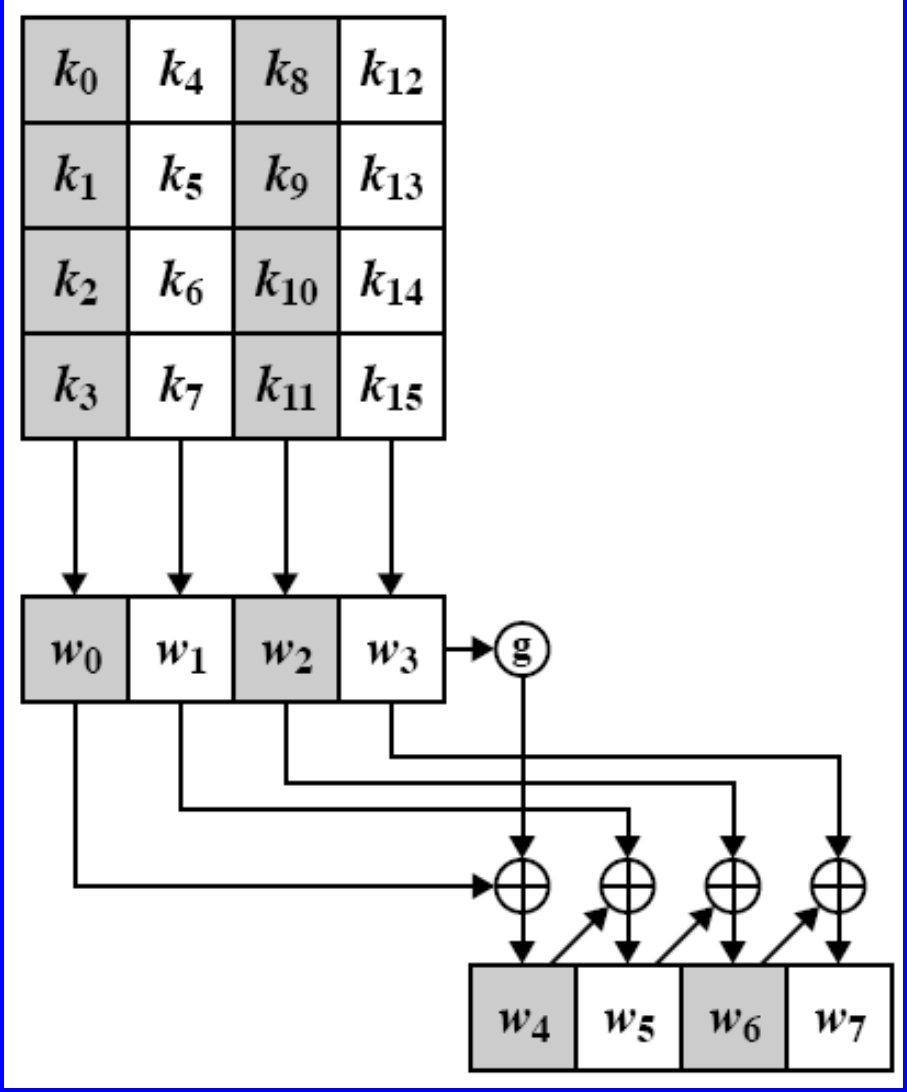
- 轮密钥是通过密钥调度算法从密钥中产生，包括两个组成部分：密钥扩展和轮密钥选取。
- 基本原理如下：
  - 所有轮密钥比特的总数等于分组长度乘轮数加1个分组。（如128比特的分组长度和10轮迭代，共需要1408比特的密钥）。
  - 将密码密钥扩展成一个扩展密钥。
  - 轮密钥按下述方式从扩展密钥中选取：第一个轮密钥由开始4个字组成，第二个轮密钥由接下来的4个字组成，如此继续下去。

# 密钥扩展的伪代码

- 密钥长度 128

```
KeyExpansion (byte key[16], word w[44])
{
    word temp
    for (i = 0; i < 4; i++)      w[i] = (key[4*i], key[4*i + 1], key[4*i + 2], key[4*i + 3]);
    for (i = 4; i < 44; i++)
    {
      temp = w[i − 1];
      if (i mod 4 = 0)      temp = SubWord (RotWord (temp)) ⊕ Rcon[i/4];
      w[i] = w[i−4] ⊕ temp
    }
}
```

# AES-128的密钥扩展

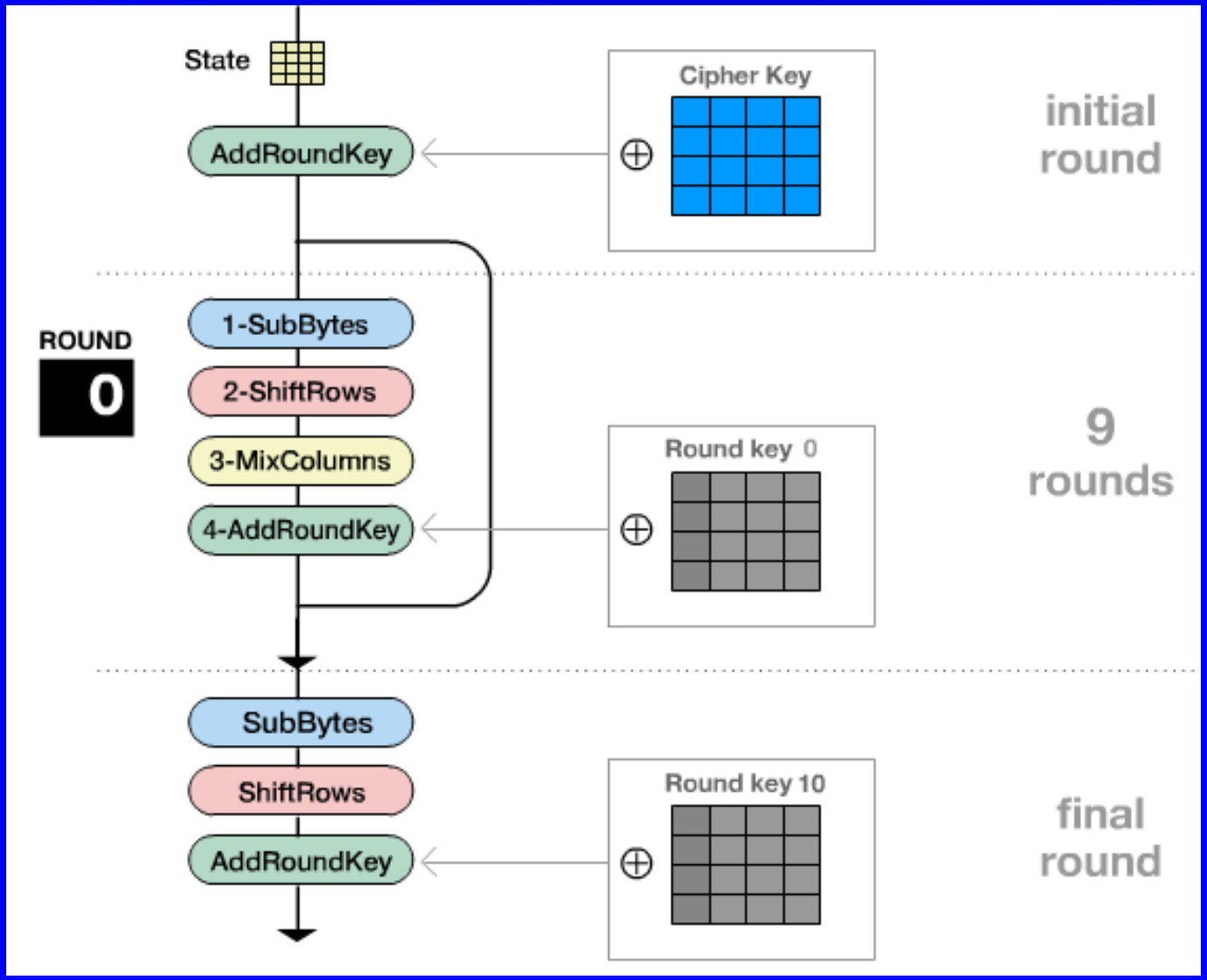


# G变换

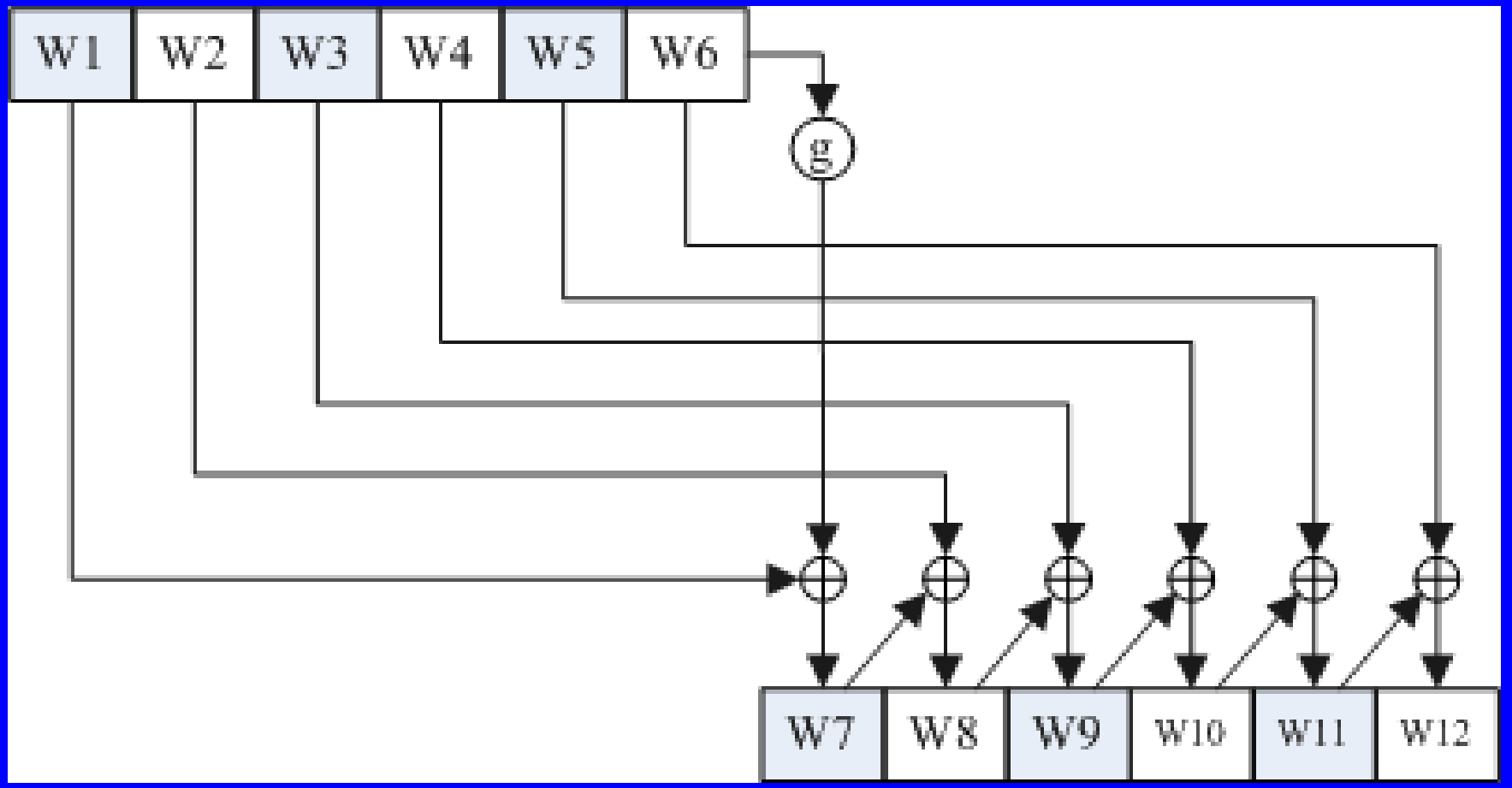
- G变换:
  - Rotword执行一字节循环左移[b0,b1,b2,b3] →[b1,b2,b3,b0]
  - Subword执行使用S-盒实行字节代换
  - 前两步的结果与轮常数Rcon[j]相异或

| j | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| RC[j] | 01 | 02 | 04 | 08 | 10 | 20 | 40 | 80 | 1B | 36 |

# AES算法的演示


State
AddRoundKey
Cipher Key
initial round
ROUND
0
1-SubBytes
2-ShiftRows
3-MixColumns
4-AddRoundKey
Round key 0
9 rounds
SubBytes
ShiftRows
AddRoundKey
Round key 10
final round

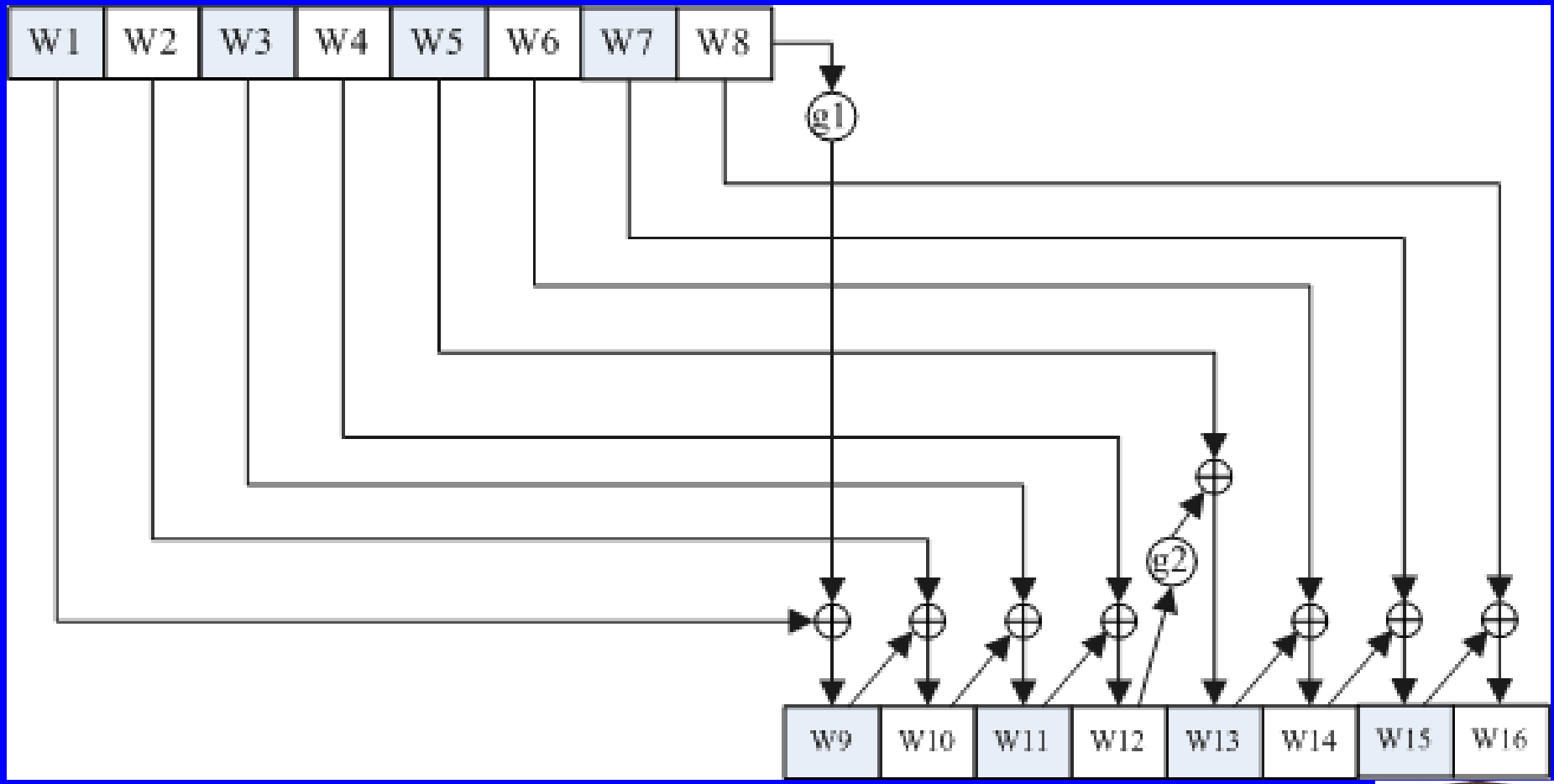
# AES-192的密钥扩展



# AES-192的密钥扩展

```
KeyExpansion(byte Key[16] Word W[52]    //52=Nb*(Nr+1)
  {
    for(i=0;i<6;i++)
      W[i]=(Key[4*i],Key[4*i+1], Key[4*i+2] Key[4*i+3]);
    for(i=6;i<52;i++)
      {
        temp = W[i-1];
        if (i%6 = = 0)
          temp = SubByte(RotByte(temp))^Rcon[i/6];
        W[i] = W[i-6]^temp;
      }
  }
```

# AES-256的密钥扩展



# AES-256的密钥扩展

```
KeyExpansion(byte Key[32] Word W[60]     //60=Nb*(Nr+1)
  {
    for(i=0;i<8;i++)
      W[i]=(Key[4*i],Key[4*i+1], Key[4*i+2] Key[4*i+3]);
    for(i=8;i<60;i++)
      {
        temp = W[i-1];
        if (i%8 = = 0)
          temp = SubByte(RotByte(temp))^Rcon[i/8];
        if (i%8 = = 4)
          temp = SubByte(temp);
        W[i] = W[i-8]^temp;
      }
  }
```

# AES算法的安全性

- 没有发现弱密钥或补密钥
- 能有效抵抗目前已知的攻击算法
  - 线性攻击
  - 差分攻击

# 问 题

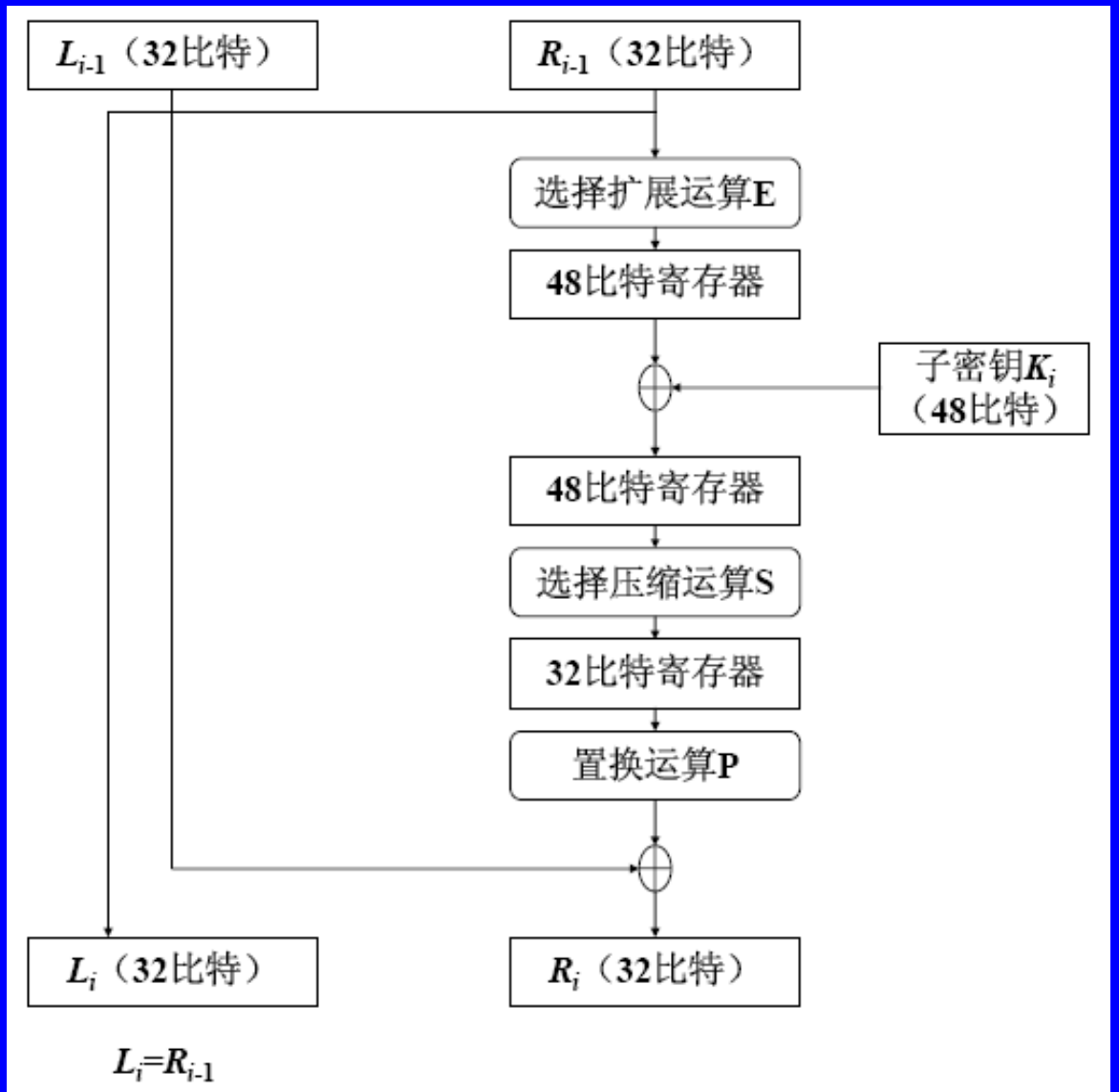
1、AES算法中，为什么AES-192只需要1次g运算，但AES-256却需要2次g运算

东南大学网络空间安全学院

密码学与安全协议

**谢谢！**

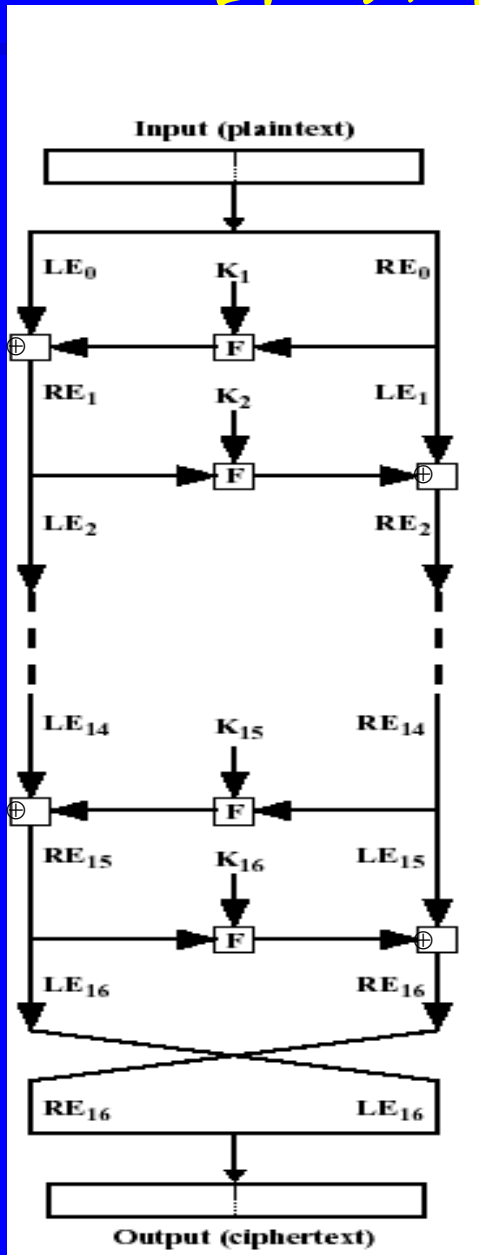
# DES的一轮迭代

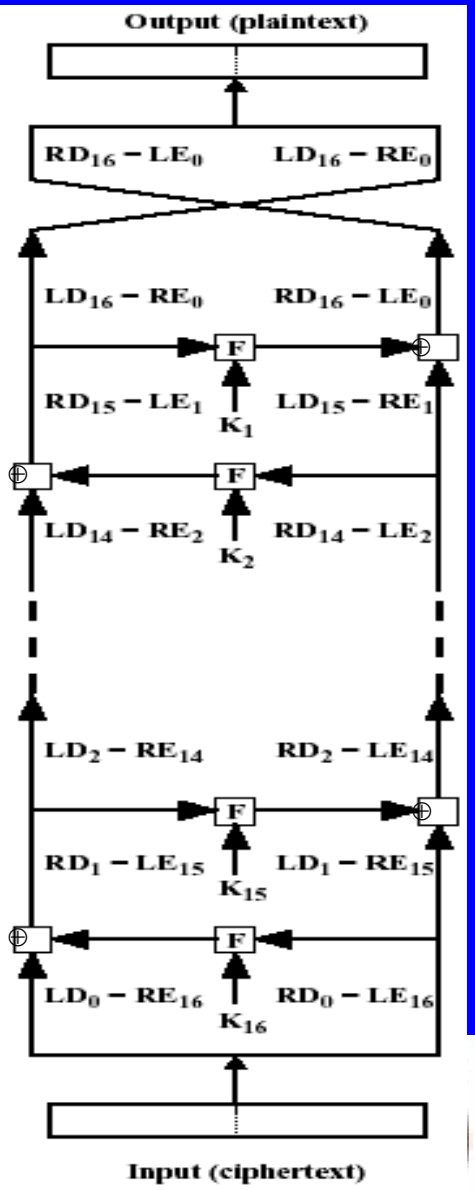

$L_{i-1}$（32比特）
$R_{i-1}$（32比特）
选择扩展运算E
48比特寄存器
子密钥$K_i$
（48比特）
48比特寄存器
选择压缩运算S
32比特寄存器
置换运算P
$L_i$（32比特）
$R_i$（32比特）
$L_i=R_{i-1}$

# Feistel加密与解密

- 加密: $L_i = R_{i-1}$; $R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$
- 解密: $R_{i-1} = L_i$

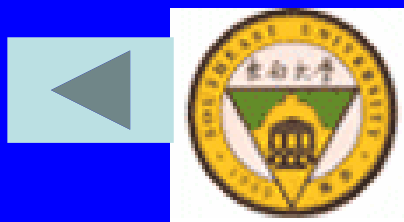
$$L_{i-1} = R_i \oplus F(R_{i-1}, K_i)$$
$$= R_i \oplus F(L_i, K_i)$$



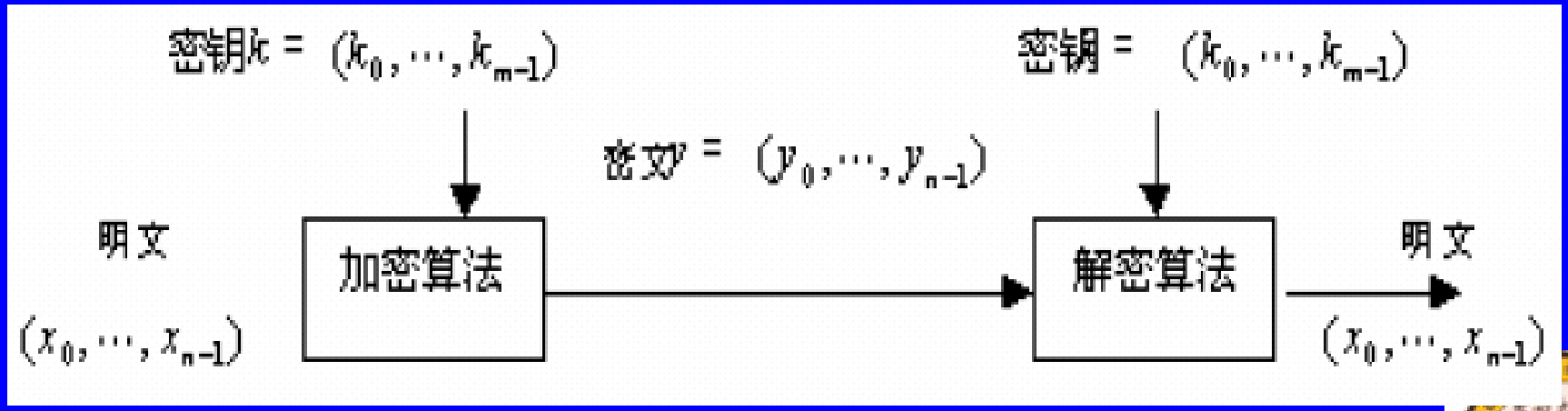
# 什么是一个好的加密算法

- 密码算法是固定的；
- 明文和密文是一个一一映射关系：单射，即
  if $x_1 != x_2$，则$E_k(x_1) != E_k(x_2)$
- 通常情况是：明文非常有序，在此条件下，我们期望得到什么样的密文？
- 密文的所有统计特征都是独立于所用密钥，即理想加密算法



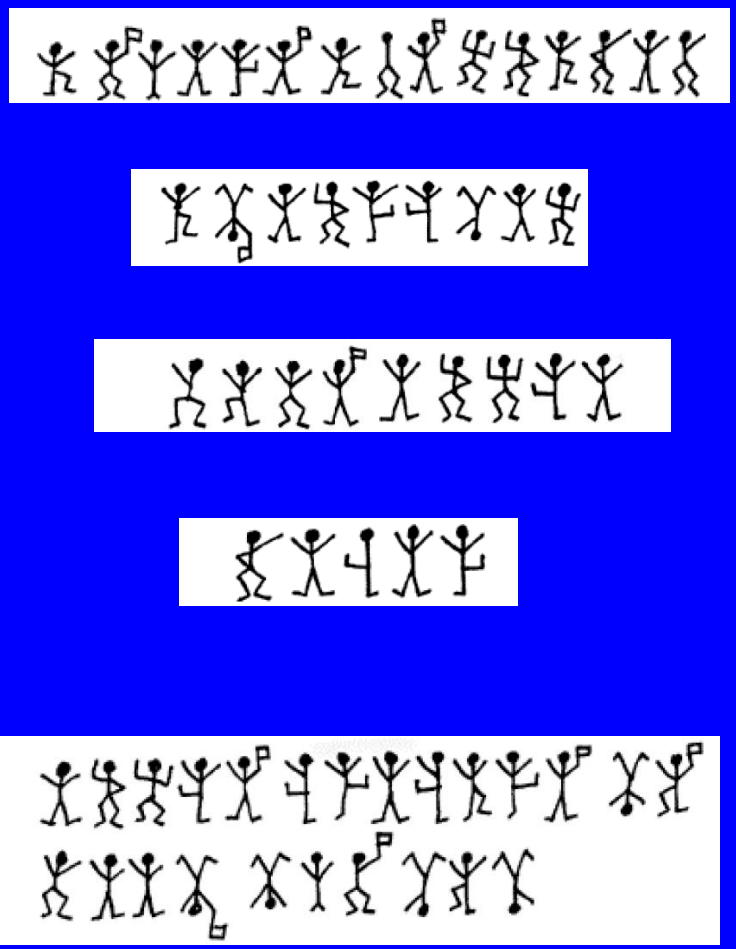
# 分组密码的模型

- 分组密码，就是一个明文分组被当作一个整体来产生一个等长（通常）的密文分组的密码，通常使用的是**128**位分组大小。
- 分组密码的实质是，设计一种算法，能在密钥控制下，把**n**比特明文简单而又迅速地置换成唯一**n**比特密文，并且这种变换是可逆的（解密）。



# 明文转换成密文的模型

- 代换案例：福尔摩斯密码



E

---E--E-

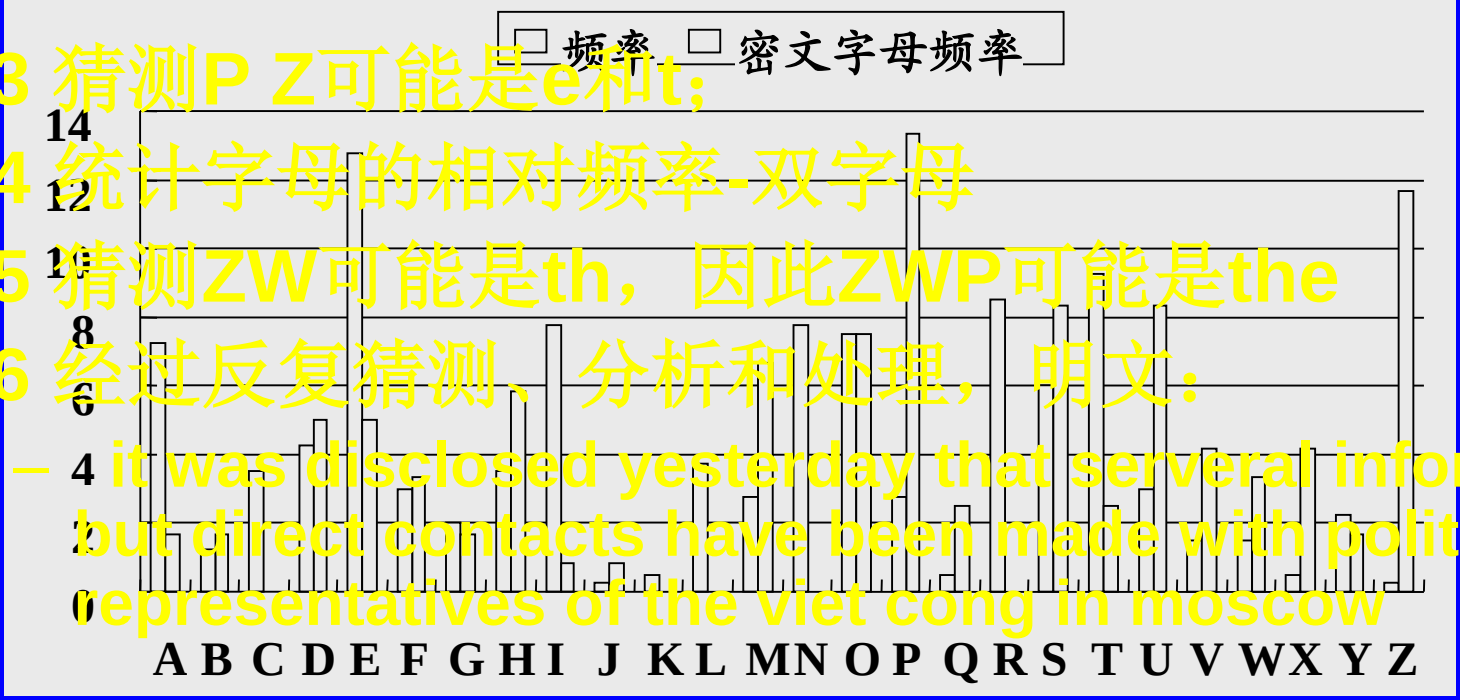
?

NEVER

N-V-R

# 基于语言统计规律的破译

- 1 密文：
  - UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESX UDBMETSXAIZVUEPHZHMDZSHZOWSFPAPPDTS VPQUZWYMXUZUHSXEPYEPOPDZSZUFPOMBZW PFUPZHMDJUDTMOHMQ
- 2 统计字母的相对频率；
- 3 猜测P Z可能是e和t；
- 4 统计字母的相对频率-双字母
- 5 猜测ZW可能是th，因此ZWP可能是the
- 6 经过反复猜测、分析和处理，明文：
  - it was disclosed yesterday that serveral informal but direct contacts have been made with political representatives of the viet cong in moscow

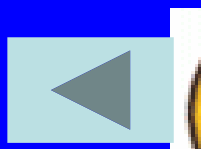


# 置换密码

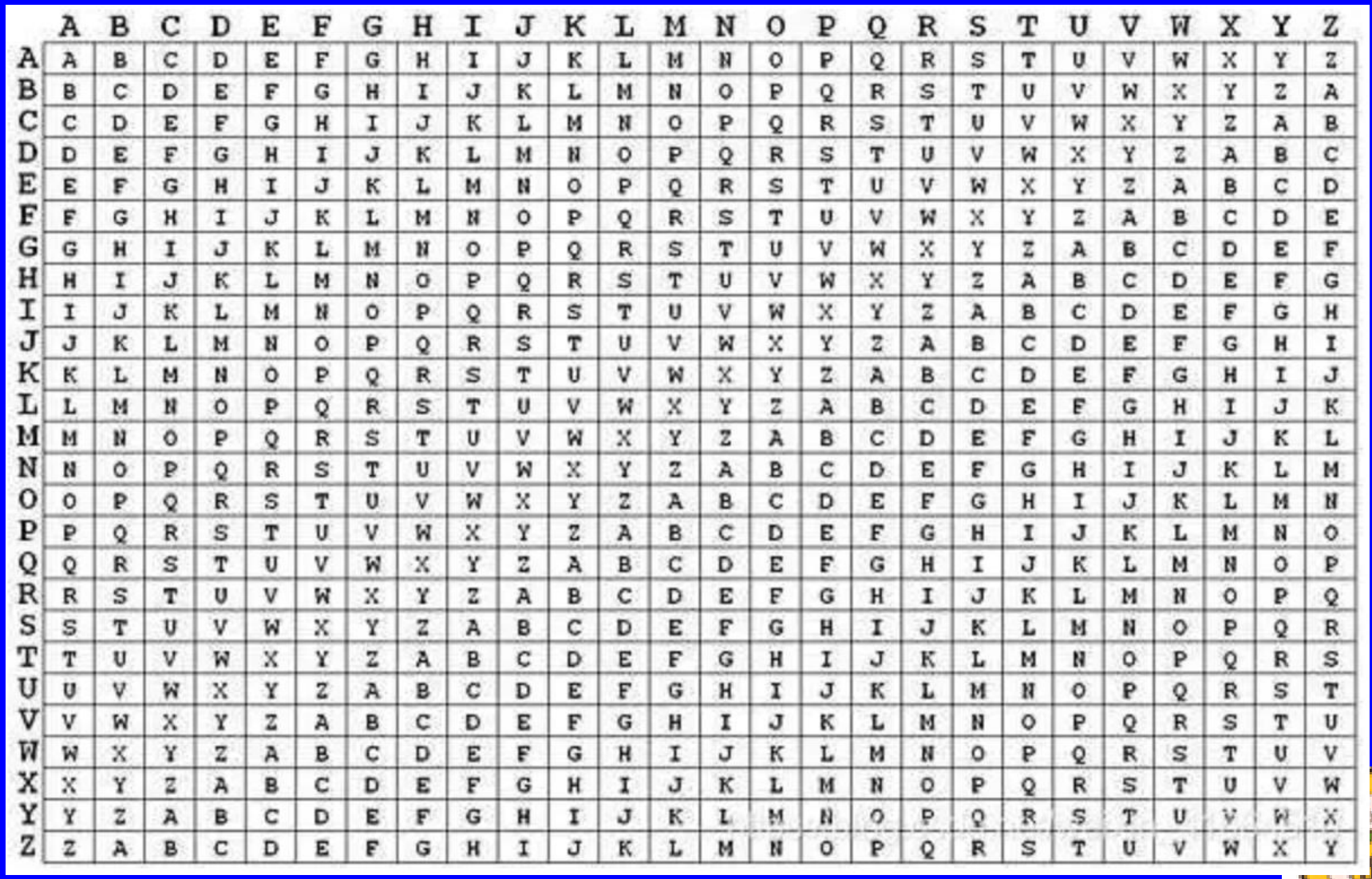
- 换位密码把明文按行写入,按列读出
- 更复杂的方法：密钥包含读出顺序
- **key:　4 3 1 2 5 6 7**
- **plaintext:**

  **a t t a c k p**
  **o s t p o n e**
  **d u n t I l t**
  **w o a m x y z**

- **cipher text: TTNA APTM TSUO AODW COIX KNLZ PETZ**
- 完全保留字符的统计信息
- 使用多轮加密可提高安全性



# 维吉尼亚（Vigenere)密码



| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| B | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A |
| C | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B |
| D | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C |
| E | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D |
| F | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E |
| G | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F |
| H | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G |
| I | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H |
| J | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I |
| K | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J |
| L | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K |
| M | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L |
| N | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M |
| O | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N |
| P | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| Q | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P |
| R | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q |
| S | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R |
| T | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S |
| U | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T |
| V | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U |
| W | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V |
| X | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W |
| Y | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X |
| Z | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y |