信息安全 微课堂

信息安全概论

Introduction to Information Security

主讲教师：赵 彦 锋
院　　系：信息工程学院 软件工程系
邮　　箱：c_zhaoyf@163.com

2021 年 03 月

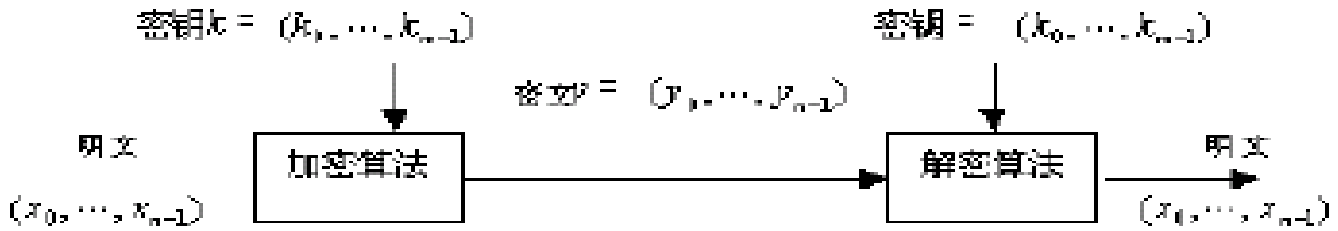
第三周

第三章 对称密码体制

- 分组密码原理
- DES
- 分组密码工作模式
- 流密码

- 对称密码体制根据加密方式不同分为：分组密码和流密码。
- 分组密码设计原理

- 分组密码是将明文消息编码表示后的数字（简称明文数字）序列，划分成长度为n的组（可看成长度为n的矢量），每组分别在密钥的控制下变换成等长的输出数字（简称密文数字）序列，



分组密码模型

安全性和软/硬件实现要求

- 分组长度应足够大
- 密钥空间应足够大
- 由密钥确定的算法要足够复杂
- 软件实现的要求
- 硬件实现的要求

Shannon利用信息论方法研究加密问题，提出了完善加密的概念，为密码学奠定了理论基础，使密码学成为一门科学。

Shannon建议两种对付统计分析方法：混乱和扩散

Shannon称之为理想密码系统中，密文的所有统计特性都与所使用的密钥独立

- 混乱(confusion)：使得密文的统计特性与密钥的取值之间的关系尽量复杂
- 扩散（Diffusion):明文的统计结构被扩散消失到密文的长程统计特性,使得明文和密文之间的统计关系尽量复杂

分组密码实现设计原则

- 软件实现的要求：使用子块和简单的运算。密码运算在子块上进行，要求子块的长度能自然地适应软件编程，如8、16、32比特等。应尽量避免按比特置换，在子块上所进行的密码运算尽量采用易于软件实现的运算。最好是用处理器的基本运算，如加法、乘法、移位等。
- 硬件实现的要求：加密和解密的相似性，即加密和解密过程的不同应仅仅在密钥使用方式上，以便采用同样的器件来实现加密和解密，以节省费用和体积。尽量采用标准的组件结构，以便能适应于在超大规模集成电路中实现。

分组密码的一般结构---Feistel结构

- 由于上世纪60年代计算机得到了迅猛的发展，大量的数据资料被集中储存在大型计算机数据库中并在计算机通信网中进行传输，其中有些通信具有高度的机密性，有些数据具有极为重要的价值，因此对计算机通信及计算机数据进行保护的需求日益增长。
- 针对这种情况，有人提出了两种对数据进行保护的方法，一种方法是对数据进行物理保护，即把重要的数据存放到安全的地方，如银行的地下室中；另一种方法是对数据进行密码保护。
- 20世纪70年代，Horst Feistel博士建立了DES（数据加密标准）的前身，在IBM公司Watson研究实验室推出了所谓Feistel密码的密码系列。

分组密码的一般结构---Feistel结构

- 乘积密码（Product Cipher）就是以某种方式连续执行两个或多个简单密码（如替代、置换），以使得所得到的最后结果或乘积比其任意一个组成密码都更强的分组密码。
- Shannon提出交替使用混淆和扩散进行乘积密码。
- 基于Shannon 理论，Feistel建议交替地使用代换和置换。

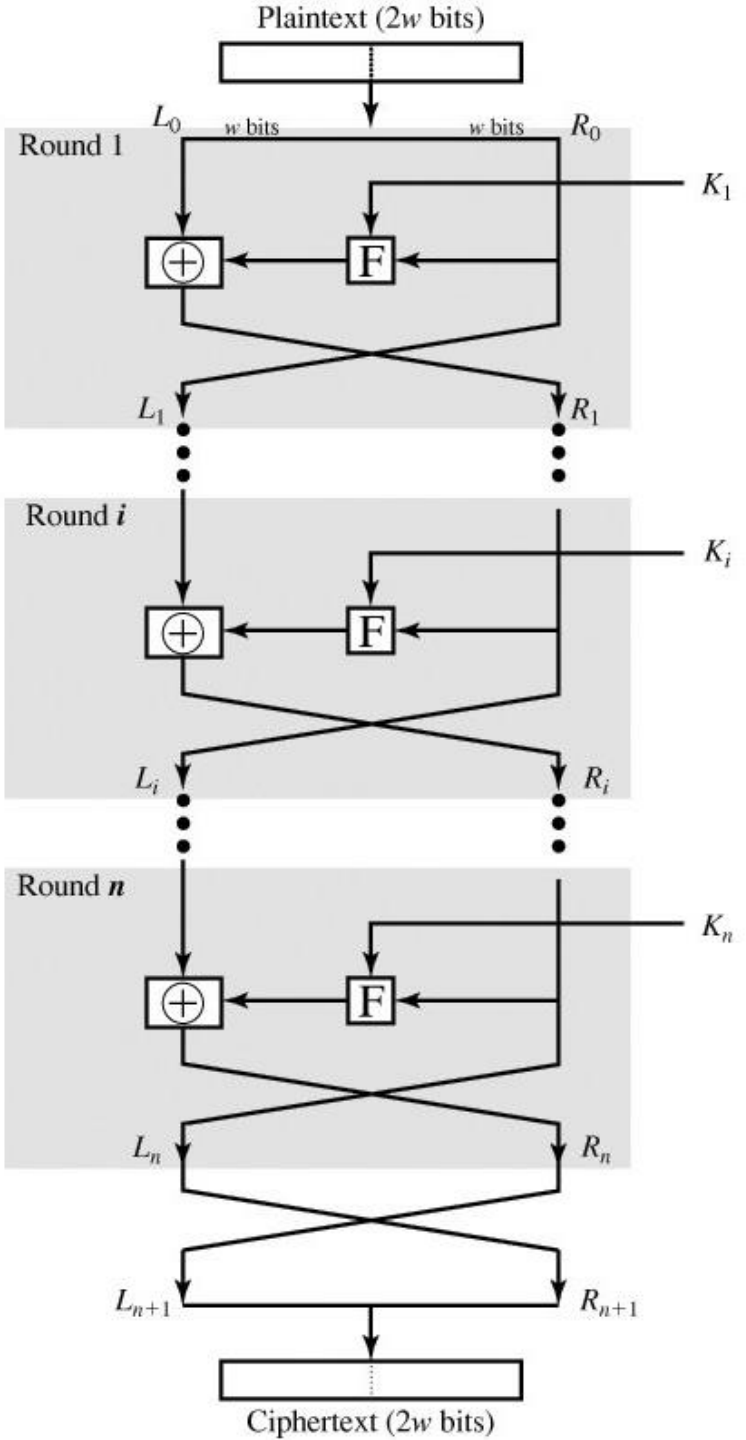
分组密码的一般结构---Feistel结构

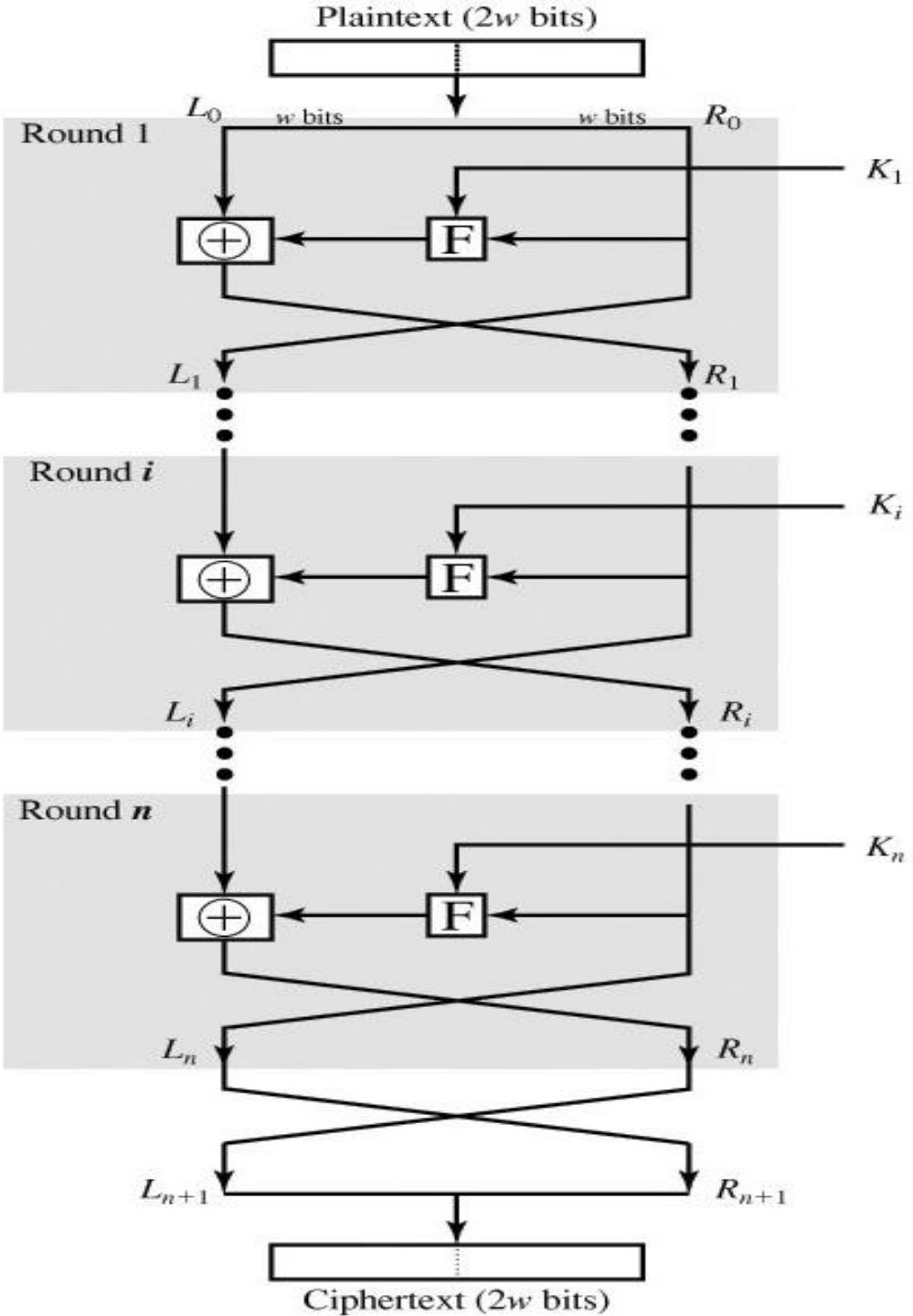
Feistel网络结构

将输入分组分成左右两部分。

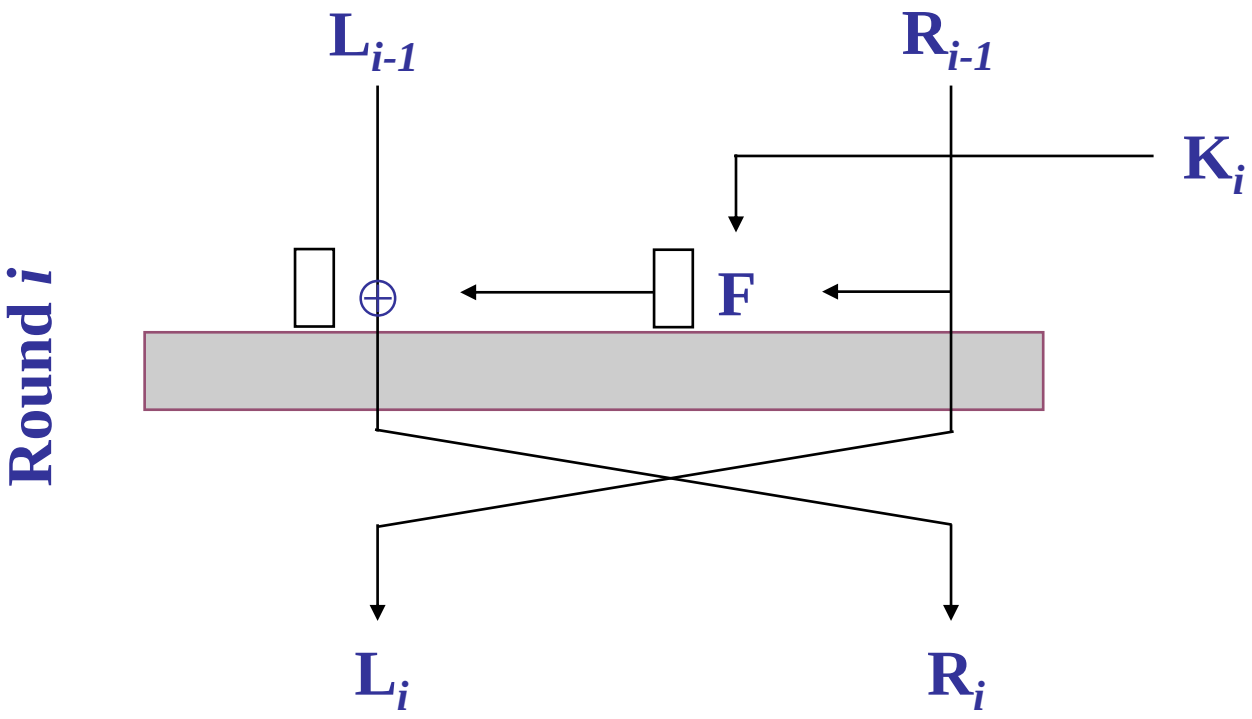
以右半部数据和子密钥作为参数，对左半部数据实施代换操作。

将两部分进行互换，完成置换操作。




Plaintext (2w bits)
L0
w bits
w bits
R0
Round 1
K1
F
L1
R1
Round i
Ki
F
Li
Ri
Round n
Kn
F
Ln
Rn
Ln+1
Rn+1
Ciphertext (2w bits)

Feistel结构每一轮的加密



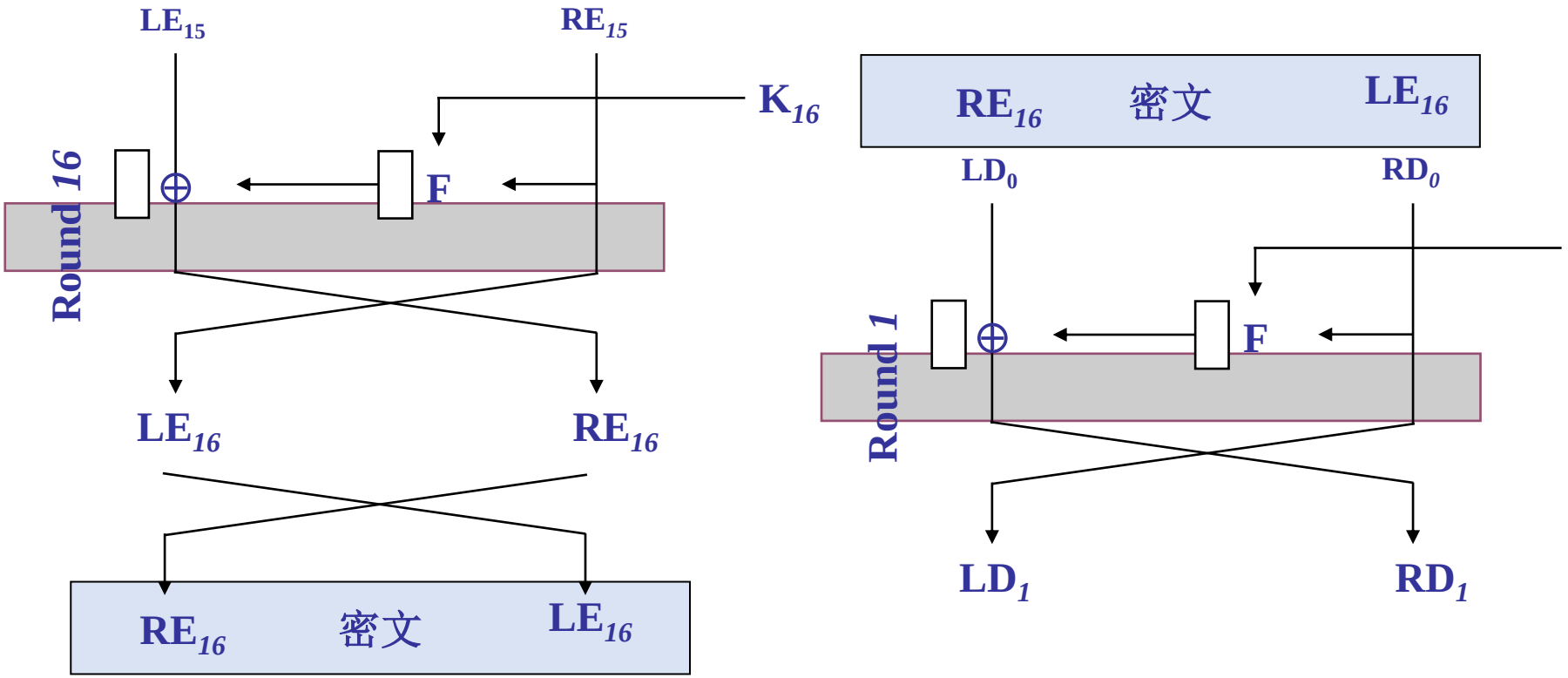
$$LE_i = RE_{i-1}$$

$$RE_i = LE_{i-1} \oplus F(RE_{i-1}, K_i)$$

Feistel结构解密

- Feistel解密算法：Feistel密码的解密过程本质上与加密过程一致。其规则如下：将密文作为算法的输入，但是逆序使用子密钥K_i。也就是说。第一轮使用K_n，第二轮使用K_{n-1}，直到最后一轮使用K_1。这是一个很好的特点，因为我们不需要实现两个算法：一个用做加密而另一个用做解密。

Feistel结构解密



$LD_1=RE_{15}$

$RD_1=LE_{15}$

Feistel结构

- 为清楚起见，我们用LE_i和RE_i表示加密过程的中间数据。而用LD_i和RD_i表示解密过程的中间数据。图中表明，每轮的解密过程中间值与加密过程中间值左右互换的结果是相同的。
- 我们来证明解密过程第一轮的输出等于加密过程第十六轮的输入(要互换左右两部分)。 也就是要证明$LD_1=RE_{15}$及$RD_1=LE_{15}$。
- 解密第一轮的输入是$RE_{16}||LE_{16}$。

Feistel结构

对于加密有：

$LE_{16} = RE_{15}$

$RE_{16} = LE_{15} \oplus F(RE_{15}, K_{16})$

对于解密则有：

$LD_1 = RD_0 = LE_{16} = RE_{15}$

$RD_1 = LD_0 \oplus F(RD_0, K_{16}) = RE_{16} \oplus F(RE_{15}, K_{16})$

$= (LE_{15} \oplus F(RE_{15}, K_{16})) \oplus F(RE_{15}, K_{16}) = LE_{15}$

异或的性质：

$[A \oplus B] \oplus C = A \oplus [B \oplus C]$

$D \oplus D = 0$

$E \oplus 0 = E$

故有$[E \oplus D] \oplus D = E \oplus [D \oplus D] = E \oplus 0 = E$

Feistel结构

一般地，每轮的加密算法：

$$LE_i = RE_{i-1}$$

$$RE_i = LE_{i-1} \oplus F(RE_{i-1}, K_i)$$

等价于：

$$RE_{i-1} = LE_i$$

$$LE_{i-1} = RE_i \oplus F(RE_{i-1}, K_i) = RE_i \oplus F(LE_i, K_i)$$

- 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

3.2 数据加密标准---背景

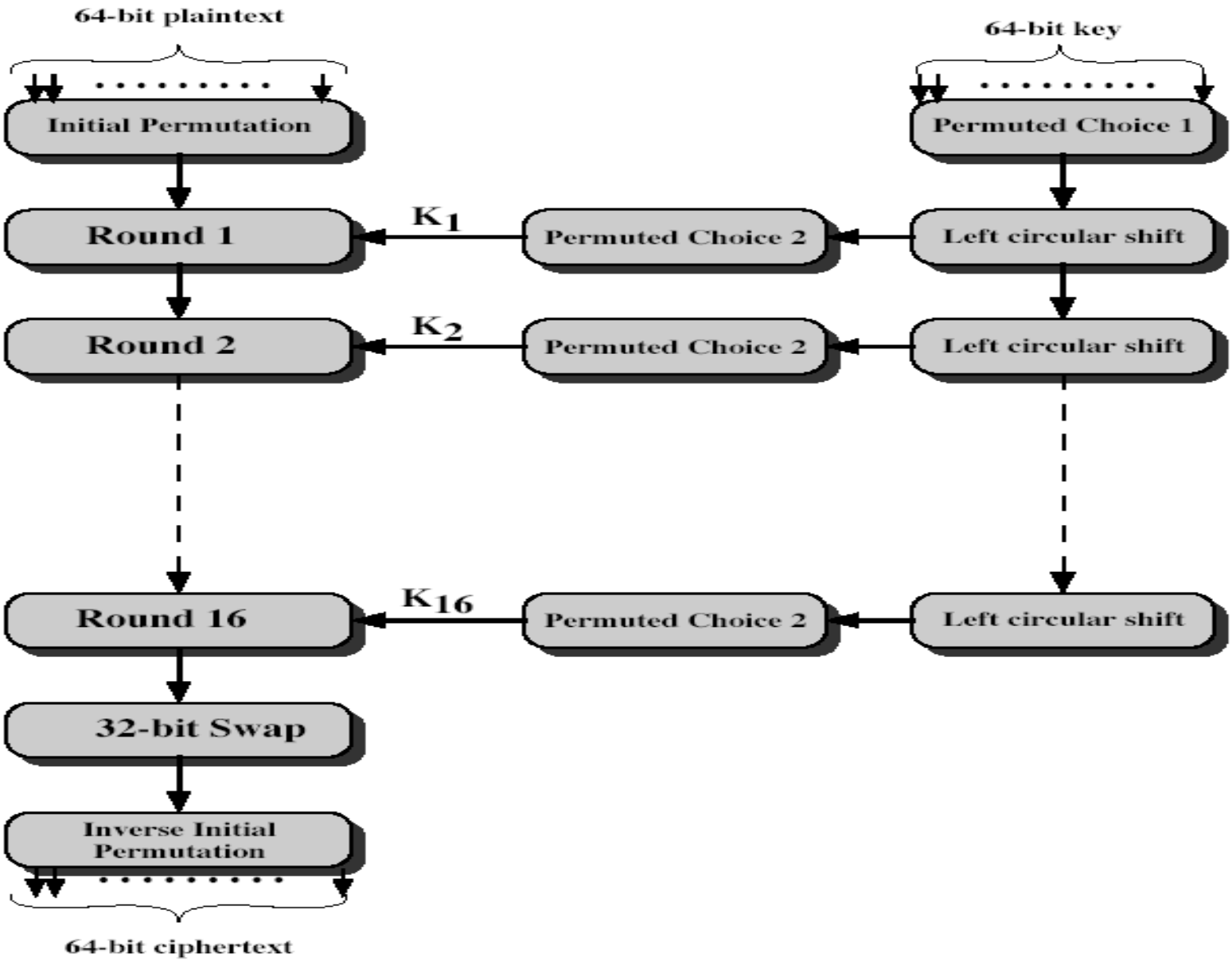
- 1974年8月27日, NBS开始第二次征集,IBM提交了算法LUCIFER，该算法由IBM的工程师在1971~1972年研制
- 1975年3月17日, NBS公开了全部细节
- 1976年,NBS指派了两个小组进行评价
- 1976年11月23日，采纳为联邦标准，批准用于非军事场合的各种政府机构
- 1977年1月15日,“数据加密标准”FIPS PUB 46发布

- 1979年，美国银行协会批准使用
- 1980年，美国国家标准局（ANSI）赞同DES作为私人使用的标准,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。

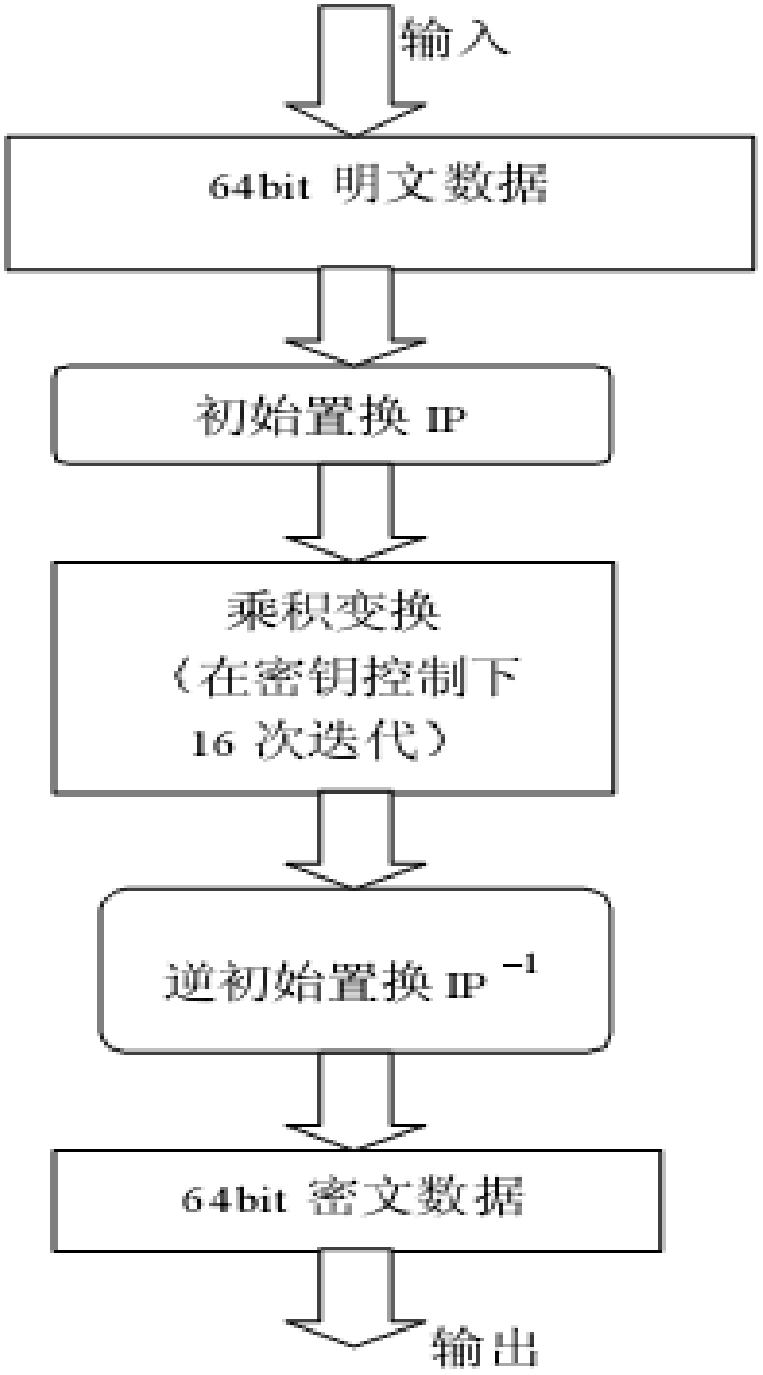
- DES是一种对称密码体制，它所使用的加密和解密密钥是相同的，是一种典型的按分组方式工作的密码。其基本思想是将二进制序列的明文分成每64bit一组，用长为64bit(56bit)的密钥对其进行16轮代换和置换加密，最后形成密文。

3.2 数据加密标准---加密一般描述



3.2 数据加密标准---加密一般描述

加密前，先将明文分成64bit的分组，然后将64bit二进制码输入到密码器中，密码器对输入的64位码首先进行初始置换，然后在64bit主密钥产生的16个子密钥控制下进行16 轮乘积变换，接着再进行逆置换就得到64位已加密的密文。



加密过程：

$$L_0R_0 \leftarrow IP(<64bit输入码>)$$

$$L_i \leftarrow R_{i-1} \qquad i=1,2,\cdots,16$$

$$R_i \leftarrow L_{i-1} \oplus f(R_{i-1},k_i) \qquad i=1,2,\cdots,16$$

$$<64bit密文> \leftarrow IP^{-1}(R_{16}L_{16})$$

解密过程：

$$R_{16}L_{16} \leftarrow IP(<64bit密文>)$$

$$R_{i-1} \leftarrow L_i \qquad i=16,15,\cdots,1$$

$$L_i \leftarrow R_{i-1} \oplus f(R_{i-1},k_i) \qquad i=16,15,\cdots,1$$

$$<64bit明文> \leftarrow IP^{-1}(R_0L_0)$$

***m*——明文消息（64bit)**
***C*——密文（64bit)**
***IP*——初始置换，*IP*$^{-1}$——*IP*的逆置换**
T_i ——第 i步迭代（i =1，2，…，16)
⊕——逐位异或运算
***F*——（多道运算的）函数**

3.2 DES加密---初始置换

- 初始置换IP：对输入的32bit消息m的一种置换，即将m的各位重排后输出。
- 逆置换IP^{-1}: 满足 $IP^{-1}.IP(m) = IP.IP^{-1}(m) = m$

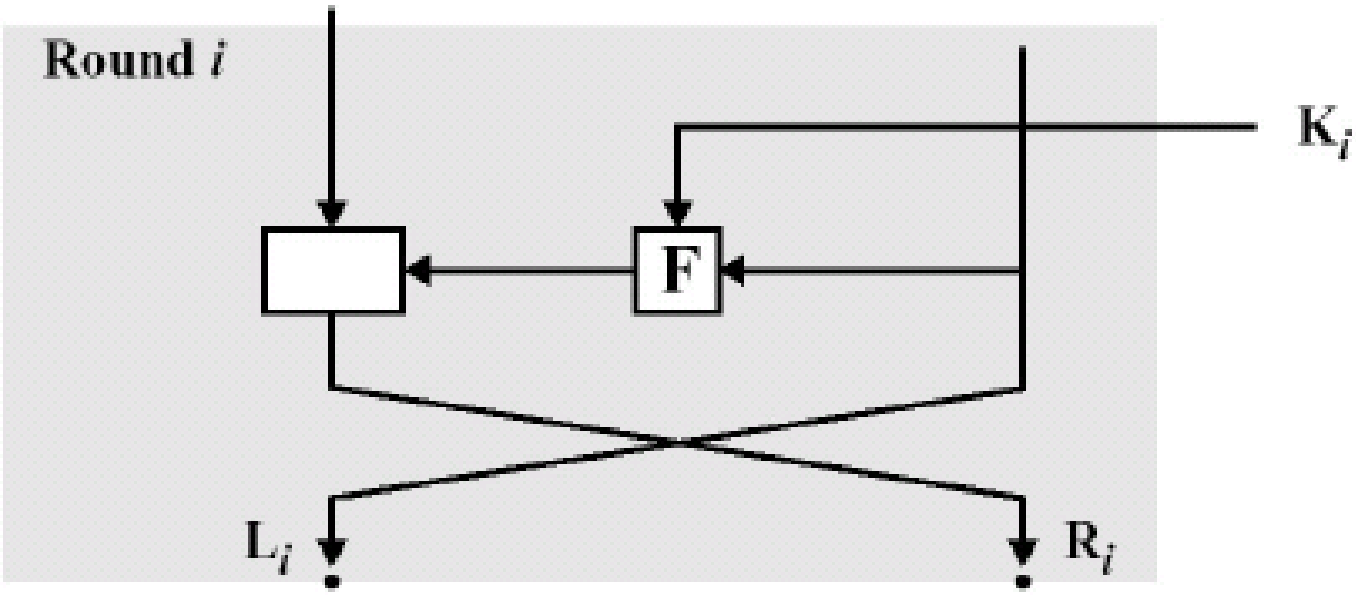
初始置换 IP

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

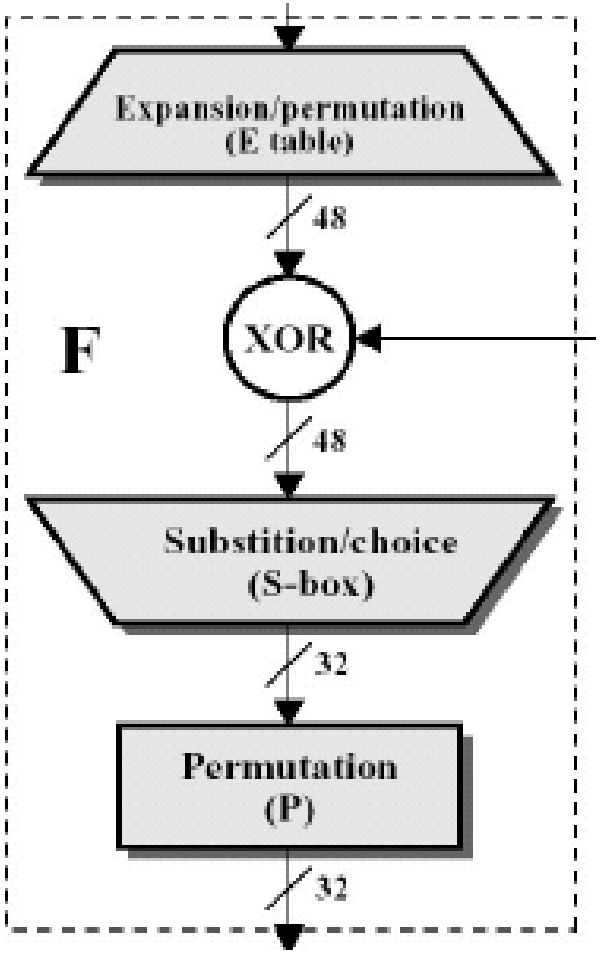
➢单轮操作结构



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

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

➢轮函数F



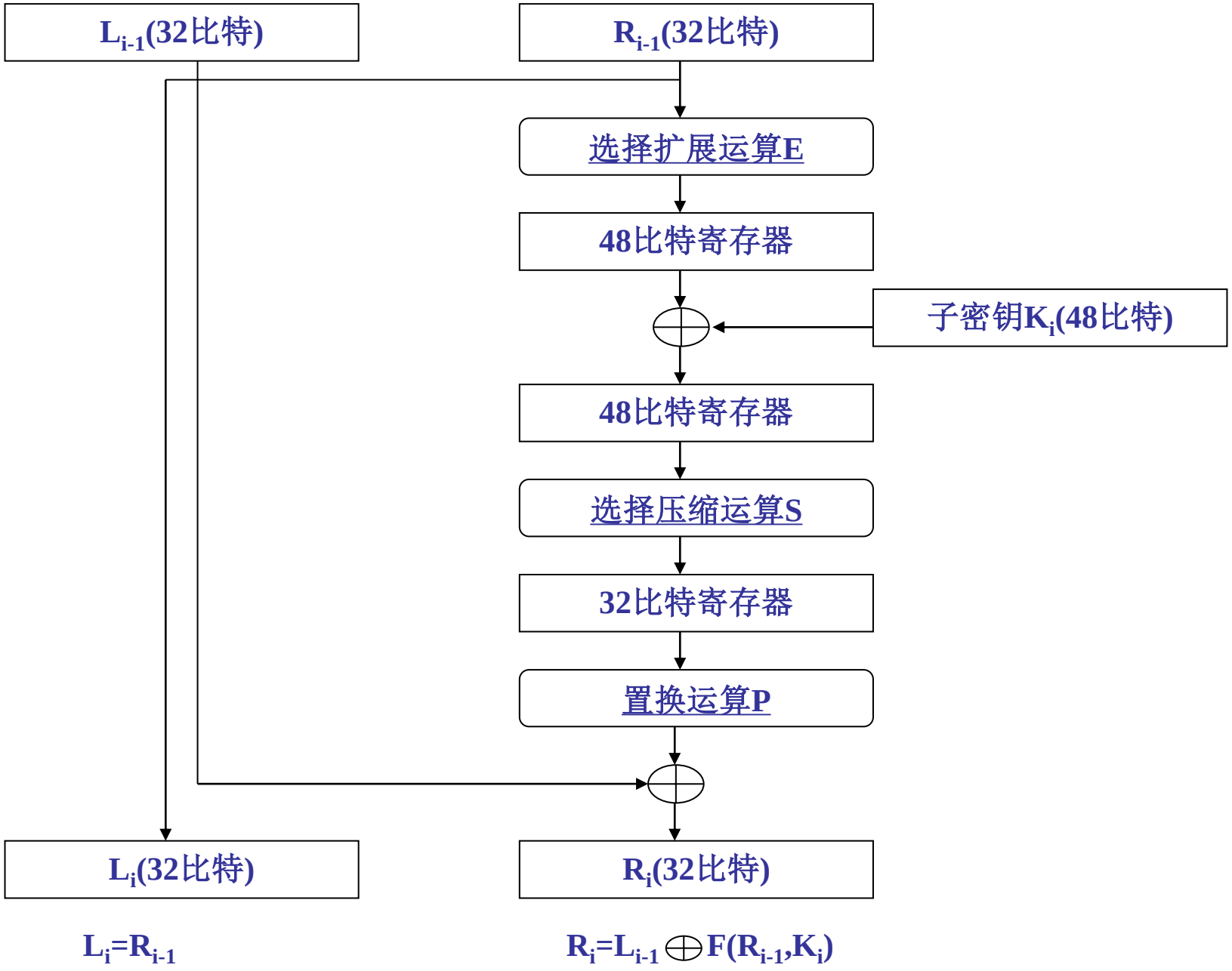
Expansion: 32 → 48

S-box: 6 → 4

Permutation:

3.2 DES加密---单轮变化详细过程

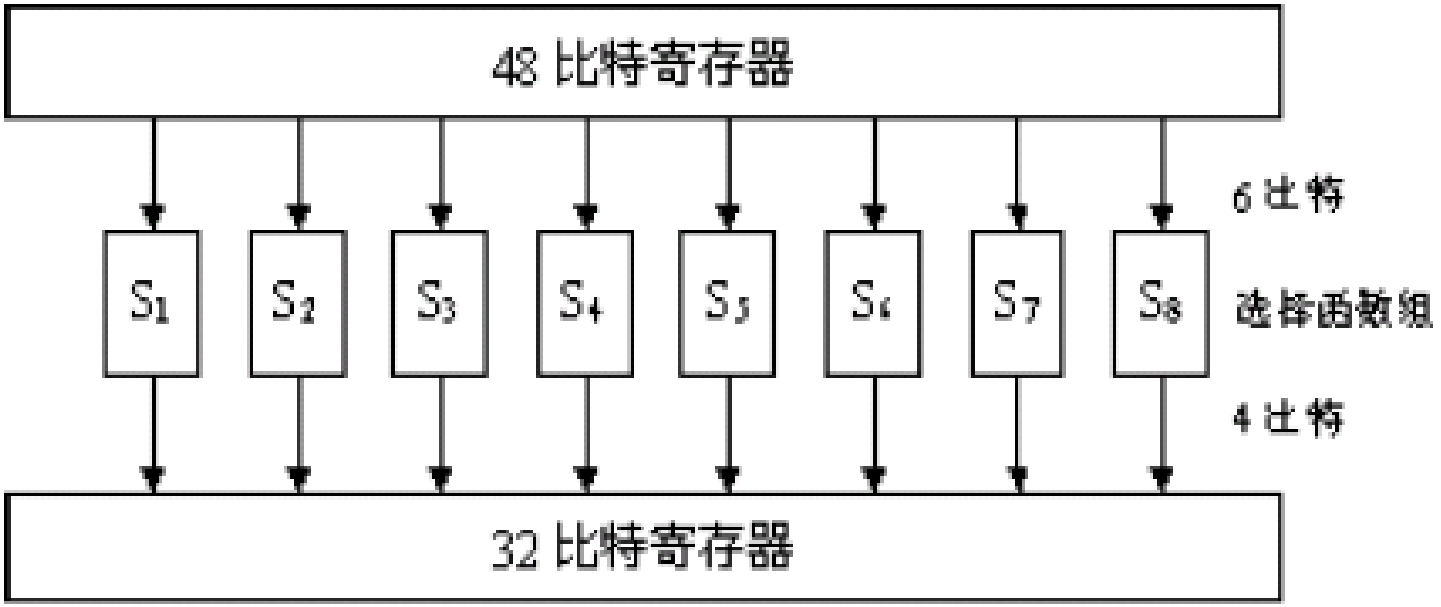
乘积变换



➢扩展置换E：32b ⟶ 48b

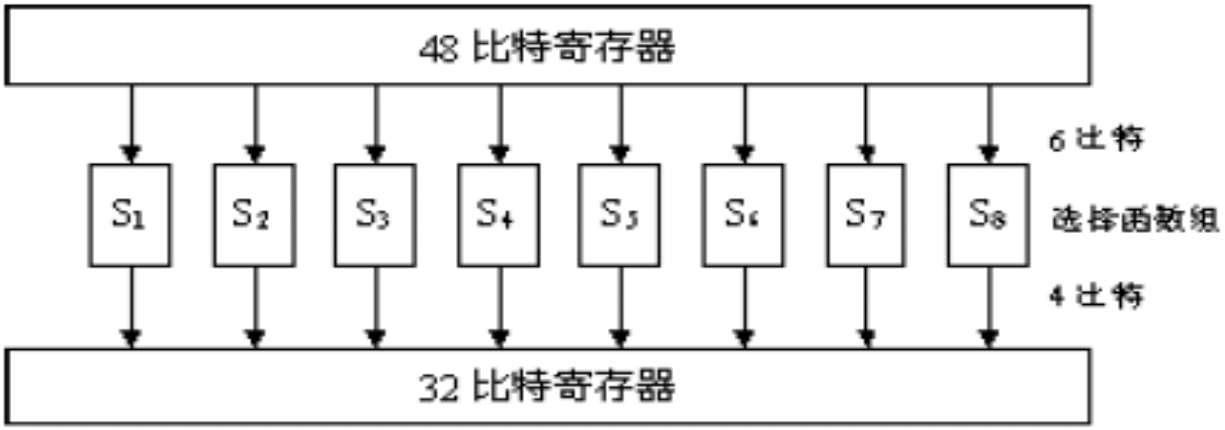
32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	01

➢压缩置换（S盒）



选择压缩运算 S

➢压缩置换（S盒）



选择压缩运算 S

	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	13	02	08	04	06	15	11	01	10	09	03	14	05	00	12	07
1	01	15	13	08	10	03	07	04	12	05	06	11	00	14	09	02
2	07	11	04	01	09	12	14	02	00	06	10	13	15	03	05	08
3	02	01	14	07	04	10	08	13	15	12	09	00	03	05	06	11

输入$b=(110101)_2$，则输出为$(1001)_2$

S盒的内部结构

列 行	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
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	S_1
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	
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	S_2
3	13	8	10	1	3	15	4	2	11	6	7	42	0	5	14	9	
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	S_3
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	
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	S_4
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14	
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	S_5
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	
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	S_6
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13	
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	S_7
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12	
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	S_8
3	2	1	14	7	1	10	7	13	15	12	9	0	3	5	6	11	

S盒的内部计算

- 若输入为$b_1b_2b_3b_4b_5b_6$其中b_1b_6两位二进制数表达了0至3之间的数。$b_2b_3b_4b_5$为四位二进制数，表达0至15之间的某个数。
- 在S1表中的b_1b_6行$b_2b_3b_4b_5$列找到一数m（$0 \le m \le 15$），若用二进制表示为$m_1m_2m_3m_4$，则$m_1m_2m_3m_4$便是它的4bit输出。
- 例如，输入为001111，b_1b_6=01=1，$b_2b_3b_4b_5$=0111=7，即在S1盒中的第1行第7列求得数1，所以它的4bit输出为0001。

关于S盒

- S 盒是DES的核心，也是DES算法最敏感的部分，其设计原理至今仍讳莫如深，显得非常神秘。所有的替换都是固定的，但是又没有明显的理由说明为什么要这样，有许多密码学家担心美国国家安全局设计S盒时隐藏了某些陷门，使得只有他们才可以破译算法，但研究中并没有找到弱点。

➢扩展置换E：32b ⟶ 48b

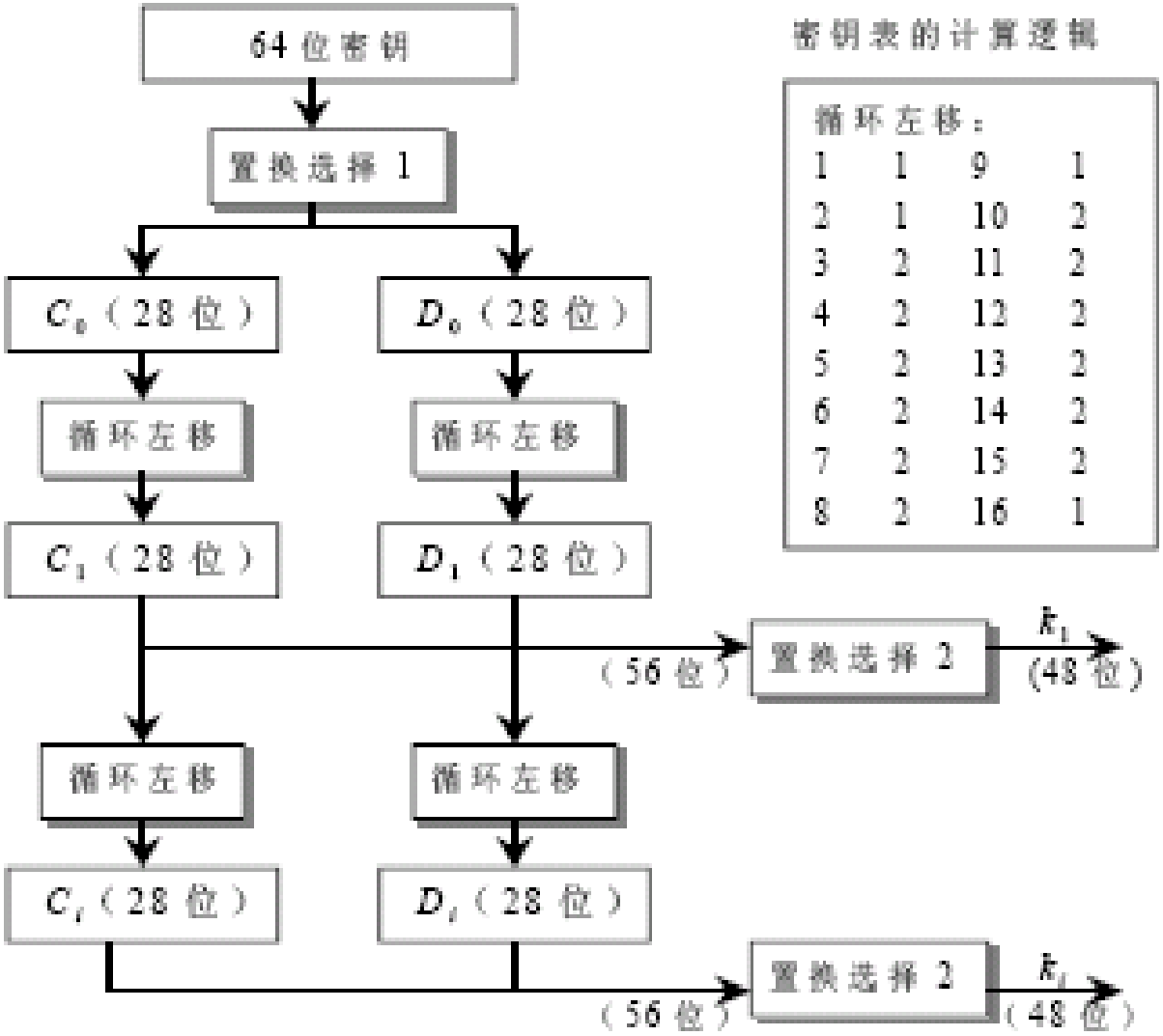
32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	01

➢P盒置换

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

3.2 DES加密---子密钥生成

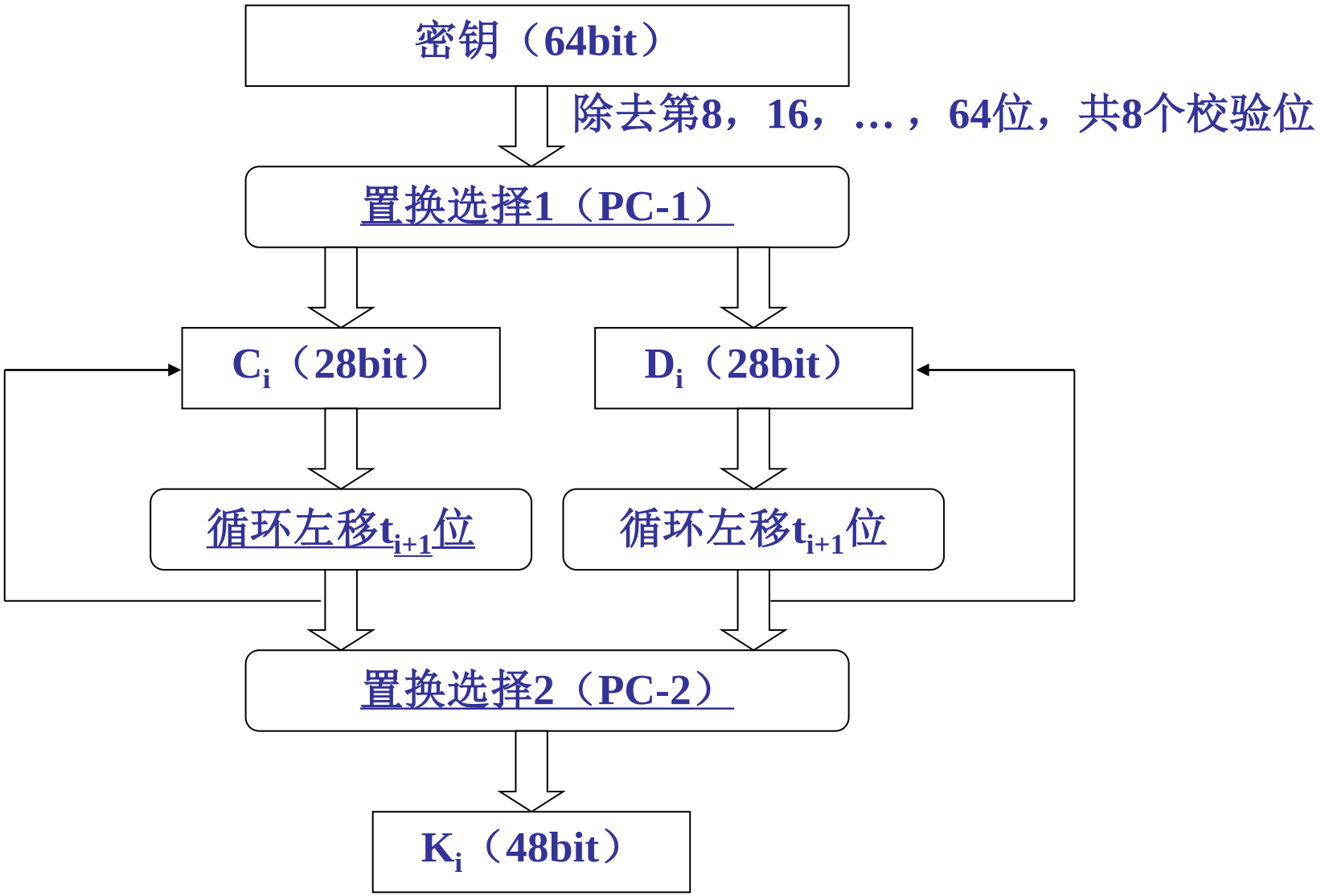
➢ 算法框架



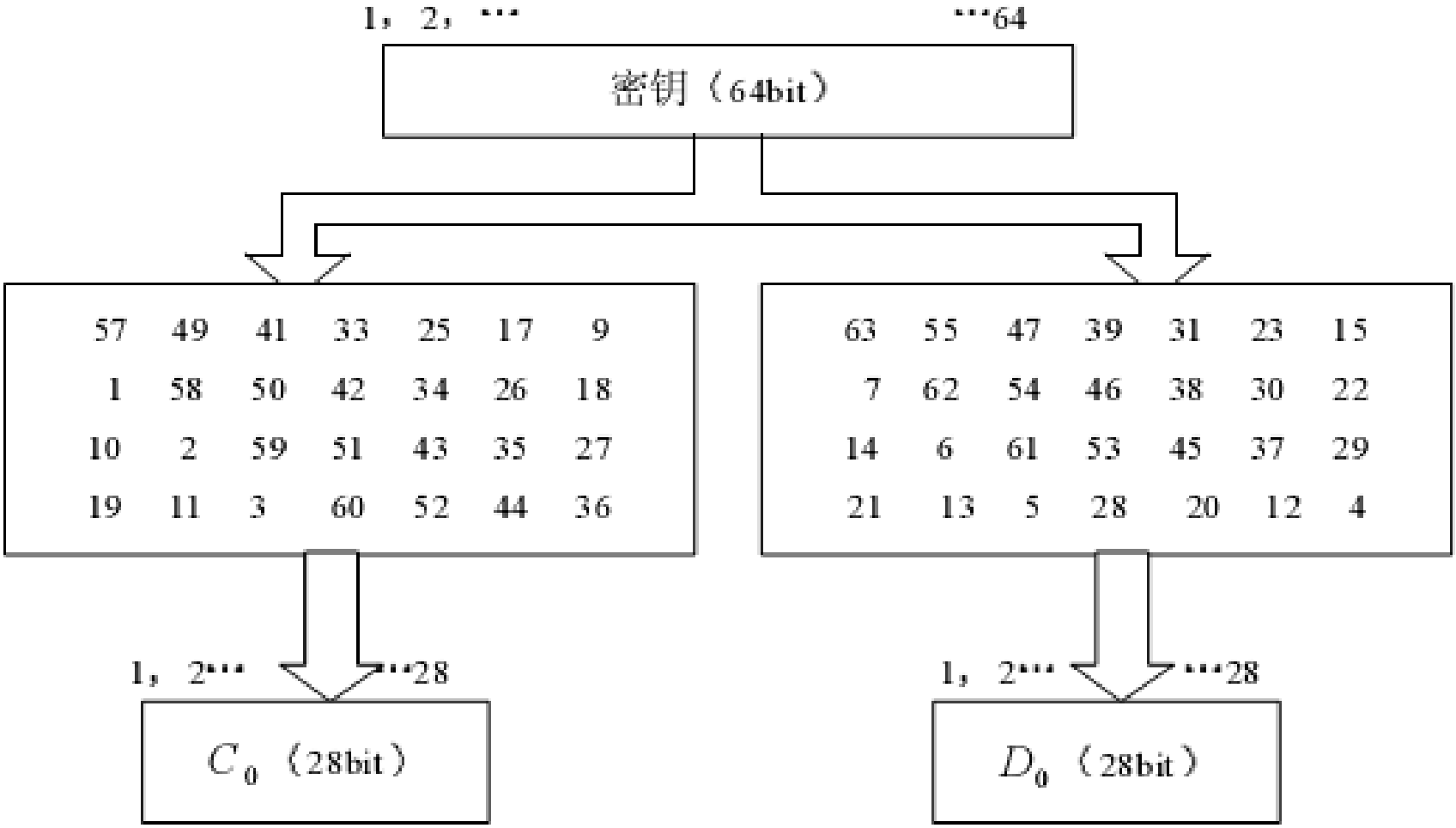
密钥表的计算逻辑

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

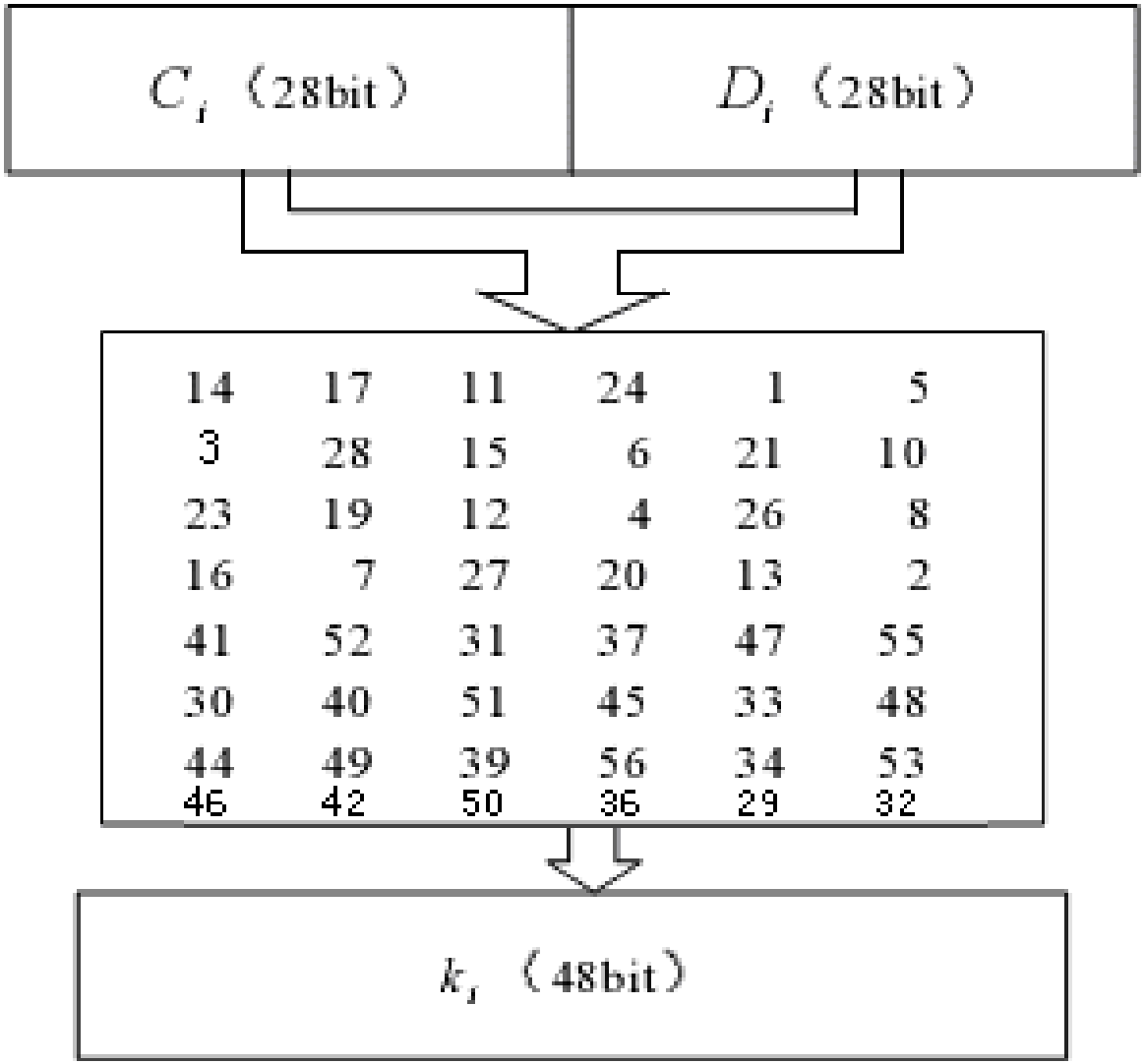
➢ 算法框架


密钥（64bit）
除去第8，16，…，64位，共8个校验位
置换选择1（PC-1）
C_i（28bit）
D_i（28bit）
循环左移t_{i+1}位
循环左移t_{i+1}位
置换选择2（PC-2）
K_i（48bit）

➢置换选择1


1，2，…
…64
密钥（64bit）
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
1，2…
…28
C_0（28bit）
1，2…
…28
D_0（28bit）

➢置换选择2



置换选择PC-2将C中第9 18 22 25位和 D中第7 9 15 26位删去，并将其余数字置换位置后送出48bit数字作为第i次迭代时所用的子密钥ki

- 循环移位

第i次迭代	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
循环左移次数	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

➢解密算法与加密算法相同，只是子密钥的使用次序相反

	加 密		解 密
0	$(L_0, R_0) = IP(m)$		$IP^{-1}(R'_{16}, L'_{16}) = IP^{-1}(L_0, R_0) = m$
1	(L_1, R_1) $= (R_0, L_0 \oplus F(R_0, k_1))$	16	$(L'_{16}, R'_{16}) = (R'_{15}, L'_{15} \oplus F(R'_{15}, k_1))$ $= (R_0, (L_0 \oplus F(R_0, k_1))$ $\oplus F(R_0, k_1))$ $= (R_0, L_0)$ 注：$(L_0, R_0) = IP(m)$
2	(L_2, R_2) $= (R_1, L_1 \oplus F(R_1, k_2))$		$(L'_{15}, R'_{15}) = (R'_{14}, L'_{14} \oplus F(R'_{14}, k_2))$ $= (R_1, (L_1 \oplus F(R_1, k_2))$ $\oplus F(R_1, k_2))$ $= (R_1, L_1)$ $= (L_0 \oplus F(R_0, k_1), R_0)$
	⋮		⋮

➢解密算法与加密算法相同，只是子密钥的使用次序相反

15	(L_{15}, R_{15}) $= (R_{14}, L_{14} \oplus F(R_{14}, k_{15}))$		$(L_2', R_2') = (R_1', L_1' \oplus F(R_1', k_{15}))$ $= (R_{14}, (L_{14} \oplus F(R_{14}, k_{15}))$ $\oplus F(R_{14}, k_{15}))$ $= (R_{14}, L_{14})$ $= (L_{13} \oplus F(R_{13}, k_{14}), R_{13})$
16	(L_{16}, R_{16}) $= (R_{15}, L_{15} \oplus F(R_{15}, k_{16}))$	1	$(L_1', R_1') = (R_0', L_0' \oplus F(R_0', k_{16}))$ $= (R_{15}, (L_{15} \oplus F(R_{15}, k_{16}))$ $\oplus F(R_{15}, k_{16}))$ $= (R_{15}, L_{15})$ $= (L_{14} \oplus F(R_{14}, k_{15}), R_{14})$
	注：$X = (R_{16}, L_{16})$； $C = IP^{-1}(X)$ $= IP^{-1}(R_{16}, L_{16})$		$(L_0', R_0') = IP(C)$ $= (R_{16}, L_{16})$ $= (L_{15} \oplus F(R_{15}, k_{16}), R_{15})$

The Avalanche Effect---DES具有良好的雪崩效应

- **The Avalanche Effect：**

  a small change in either the plaintext or the key should produce a significant change in the ciphertext.

- **DES显示了很强的雪崩效应**

雪崩效应举例a:

- two plaintexts that differ by one bit :

P1:00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000

P2:10000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000

- with the key

0000001 1001011 0100100 1100010 0011100
0011000 0011100 0110010

雪崩效应举例b:

- a single plaintext is input:

  01101000 10000101 00101111 01111010 00010011 01110110 11101011 10100100

- with two keys that differ in only one bit position:

  1110010 1111011 1101111 0011000 0011101 0000100 0110001 11011100

  0110010 1111011 1101111 0011000 0011101 0000100 0110001 11011100

(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

- DES在20多年的应用实践中，没有发现严重的安全缺陷，在世界范围内得到了广泛的应用，为确保信息安全做出了不可磨灭的贡献。
- 存在的安全弱点：
  - The Use of 56-Bit Keys：56位密钥空间约为$7.2*10^{16}$。1997年6月Rocke Verser小组通过因特网利用数万台微机历时4个月破译了DES；1998年7月，EFF用一台25万美元的机器，历时56小时破译DES。
  - 存在弱密钥：有的密钥产生的16个子密钥中有重复者。
  - 互补对称性： C=DES(M,K), 则 C’=DES(M’,K’)， 其中，M’,C’,K’是M,C,K的非。

DES的强度

- 56位的密钥
- S盒的问题
- 计时攻击：算法对不同的输入在时间上有细微差别，来获得密钥与明文的信息
- 弱密钥

弱密钥

问题 $DES_k(P)=DES_k^{-1}(P)$（当 $K_1=K_2=\cdots=K_{16}$）

条件：C_0=全 0 或全 1，D_0=全 0 或全 1，即对原始密钥 $k=(k_1k_2\cdots k_{64})$，（由置换 PC—1 知）有 $k_{57}=k_{49}=k_{41}=\cdots=k_{44}=k_{36}=0$ 或 1，$k_{63}=k_{55}=k_{47}=\cdots=k_{12}=k_4=0$ 或 1

具体为 $(01\ 01\ 01\ 01\ 01\ 01\ 01\ 01)_{16}$，对应（$C_0$，$D_0$）$=(0000000\quad 0000000)_{16}$，

$(EF\ EF\ EF\ EF\ EF\ EF\ EF\ EF)_{16}$，对应（$C_0$，$D_0$）$=(0000000\quad FFFFFFF)_{16}$，

$(E0\ E0\ E0\ E0\ E0\ E0\ E0\ E0)_{16}$，对应（$C_0$，$D_0$）$=(FFFFFFF\quad 0000000)_{16}$，

$(E0\ E0\ E0\ E0\ E0\ E0\ E0\ E0)_{16}$，对应（$C_0$，$D_0$）$=(FFFFFFF\quad 0000000)_{16}$，

实质上应为 8 个弱密钥，即上述 4 种的每一个的二进制表示中的第 8j 位取反，如第一种改为$(00\ 00\ 00\ 00\ 00\ 00\ 00\ 00)_{16}$，仍对应（$C_0$，$D_0$）$=(0000000\quad 0000000)_{16}$。

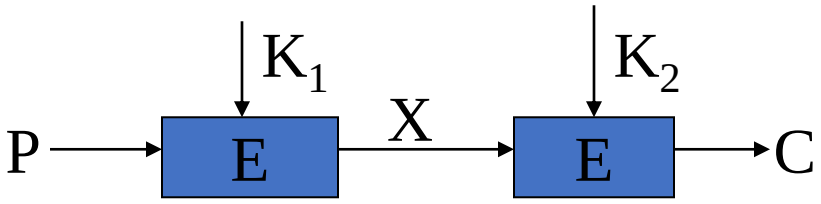
二重DES—Double DES

- 给定明文P和两个加密密钥k_1和k_2，采用二重DES对P进行加密E，有

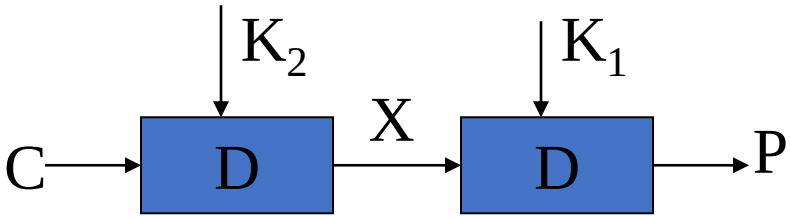
  密文　$C=E_{K2}(E_{K1}(P))$

对C进行解密D，有

明文　$P=D_{K1}(D_{K2}(C))$



加密图



解密图

二重DES—Double DES

- 如果采用普通穷举法来攻击二重DES，其攻击代价并不是2^{112}。这是因为一个分组为64bit，明文空间为2^{64}，而密钥空间为2^{112}。因此对于某个明文P，可产生密文C的密钥平均个数为$2^{112}/2^{64}=2^{48}$个。换句话说，大约有2^{48}个不同的密钥，对明文P加密后得到的密文相同且都等于C。这就需要另一个（P，C）对，以从这2^{48}个不同的密钥中确定真正的密钥。因此总的计算代价为$2^{112}+2^{48}$。

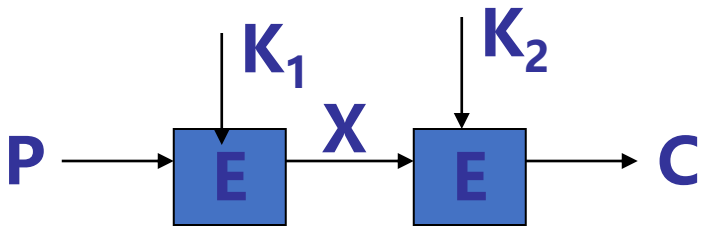
二重DES很难抵挡住中间相遇攻击法

(Meet-in-the-Middle Attack)

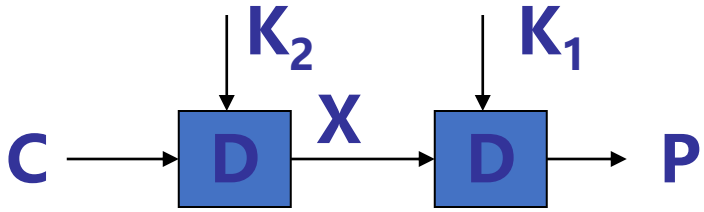
$$C=E_{K2}(E_{K1}(P))$$

从图中可见

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



加密



解密

若给出一个已知的明密文对（P,C)做：对2^{56}个所有密钥K_1做对明文P的加密，得到一张密钥对应于密文X的一张表；类似地对2^{56}个所有可能的密钥K_2做对密文C的解密，得到相应的“明文”X。做成一张X与K_2的对应表。比较两个表就会得到真正使用的密钥对K_1,K_2。

三重两密DES

(Triple DES with Two Keys)

- Tuchman给出双密钥的EDE模式（加密-解密-加密）：

$C=E_{K1}(D_{K2}(E_{K1}(P)))$ ……对P加密

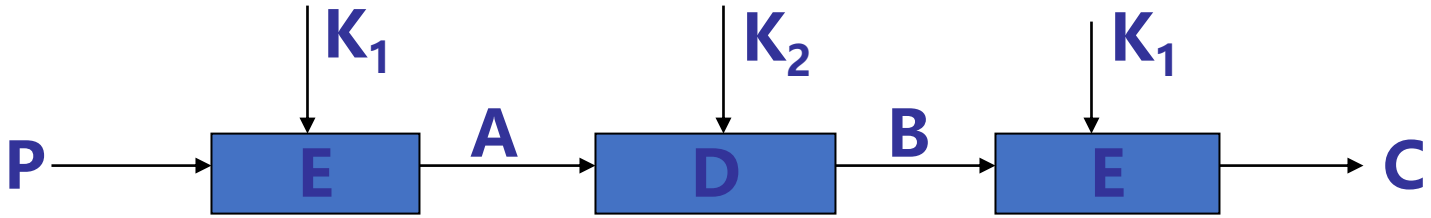
$P=D_{K1}(E_{K2}(D_{K1}(C)))$ ……对C解密

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

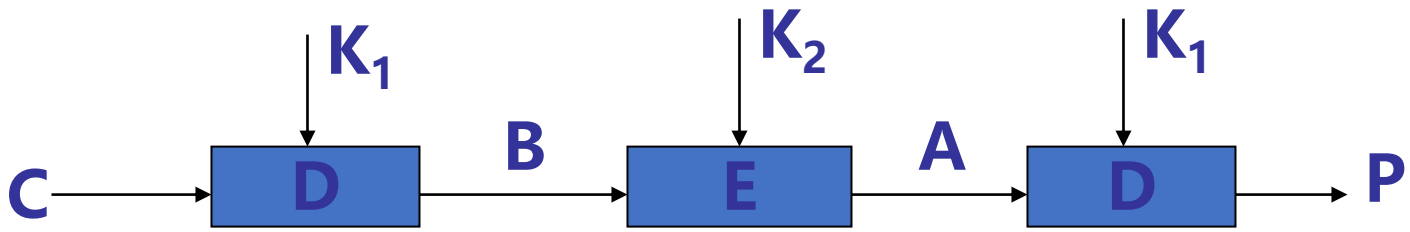
(The Key Manager Standards ANSX9.17和ISO8732).

三重两密DES

(Triple DES with Two Keys)



加密图



解密图

三重两密DES

(Triple DES with Two Keys)

- 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。

注意：

- Merkle和Hellman设法创造一个条件，想把**中间相遇攻击（meet-in-the-middle attack）**的方法用于三重DES，但目前也不太成功。

三重两密DES

(Triple DES with Two Keys)

- 虽然对上述带双密钥的三重DES到目前为止还没有好的实际攻击办法，但人们还是放心不下，又建议使用三密钥的三重DES，此时密钥总长为168bits.

$$C=E_{K3}(D_{K2}(E_{K1}(P)))$$

- A number of Internet-based applications have adopted three-key 3DES, including PGP and S/MIME.

AES---Advanced Encryption Standard

- 3DES的根本缺点在于用软件实现该算法的速度比较慢。
- 分组长度64位，从运算效率和安全性考虑，分组长度应该更大。
- 1997年1月，美国国家标准局NIST向全世界密码学界发出征集21世纪高级加密标准算法的公告，并成立了AES标准工作研究室，1997年4月15日的例会制定了对AES的评估标准。

AES的要求

(1) AES是公开的;

(2) AES为单钥体制分组密码;

(3) AES的密钥长度可变，可按需要增大;

(4) AES适于用软件和硬件实现;

(5) AES可以自由地使用，或按符合美国国家标准（ANST）策略的条件使用;

算法衡量条件

- 满足以上要求的AES算法，需按下述条件判断优劣

  a. 安全性

  b. 计算效率

  c. 内存要求

  d. 使用简便性

  e. 灵活性。

AES的评审

- 1998年4月15日全面征集AES算法的工作结束。1998年8月20日举行了首届AES讨论会，对涉及14个国家的密码学家所提出的候选AES算法进行了评估和测试，初选并公布了15个被选方案，供大家公开讨论。

  CAST-256，RC-6，CRYPTON-128，DEAL-128，
  FROG，DFC，LOKI-97，MAGENTA，
  MARS，HPC，RIJNDAEL，SAFER+，
  SERPENT，E-2，TWOFISH。

- 这些算法设计思想新颖，技术水平先进，算法的强度都超过3-DES，实现速度快于3-DES。

AES的评审

- 1999年8月9日NIST宣布第二轮筛选出的5个候选算法为：

  MARS(C.Burwick等,IBM），

  RC6™（R. Rivest等,RSA Lab.),

  RIJNDEAL(J. Daemen,比),

  SERPENT(R. Anderson等，英、以、挪威),

  TWOFISH(B. Schiener)。

- 2000年10月2日，NIST宣布Rijndael作为新的AES

群

- 群:是一个代数系统，它由一个非空集合组成，在集合上定义了一个二元运算，其满足：

  封闭性：对任意的 $a,b\in G$ ，$a\cdot b\in G$ ；

  结合律：对任何的 $a,b,c\in G$ ，有

  $$a\cdot b\cdot c=(a\cdot b)\cdot c=a\cdot(b\cdot c)$$ ；

  单位元：存在一个元素 $1\in G$ (称为单位元)，对任意元素有：$a\cdot 1=1\cdot a=a$ ；

  逆元：对任意 $a\in G$ ，存在一个元素 $a^{-1}\in G$ (称为逆元)，使得 $a\cdot a^{-1}=a^{-1}\cdot a=1$ 。

- 记作：$<G,\cdot>$

交换群与有限群

- 若群还满足交换律，即对任何 $a,b \in G$ ，有：$a \cdot b = b \cdot a$，则称为交换群(或加法群，阿贝尔群等)。
- 若集合G中只含有有限多个元素，则我们称 $<G,\cdot>$ 为有限群，此时，把集合G中元素的个数称为有限群的阶。

群的性质

- 群中的单位元是惟一的；
- 消去律成立，即对任意的 $a,b,c \in G$ ：

  如果 $ab = ac$ ，则 $b = c$

  如果 $ba = ca$ ，则 $b = c$
- 群中的每一元素的逆元是惟一的。

域

- 域是一个代数系统，它由一个(至少包含两个元素的)非空集合F组成，在集合F上定义有两个二元运算：加法与乘法，并满足下面条件：

  F的元素关于加法成交换群，记其单位元为“0”(称为域的零元)

  F关于乘法成交换群，记其单位元为“1”(称其为域的单位元)

  乘法在加法上满足分配律，即

$$a\cdot(b+c)=ab+ac \quad (a+b)\cdot c=ac+bc$$

- 记为 $<F,+,\cdot>$

有限域

- 若集合F只包含有限个元素，则称这个域F为有限域。有限域中元素的个数称为该有限域的阶。

- 若有一任意的素数P和正整数 $n \in Z^+$ ，存在 P^n 阶有限域，这个有限域记为 $GF(P^n)$。当 n=1 时，有限域 $GF(P)$ 称为素域，定义为整数{0，1，...，p-1}的集合 Z_p，其运算为模p运算
- 每个元素都与P互素，所以有乘法逆元。

GF(2^8)域上的多项式表示及运算

- 在AES加密系统中，是在不可约多项式 $m(x)=x^8+x^4+x^3+x+1$上构造的有限域
- 一个字节的 $GF(2^8)$ 元素的二进制展开成的多项式系数为：$b_7b_6b_5b_4b_3b_2b_1b_0$
- 例如，$GF(2^8)$ 上的“37”(为十六进制)，其二进制为“00110111”，对应多项式为：

$$x^5+x^4+x^2+x^1+1$$

GF(2^8)域上的加法

加法运算就是以字节为单位进行比特异或运算。如16进制37+83=B4，采用二进制表示为

00110111+10000011=10110100

采用多项式表示为：

$$(x^5+x^4+x^2+x+1)+(x^7+x+1)=x^7+x^5+x^4+x^2$$

GF(2^8)域上的模运算

采用以字节为单位的比特异或运算来进行

37 mod (07) =01

采用多项式运算有

$(x^5+x^4+x^2+x+1) \bmod (x^2+x+1) = 1$

采用长除法进行运算

$$
\begin{array}{r}
x^3+x \\
x^2+x+1 \overline{)\ x^5+x^4+x^2+x+1} \\
\underline{x^5+x^4+x^3\ \ \ \ \ \ \ \ } \\
x^3+x^2+x+1 \\
\underline{x^3+x^2+x\ \ \ \ } \\
1
\end{array}
$$

GF(2^8)域上的乘法

$$57 \cdot 83 = C1$$

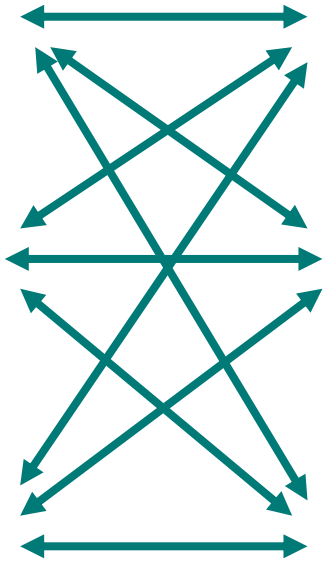
$$(x^6 + x^4 + x^2 + x + 1)(x^7 + x + 1) \bmod (m(x))$$

$$= (x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1)$$

$$\bmod (x^8 + x^4 + x^3 + x + 1)$$

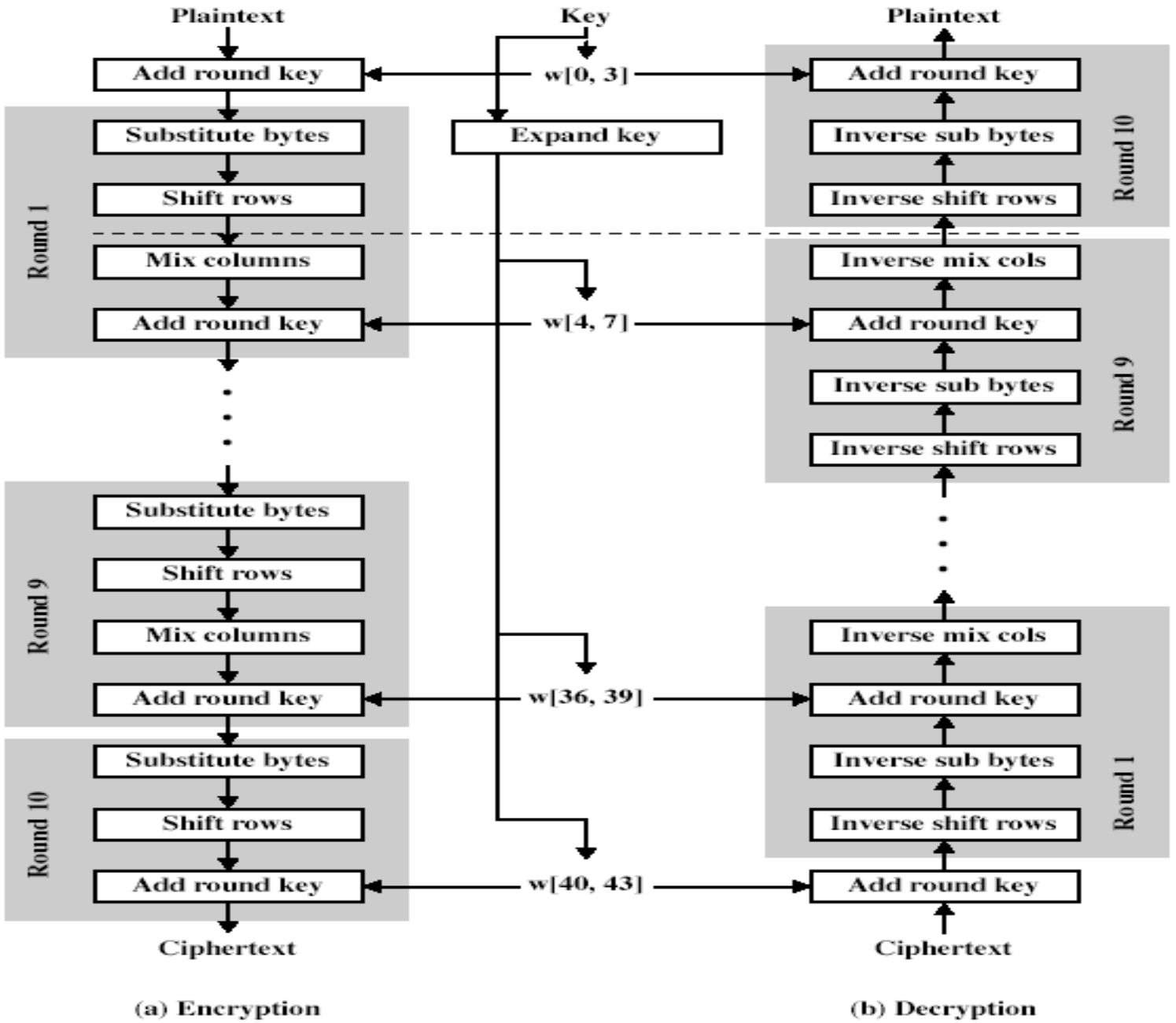
$$= x^7 + x^6 + 1$$ （表示为16进制为C1）。

分组长度（bit）
128
192
256



密钥长度（bit）
128
192
256

3.3 高级加密标准---AES算法的一般描述

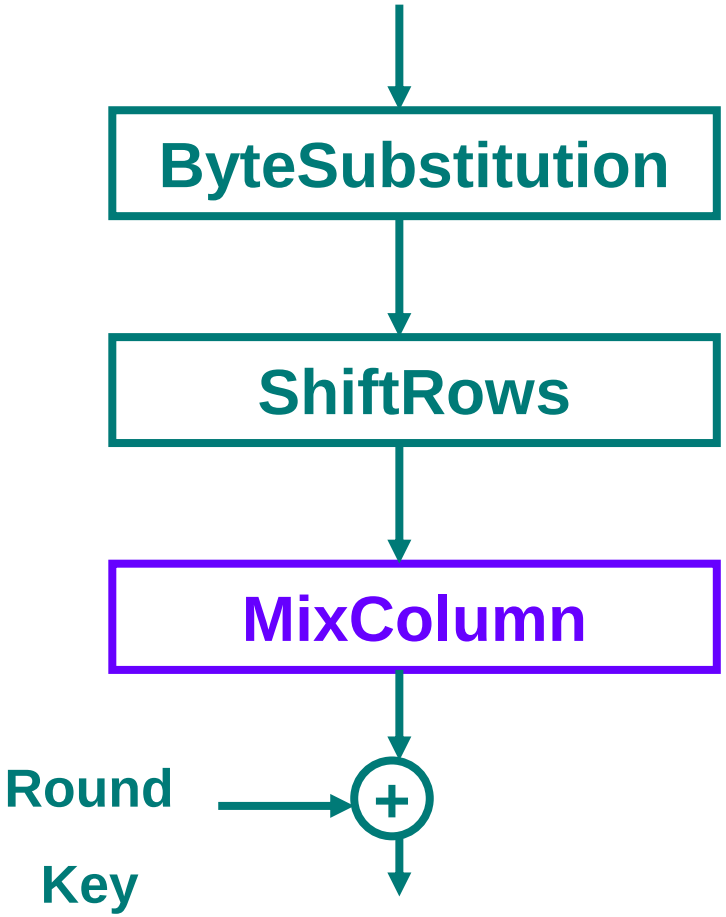


轮数（Round）的不同取值

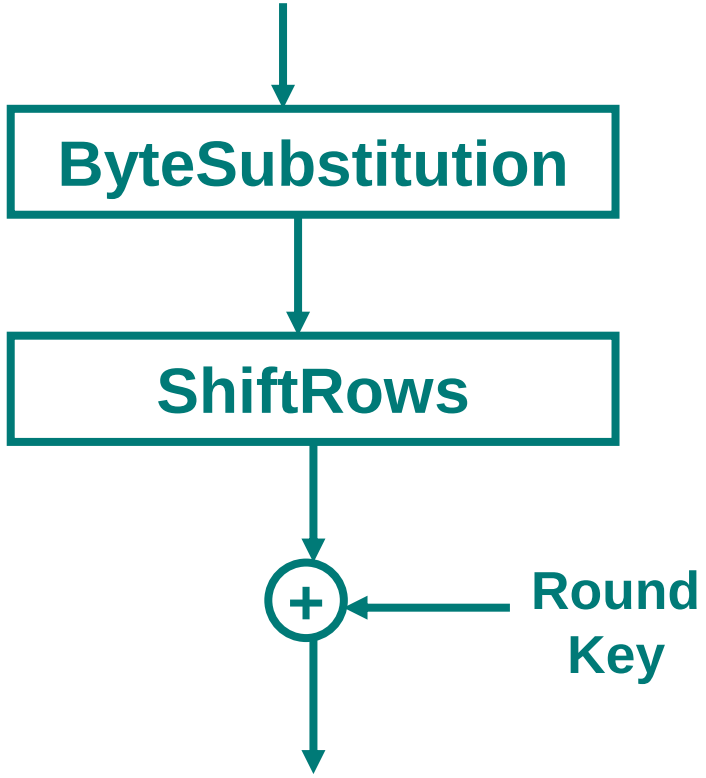
轮数（Round）	Block Length=128	Block Length=192	Block Length=256
Key Length=128	10	12	14
Key Length=192	12	12	14
Key Length=256	14	14	14

作用：通过重复简单的非线性变换、混合函数变换，将字节代换运算产生的非线性扩散，达到充分的混合，使加密后的分组信息统计特性分布更均匀，在每轮迭代中引入不同的密钥，这样便以最简单的运算代价得到最好的加密效果，实现加密的有效性。

Rijndael Round的构成



一般的轮变换



最后一轮的轮变换

一些相关的的术语定义和表示

- 状态（State）：密码运算的中间结果称为状态。
- State的表示：状态用以字节为基本构成元素的矩阵阵列来表示，该阵列有4行，列数记为Nb。
  Nb=分组长度（bits）÷ 32
  Nb可以取的值为4，6，8，对应的分组长度为128，192， 256 bits。
- 密码密钥（Cipher Key）的表示： Cipher Key类似地用一个4行的矩阵阵列来表示，列数记为Nk。
  Nk=密钥长度（bits）÷32
  Nk可以取的值为4，6，8，对应的密钥长度为128，192， 256 bits。

一些相关的的术语定义和表示

- 状态（State）：密码运算的中间结果称为状态。
- State的表示：状态用以字节为基本构成元素的矩阵阵列来表示，该阵列有4行，列数记为Nb。
  Nb=分组长度（bits）÷ 32
  Nb可以取的值为4，6，8，对应的分组长度为128，192， 256 bits。
- 密码密钥（Cipher Key）的表示： Cipher Key类似地用一个4行的矩阵阵列来表示，列数记为Nk。
  Nk=密钥长度（bits）÷32
  Nk可以取的值为4，6，8，对应的密钥长度为128，192， 256 bits。

- AES每轮要经过四次变换，分别是
  - 字节代换运算(SubByte ())
  - ShiftRows()(行移位)变换
  - MixColumns()（列混合）变换
  - AddRoundKey() (轮密钥的混合)变换

我们用的128bit的密钥（循环次数为10），那么每次加密的分组长为128bit，16个字节，每次从明文中按顺序取出16个字节假设为$a_0a_1a_2a_3a_4a_5a_6a_7a_8a_9a_{10}a_{11}a_{12}a_{13}a_{14}a_{15}$，这16个字节在进行变换前先放到一个4×4的矩阵中。如下页图所示：

S_0	S_4	S_8	S_{12}
S_1	S_5	S_9	S_{13}
S_2	S_6	S_{10}	S_{14}
S_3	S_7	S_{11}	S_{15}

这个矩阵称为状态(state)

以后所有的变换都是基于这个矩阵进行的，到此，准备工作已经完成。现在按照前面的顺序进行加密变换，首先开始第一次循环的第一个变换：字节代换(SubByte ())。

1. ByteSubstitution(字节代换)

ByteSubstitution是一个非线性的字节替代，独立地在每个状态字节上进行运算。它包括两个变换。

1. 在有限域GF(2^8)上求乘法逆。
2. 在GF(2)上进行下面的仿射变换：

$$\begin{bmatrix} y0 \\ y1 \\ y2 \\ y3 \\ y4 \\ y5 \\ y6 \\ y7 \end{bmatrix} = \begin{bmatrix} 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \\ 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \end{bmatrix} \begin{bmatrix} x0 \\ x1 \\ x2 \\ x3 \\ x4 \\ x5 \\ x6 \\ x7 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

若$b(x) \in GF(2^8)$,则有$b^{-1}(x) \in GF(2^8)$,则有$b(x) \cdot b^{-1}(x) = 1\ mod(m(x))$,这里$m(x) = x^8 + x^4 + x^3 + x + 1$。如$\{f6\}$（十六进制）表示为$x^7 + x^6 + x^5 + x^4 + x^2 + x$，$\{03\}$表示为$x + 1$，显然有$\{f6\} \cdot \{03\} = 1mod(m(x))$。实际求逆采用设未知数解方程组的方法来实现。

例如求{f6}的SubByte()变换。先求其逆为{03}，即为{00000011}，将其带入上式可计算$\{y_7y_6y_5y_4y_3y_2y_1y_0\} = \{01000010\}$, 即为{42}。

取数{f6}的高低位对应xy值为，x=f，y=6；再查表得到第x行第y列（即f行6列）的数值为{42}。

3.3 高级加密标准--- AES加密算法的具体实现

		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

2. ShiftRows()(行移位)变换

状态阵列的后3行循环移位不同的偏移量。第1行循环移位C1字节，第2行循环移位C2字节，第3行循环移位C3字节。

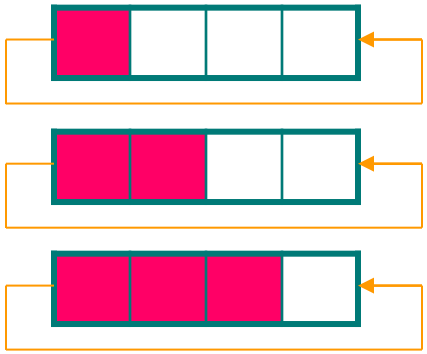
偏移量C1、C2、C3与分组长度Nb有关，如下表所示：

N_k	C1	C2	C3
4	1	2	3
6	1	2	3
8	1	3	4

2. ShiftRows()(行移位)变换

S

S_{00}	S_{01}	S_{02}	S_{03}
S_{10}	S_{11}	S_{12}	S_{13}
S_{20}	S_{21}	S_{22}	S_{23}
S_{30}	S_{31}	S_{32}	S_{33}



行变换示意图

S'

S_{00}	S_{01}	S_{02}	S_{03}
S_{11}	S_{12}	S_{13}	S_{10}
S_{22}	S_{23}	S_{20}	S_{21}
S_{33}	S_{30}	S_{31}	S_{32}

3. MixColumns()(列混合)变换

$$\begin{bmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ s_{10} & s_{11} & s_{12} & s_{13} \\ s_{20} & s_{21} & s_{22} & s_{23} \\ s_{30} & s_{31} & s_{32} & s_{33} \end{bmatrix} \xrightarrow{\text{MixColumns()}} \begin{bmatrix} s_{00}' & s_{01}' & s_{02}' & s_{03}' \\ s_{10}' & s_{11}' & s_{12}' & s_{13}' \\ s_{20}' & s_{21}' & s_{22}' & s_{23}' \\ s_{30}' & s_{31}' & s_{32}' & s_{33}' \end{bmatrix}$$

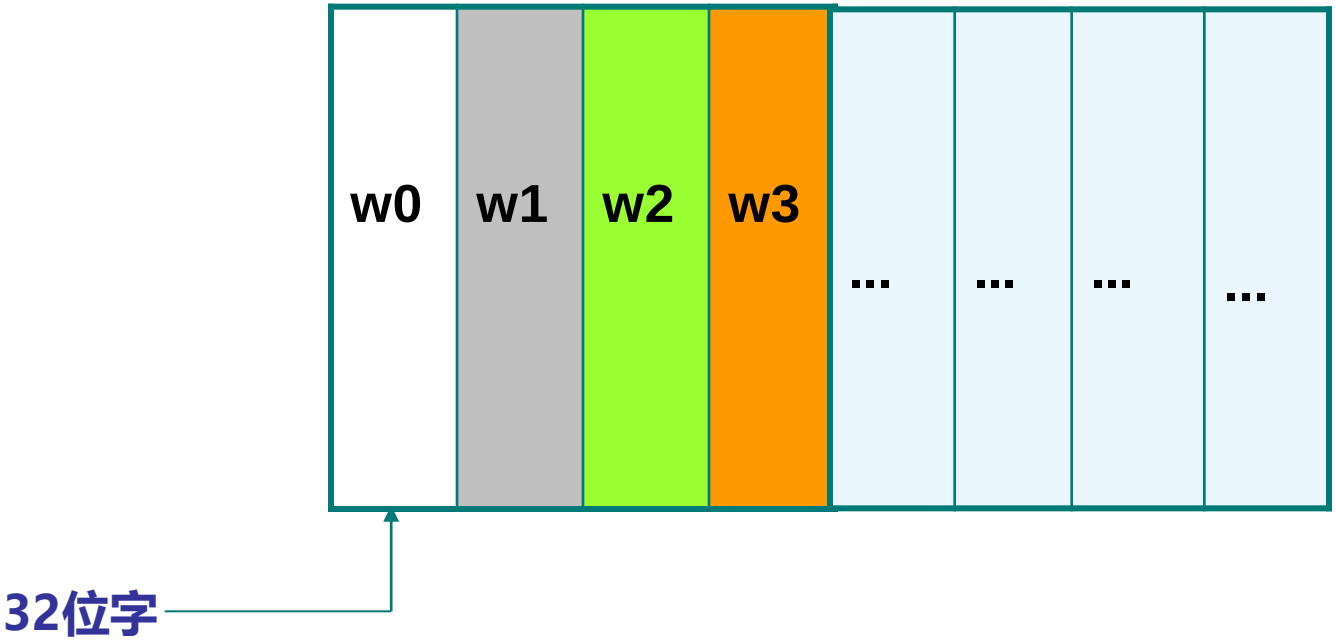
$$\begin{bmatrix} s_{0i}' \\ s_{1i}' \\ s_{2i}' \\ s_{3i}' \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0i} \\ s_{1i} \\ s_{2i} \\ s_{3i} \end{bmatrix}$$

4. AddRoundKey()轮密钥加变换

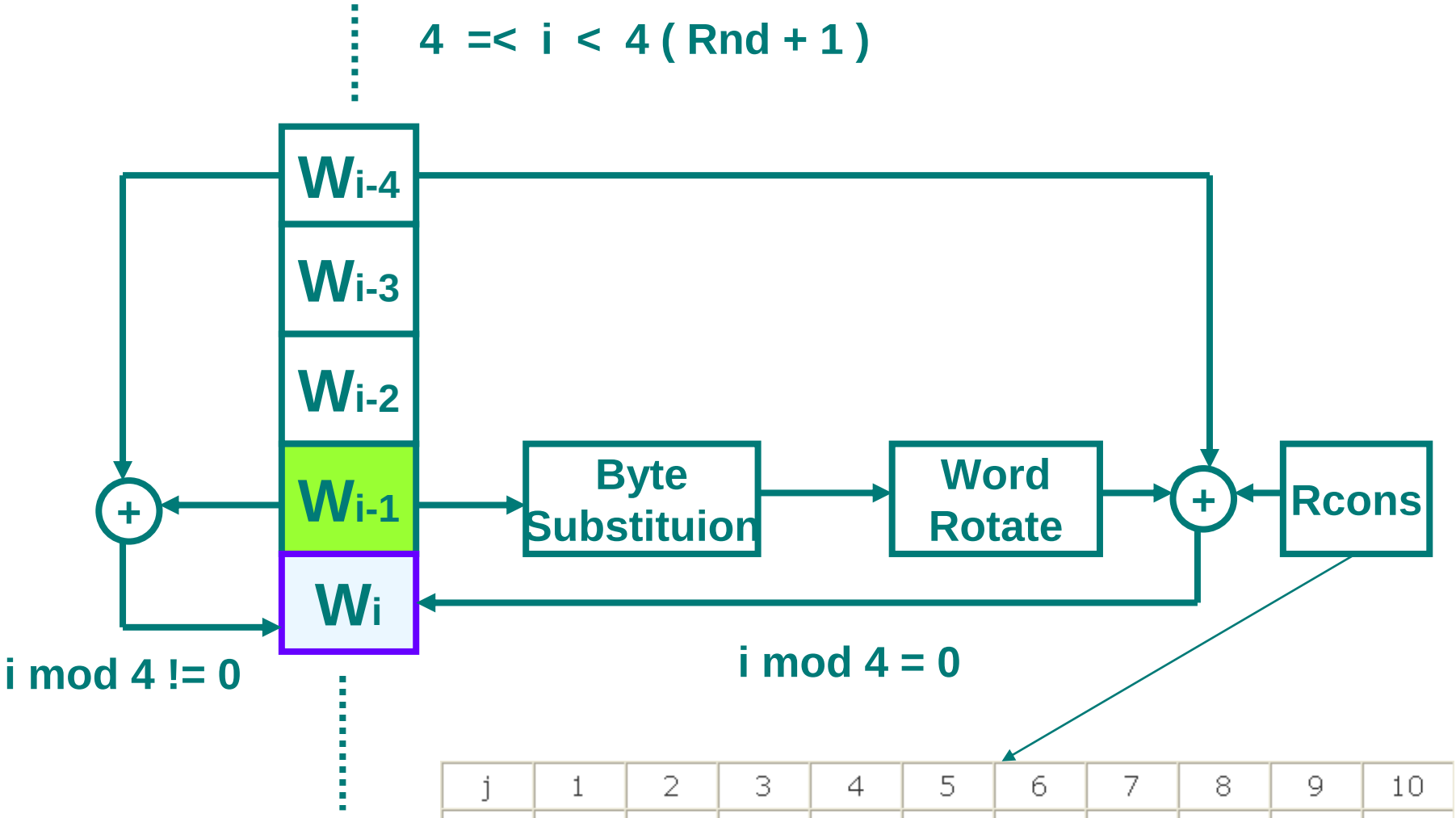
$K_{0,0}$	$K_{0,1}$	$K_{0,2}$	$K_{0,3}$
$K_{1,0}$	$K_{1,1}$	$K_{1,2}$	$K_{1,3}$
$K_{2,0}$	$K_{2,1}$	$K_{2,2}$	$K_{2,3}$
$K_{3,0}$	$K_{3,1}$	$K_{3,2}$	$K_{3,3}$

当分组长度和密钥长度都为128bit时：
AES的加密算法共迭代10轮，需11个子密钥，即44个32位字。

扩展的目的：将一个128bit的密钥，扩展为11个128bit的密钥。



4. AddRoundKey()轮密钥加变换



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

Key expansion

- AES有更长的密钥，密钥长度可为128bits、192bits和256bits三种情况，明显提高了加密的安全性，同时，对不同机密级别的信息，可采用不同长度的密钥，执行灵活性较高。
- AES的均衡对称结构既可以提高执行的灵活度，又可防止差分分析方法的攻击。
- AES算法的迭代次数最多为14次，S盒只有一个，较之DES的16次迭代和8个S盒要简单得多。
- AES算法在所有的平台上都表现良好

- “一次一密”密码在理论上是不可攻破的。流密码则由“一次一密”密码启发而来。
- 流密码目前的理论已经比较成熟，工程实现也比较容易，加密效率高，在许多重要领域得到应用。
- “一次一密”密码使用的密钥是和明文一样长的随机序列，密钥越长越安全，但长密钥的存储、分配都很困难。

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12

n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

明文为attack begins at five，密钥为cipher，

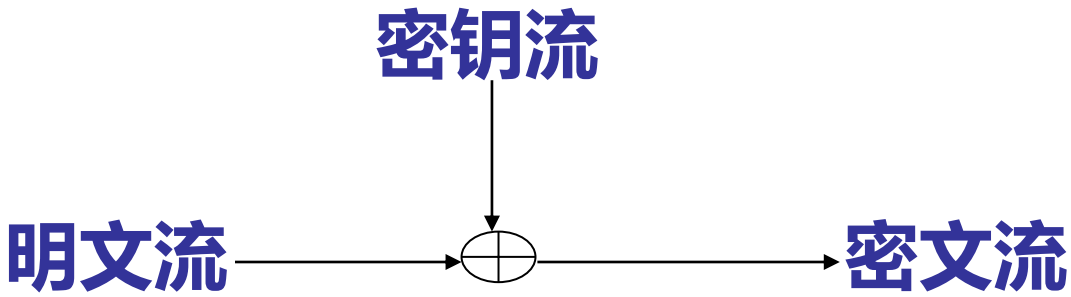
attack begins at five　明文

+ cipher cipher ci pher　密钥

= cbihgb dmvprj cb upzv　密文

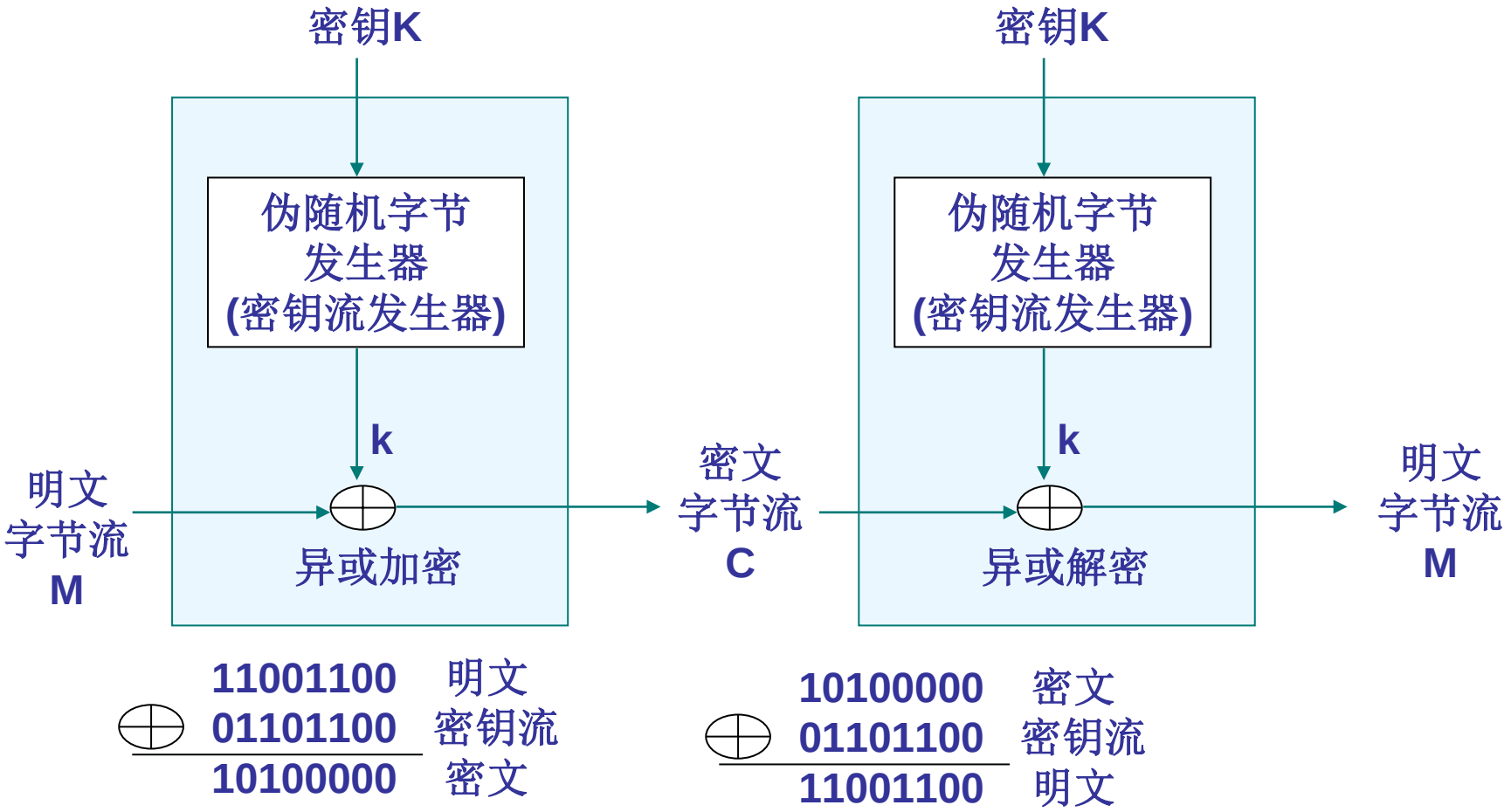
去除空格后为cbihgbdmvprjcbupzv

序列密码采用密钥生成器，从原始密钥生成一系列密钥流用来加密信息，每个明文可以选用不同的密钥加密。如果序列密码所使用的是真正随机产生的、与消息流长度相同的二进制序列，此时的序列密码就是“一次一密”的密码体制，这种密码的破解很困难。



- 序列密码的关键就是产生密钥流的算法，该算法必须能够产生**可变长的、随机的、不可预测的**密钥流。
- 保持通信双方的精确同步是序列密码实际应用中的关键技术。由于通信双方必须能够产生相同的密钥流，所以这种密钥流不可能是真随机序列，只能是**伪随机流**。

- 典型的序列密码每次加密一位或一个字节明文。
- 将初始密钥(种子)输入到发生器，输出一个随机数(密钥)。



- 密钥流的周期要长。伪随机数发生器产生的并非完全随机的序列，它是一个产生确定的比特流的函数，该比特流最终将产生重复。重复的周期越长，相当于密钥越长，密码分析也就越困难。
- 密钥流应尽可能地接近于一个真正的随机数流的特征。例如1和0的个数应大致相同。密钥流越随机，加密所得的密文也越随机，分析就越困难。
- 伪随机数发生器的输出取决于输入的密钥的值。

- RC4是Ron Rivest为RSA公司在1987年设计的一种流密码。
- 它是一种可变密钥长度、面向字节操作的流密码。
- RC4可能是应用最广泛的流密码

  用于SSL/TLS(安全套接字/传输层安全协议)

  用于IEEE802.1无线局域网中的WEP协议。

3.5 其他对称密码---RC4流密码

- **IDEA**

  由旅居瑞士的华人来学嘉和他的导师J.L. Massey共同开发的。IDEA使用128位密钥，明文和密文分组长度为64位。已被用在多种商业产品中。

- **CLIPPER密码**

  采用SKIPJACK算法，明文和密文分组长度为64位，密钥长度为80位。

- **Blowfish**

  Blowfish允许使用最长为448位的不同长度的密钥，并针对在32位处理器上的执行进行了优化。

- **Twofish**

  Twofish使用128位分组，可以使用128、192或256位密钥。

- **CAST-128**

  CAST-128使用128位密钥。它在更新版本的PGP中使用。

- **GOST**

  GOST是为了回应DES而开发的俄罗斯标准，它使用256位密钥。

分组密码在加密时明文分组的长度是固定的。而实用中待加密消息的数据量是不定的，数据格式也可能是多种多样的，为了能在不同应用场合安全地使用分组密码，通常对不同的使用目的运用不同的工作模式。所谓分组密码的工作模式就是以该分组密码为基础构造的一个密码系统。目前已提出许多种分组密码的工作模式：

3.6 分组密码的工作模式

模式	描述	典型应用
电码本（ECB)		
密码分组链接（CBC)		
密码反馈（CFB)		
输出反馈（OFB)		
计数器（CTR)		

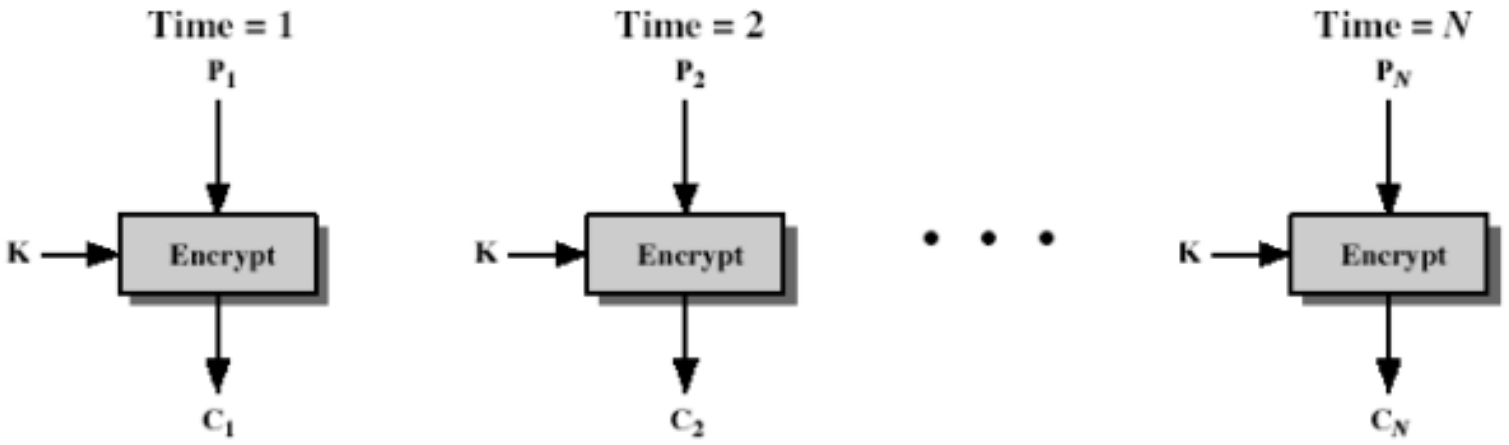
Electronic Code Book,ECB

工作模式：

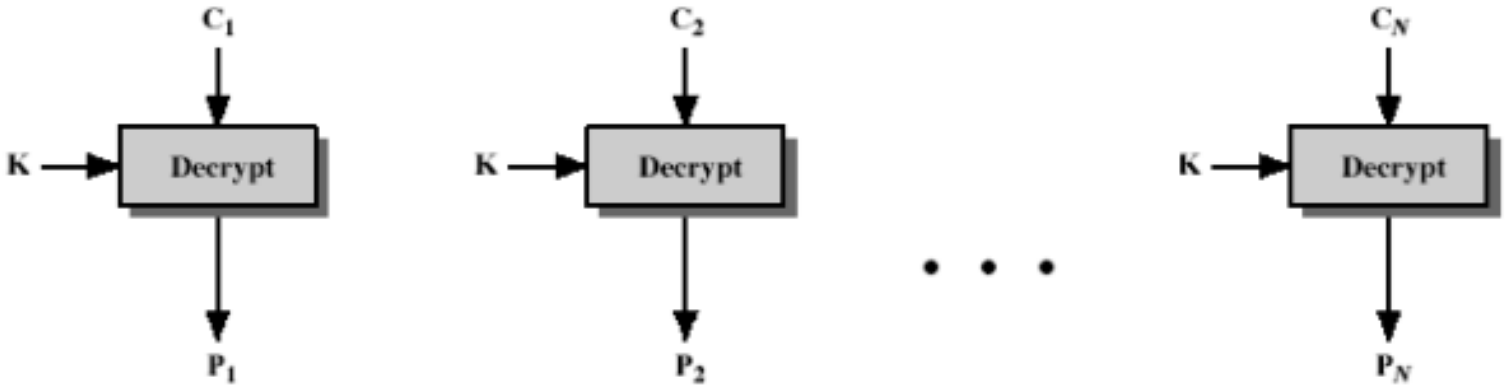
1：加密 $C_j = E_k(P_j), (j = 1, 2, \ldots, N)$

2：解密 $P_j = D_k(C_j)$

3.6 分组密码的工作模式---电码本模式



(a) Encryption



(b) Decryption

应用：短数据

错误传播：不传播，即某个C_t的传输错误只影响到的P_j恢复，不影响后续的P_j

Cipher Block Chaining,CBC

工作模式：

1.加密 令$C_0 = IV$（初始向量），则$C_j = E_k(C_{j-1} \oplus P_j)$, $j = 1, 2, \ldots, N$

2.解密 $P_j = C_{j-1} \oplus D_k(C_j)$

应用：

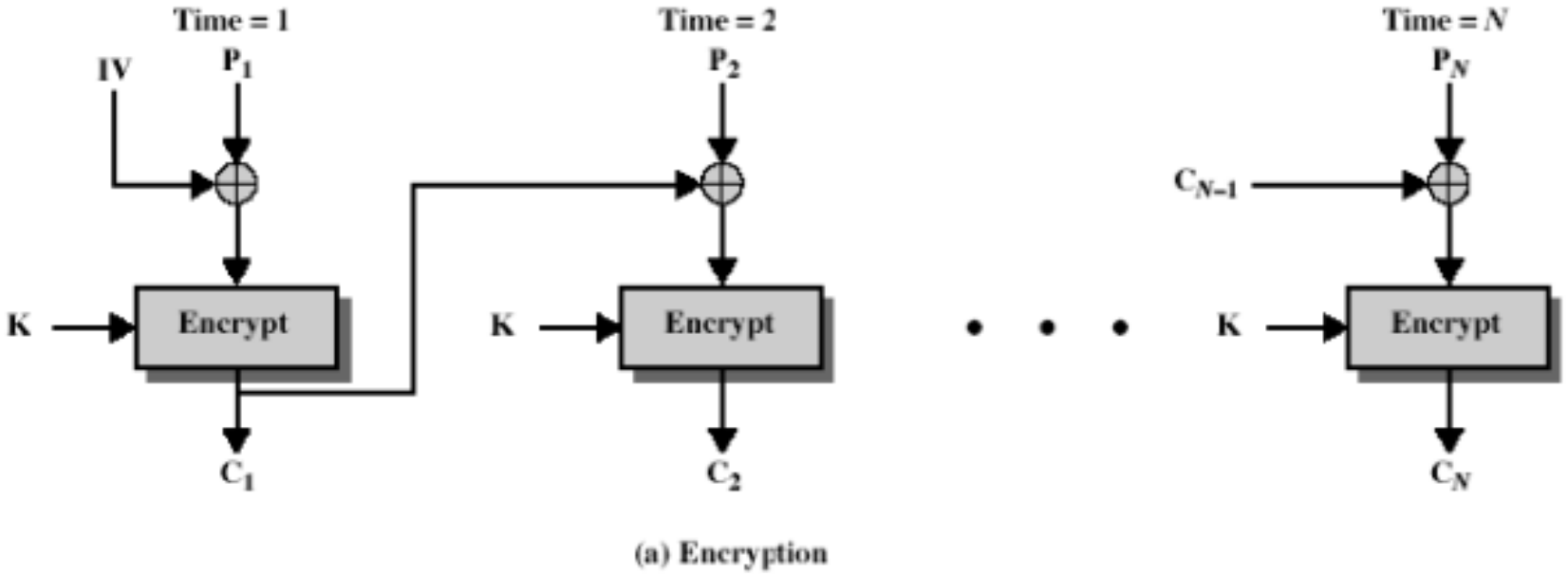
1.加密长度大于64位的明文P.

2.认证。

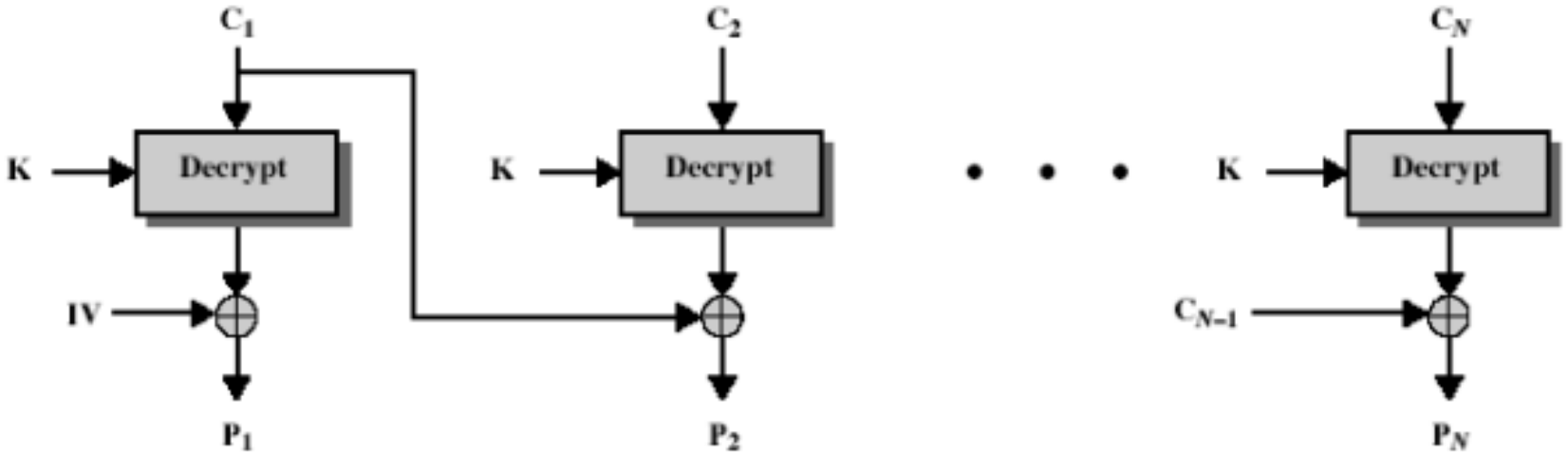
错误传播：传播，即错误的C_t会影响到P_t和P_{t+1}。因为：

$$P_t = C_{t-1} \oplus D_k(C_t), P_{t+1} = C_t \oplus D_k(C_{t+1})$$

3.6 分组密码的工作模式---密码分组链接模式



(a) Encryption



(b) Decryption

Cipher FeedBack,CFB

工作模式：

设明文$P=(P_1P_2\dots P_N)$，密文$C=(C_1C_2\dots C_N)$，P_j,C_j均为s位。

1.加密：$C_j=P_j\oplus S_s(E_k(C_{j-1},C_{j-2},\dots,C_{j-r}))$，

$j=1,2,\dots,N$。其中初值为$(C_{-j+1},C_{-j+2},\dots,C_0)=IV$（初始向量），

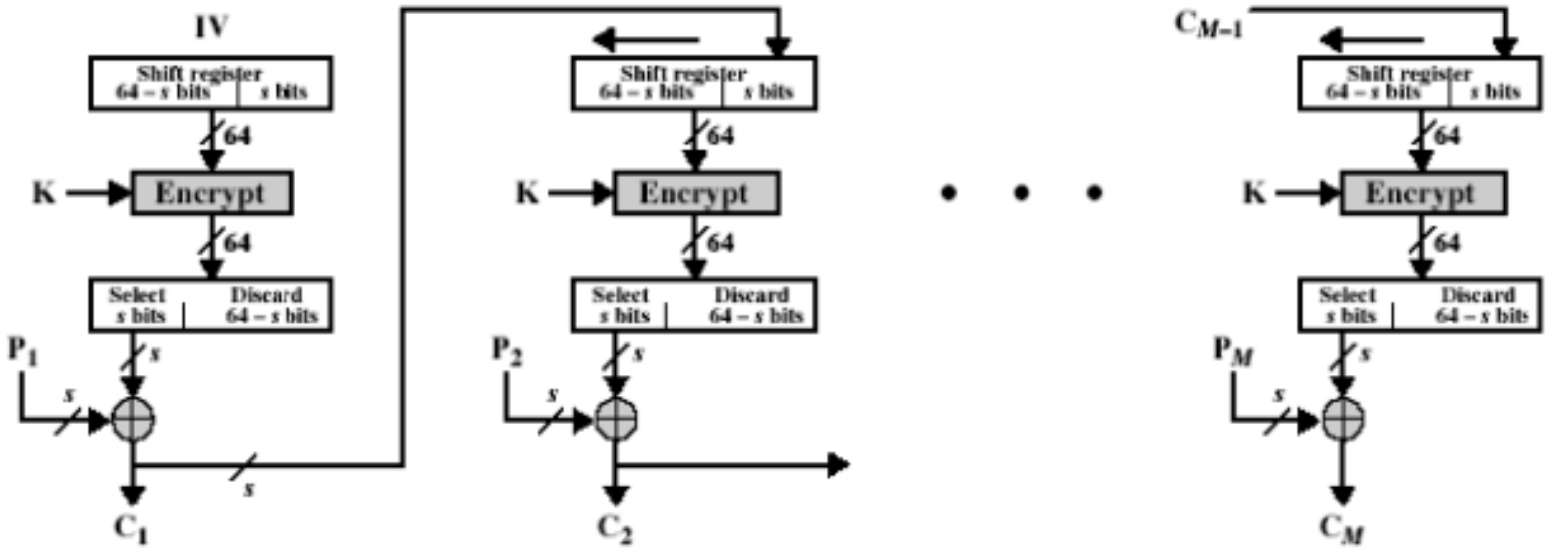
2.解密：$P_j=C_j\oplus S_s(E_k(C_{j-1},C_{j-2},\dots,C_{j-r}))$

应用：

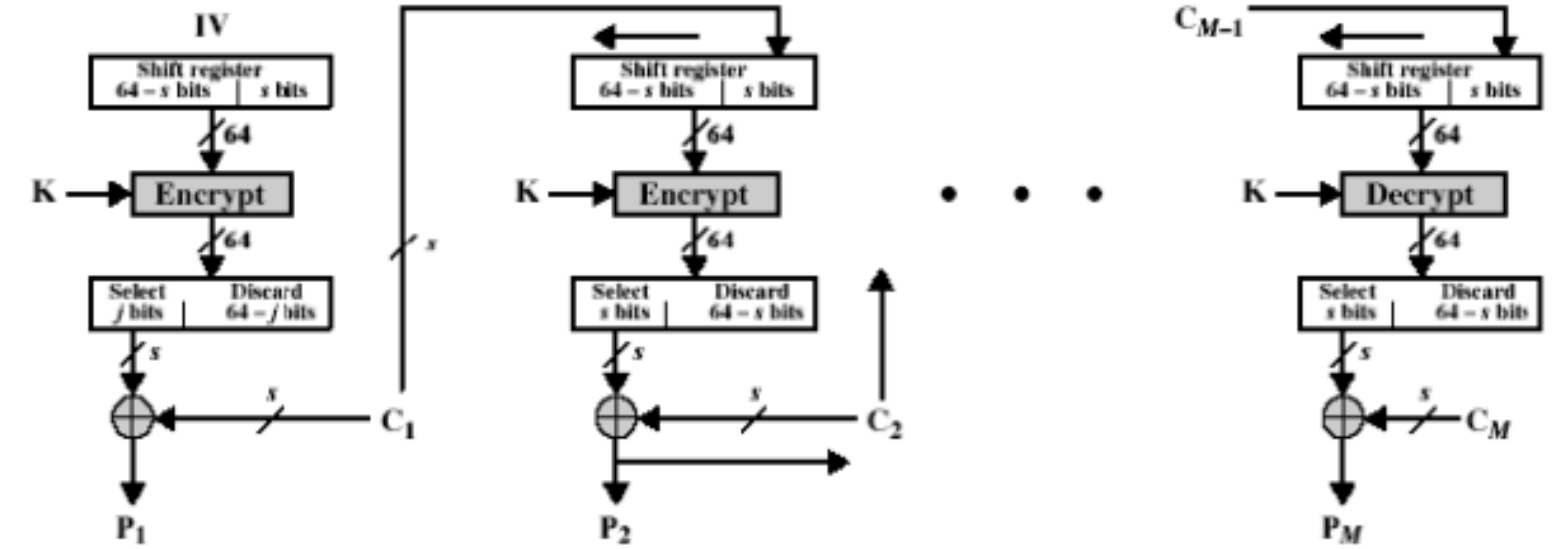
1.保密：加密长度大于64位的明文P，分组随意。

2.认证。

3.6 分组密码的工作模式---密码反馈模式



(a) Encryption



(b) Decryption

Cipher FeedBack,CFB

特点：分组任意，安全性优于ECB模式，加密、解密都使用加密运算。

错误传播：传播

Output FeedBack,OFB

工作模式：

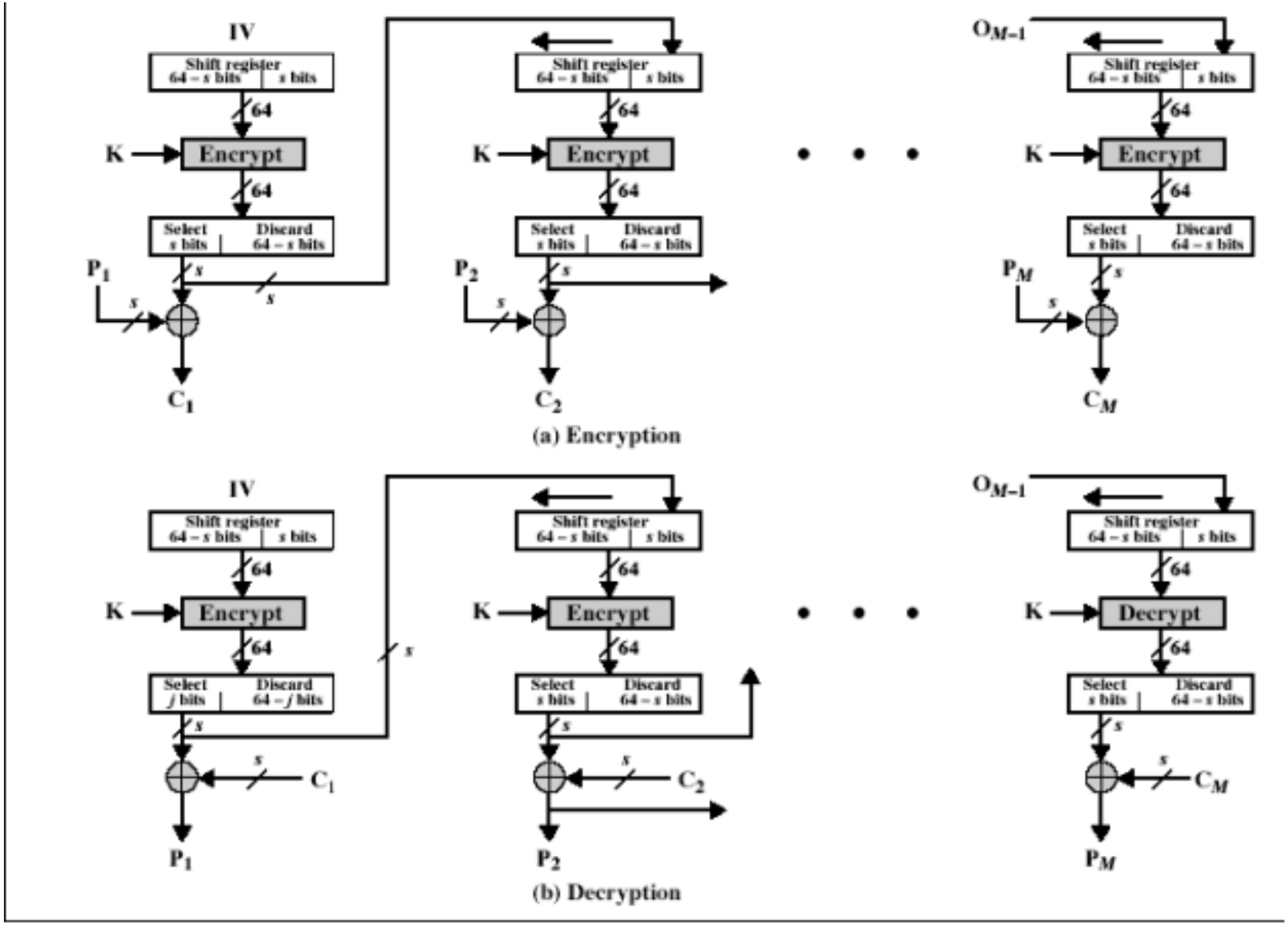
1. 加密　令 $IV_0=IV$（初始向量），则 $C_j = P_j \oplus S_s(E_k(IV_{j-1}))$,j=1,2,…,N.其中$P_j$,$C_j$均为s位。

2.解密 $P_j = C_j \oplus S_s(E_k(IV_{j-1}))$

特点：抗消息流篡改攻击的能力不如CFB。

错误传播：不传播

3.6 分组密码的工作模式---输出反馈模式



工作模式：

1.加密：给定计数器初值IV，则$C_j = P_j \oplus E_k(IV + j - 1)$，

$j = 1, 2, \ldots, N$。

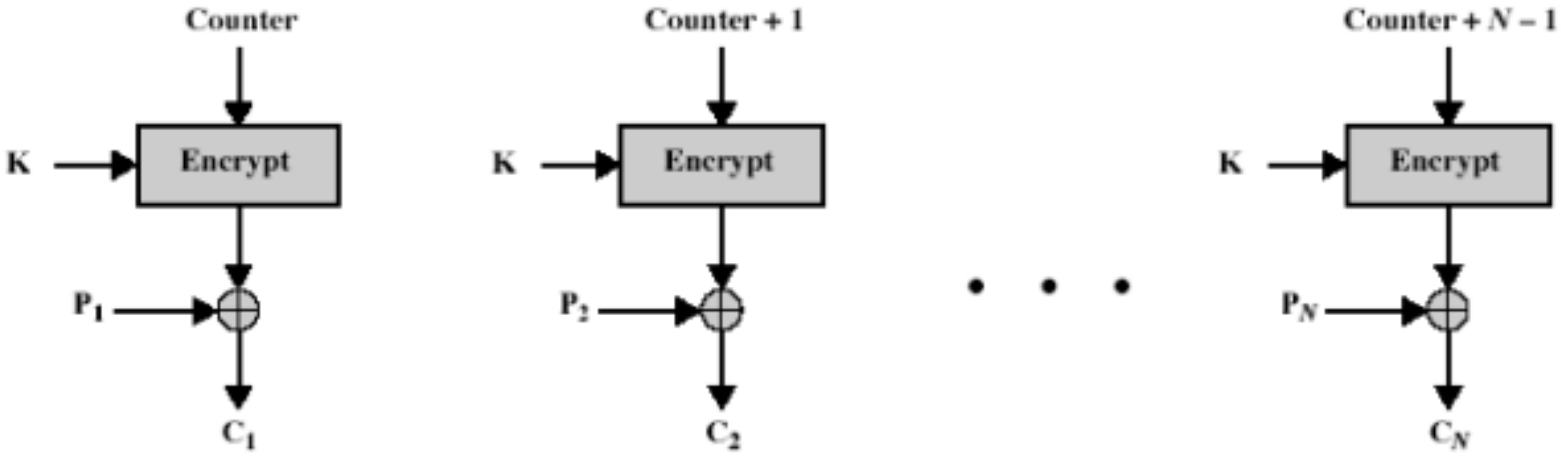
2.解密:$P_j = C_j \oplus E_k(IV + j - 1)$，

优点：

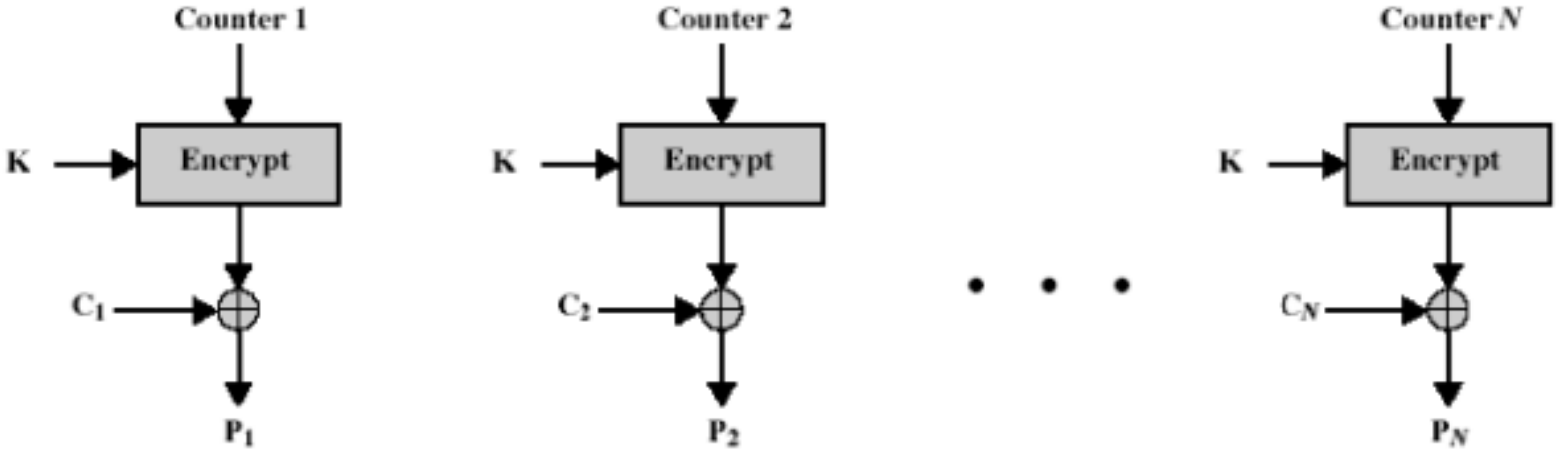
硬件效率：可并行处理；
软件效率：并行化；
预处理
随机访问：比链接模式好；
可证明的安全性：至少与其他模式一样；
简单性：只用到加密算法。

错误不传播

3.6 分组密码的工作模式---计数器模式CTR



(a) Encryption



(b) Decryption

Thank you!

网络攻击是实现“不战而屈人之兵”最有效的武器之一

没有网络信息安全就没有国家安全