第三章 分组密码与数据加密标准

- **DES(Data Encryption Standard)**
- **Triple DES**
- **IDEA**
- **Blowfish**
- **RC5**
- **CAST-128**
- **……**

DES的产生-i

- 1973年5月15日,美国国家标准局 NBS(National Bureau of Standards)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-ii

- 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

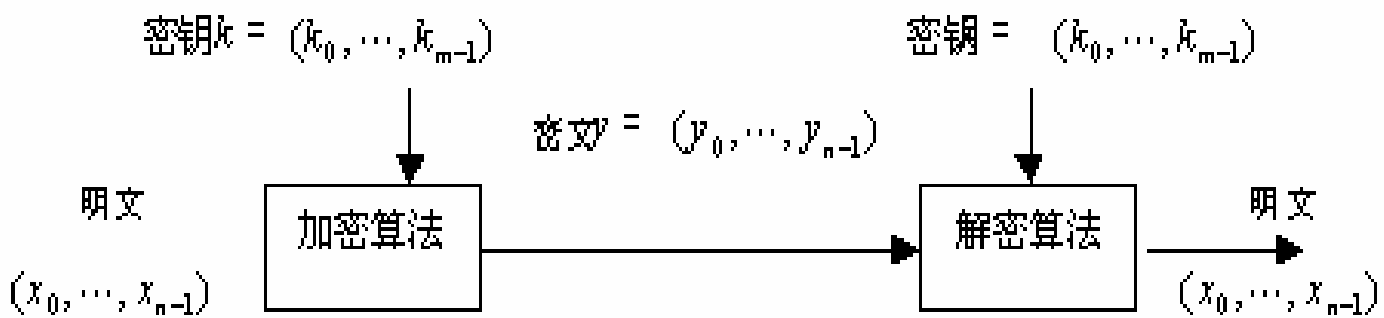
DES的应用

- 1979年，美国银行协会批准使用
- 1980年，美国国家标准协会ANSI（American National Standards Institute)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，决定1998年12月以后，DES将不再作为联邦加密标准。

分组密码的一般设计原理

- 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，



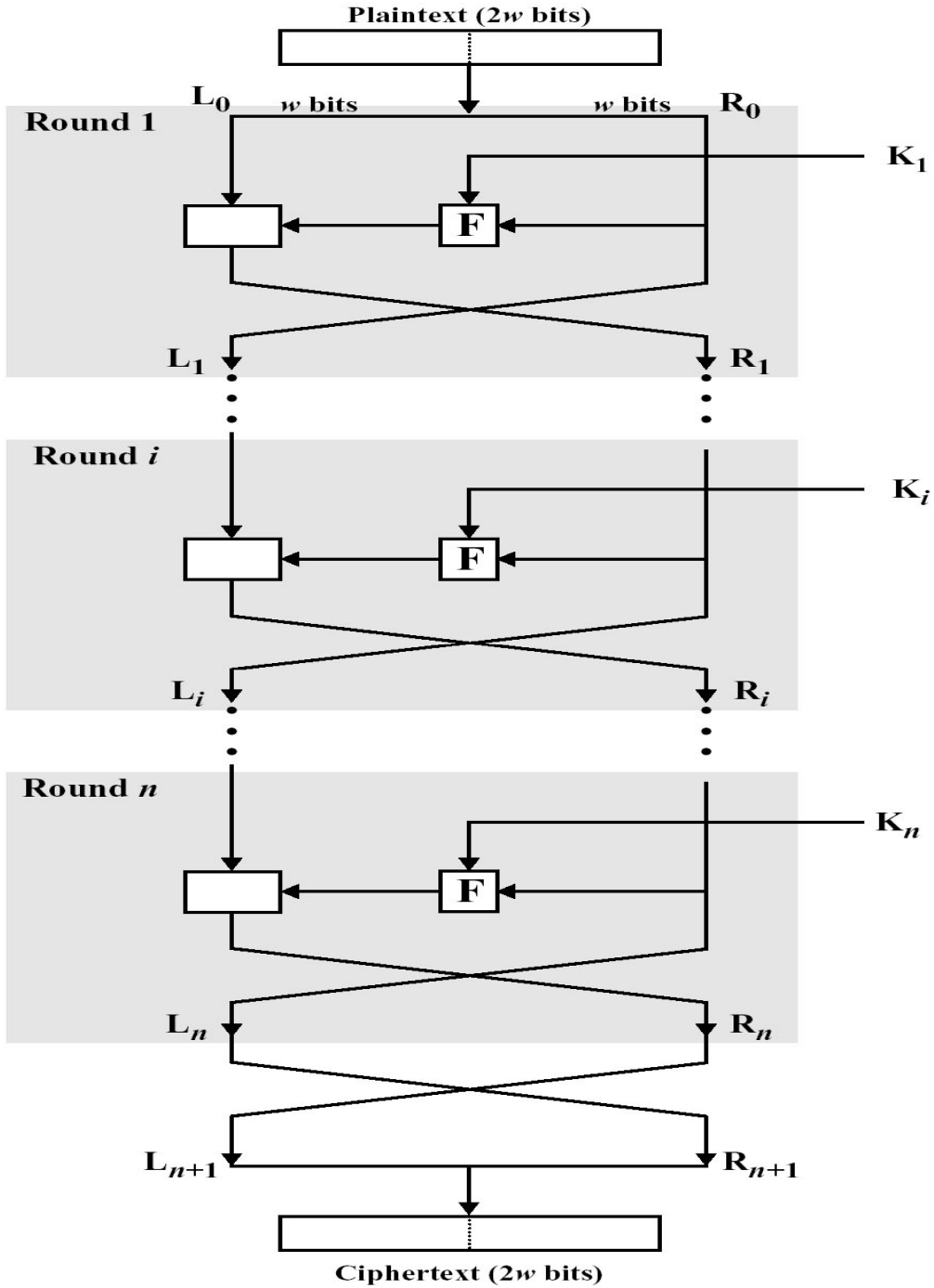
分组密码模型

- 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
- 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

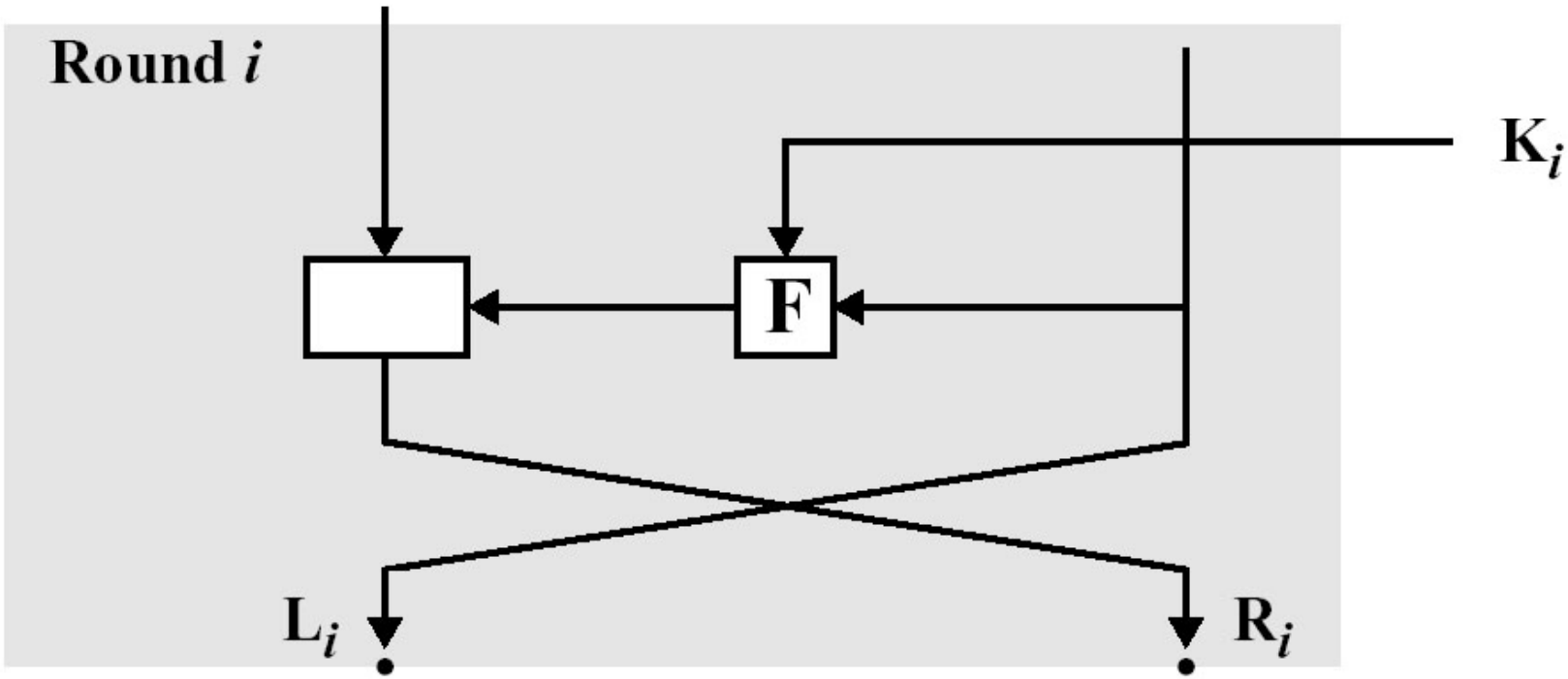
实现的设计原则

- 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。

Feistel 结构图


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结构定义



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

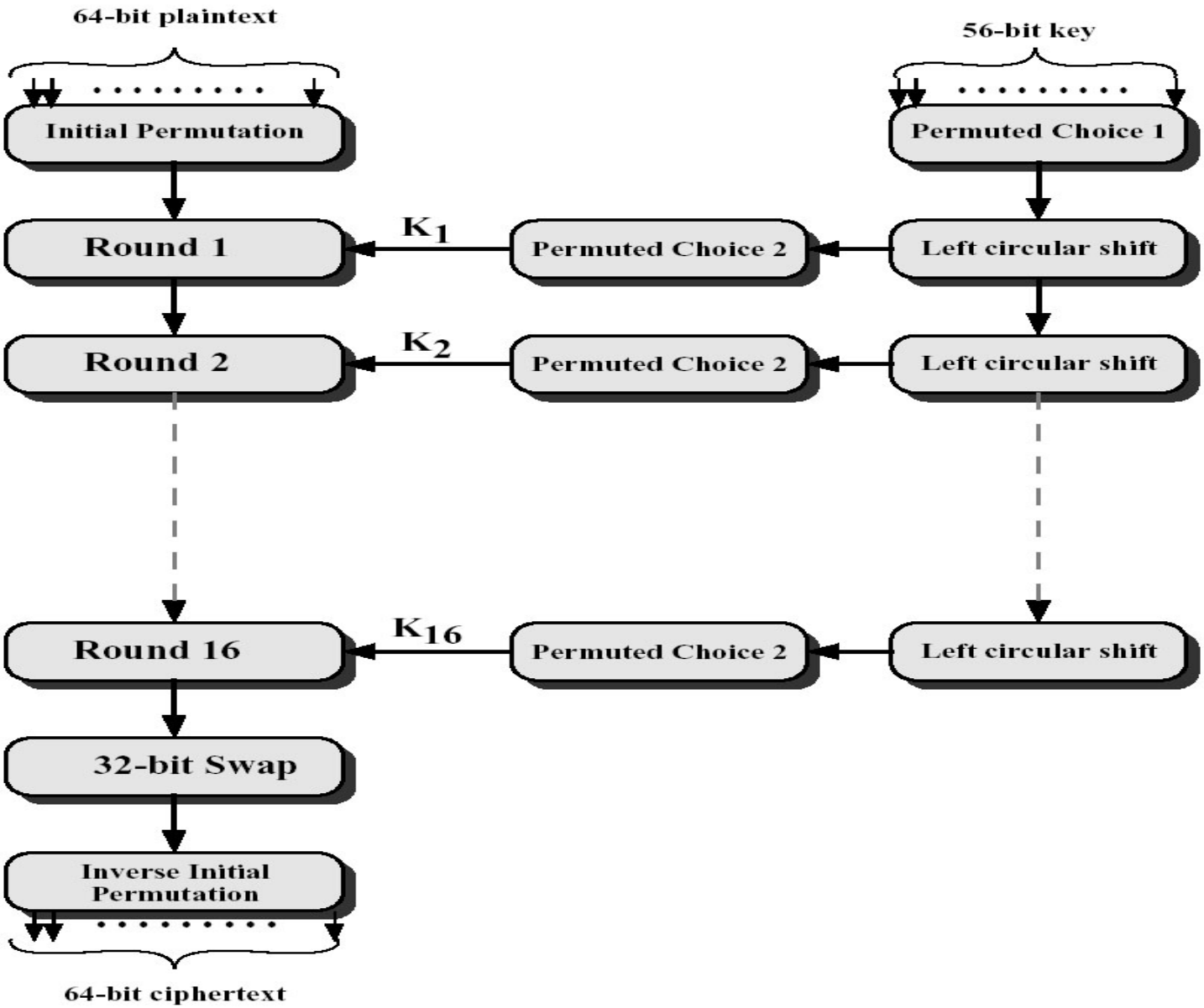
$$L_{i-1} = R_i \oplus F(R_{i-1}, K_i)$$
$$= R_i \oplus F(L_i, K_i)$$

Feistel结构分析

- Block size(64 → 128)
- Key size(56 → 128~256)
- Number of rounds(16)
- Subkey generation
- Round function(F)

- Fast software encryption/decryption
- Easy hardware implementation
- Simple structure
- Ease of analysis

DES示意图


64-bit plaintext
56-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

DES的描述

- DES利用56比特串长度的密钥K来加密长度为64位的明文，得到长度为64位的密文
- 该算法分三个阶段实现：

1. 给定明文X，通过一个固定的初始置换IP来排列X中的位，得到X_0。

$X_0=IP（X）=L_0R_0$

其中L_0由X_0前32位组成，R_0由X_0的后32位组成。

2.计算函数F的16次迭代, 根据下述规则来计算$L_iR_i(1<=i<=16)$

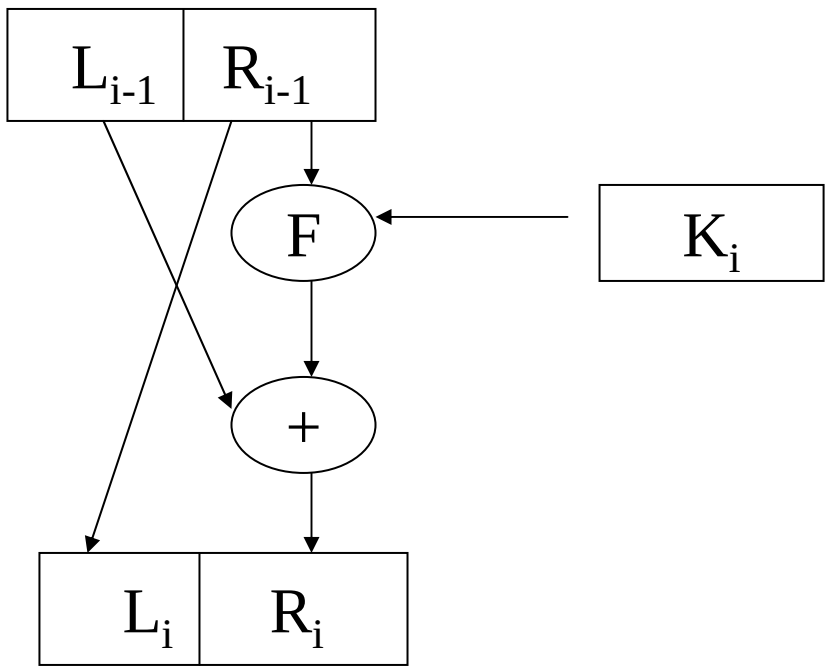
$Li=R_{i-1}, Ri=L_{i-1} \oplus F(R_{i-1}, Ki)$

其中Ki是长为48位的子密钥。子密钥K_1，K_2，…，K_{16}是作为密钥K（56位）的函数而计算出的。

3. 对比特串$R_{16}L_{16}$使用逆置换IP^{-1}得到密文Y。

$Y=IP^{-1}（R_{16}L_{16}）$

一轮加密的简图



DES加解密过程

令***i***表示迭代次数，⊕表示逐位模**2**求和，***f***为加密函数。**DES**的加密和解密过程表示如下。

加密过程：

$$L_0R_0 \leftarrow IP(<64bit\text{输入码}>)$$

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

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

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

解密过程：

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

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

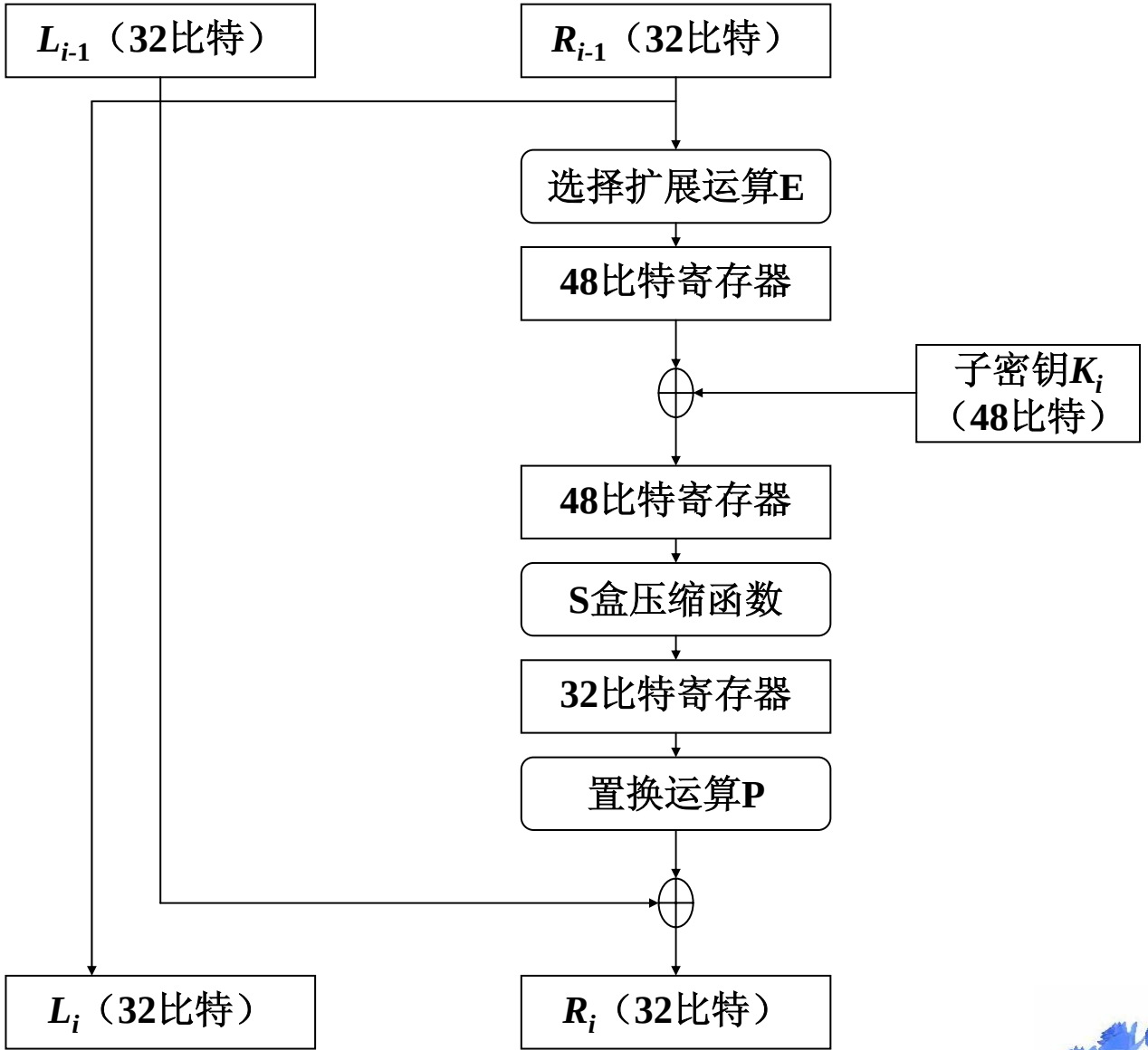
$$L_i \leftarrow R_{i-1} \oplus f(R_{i-1},k_i) \qquad i=16,15,\cdots,1$$

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

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

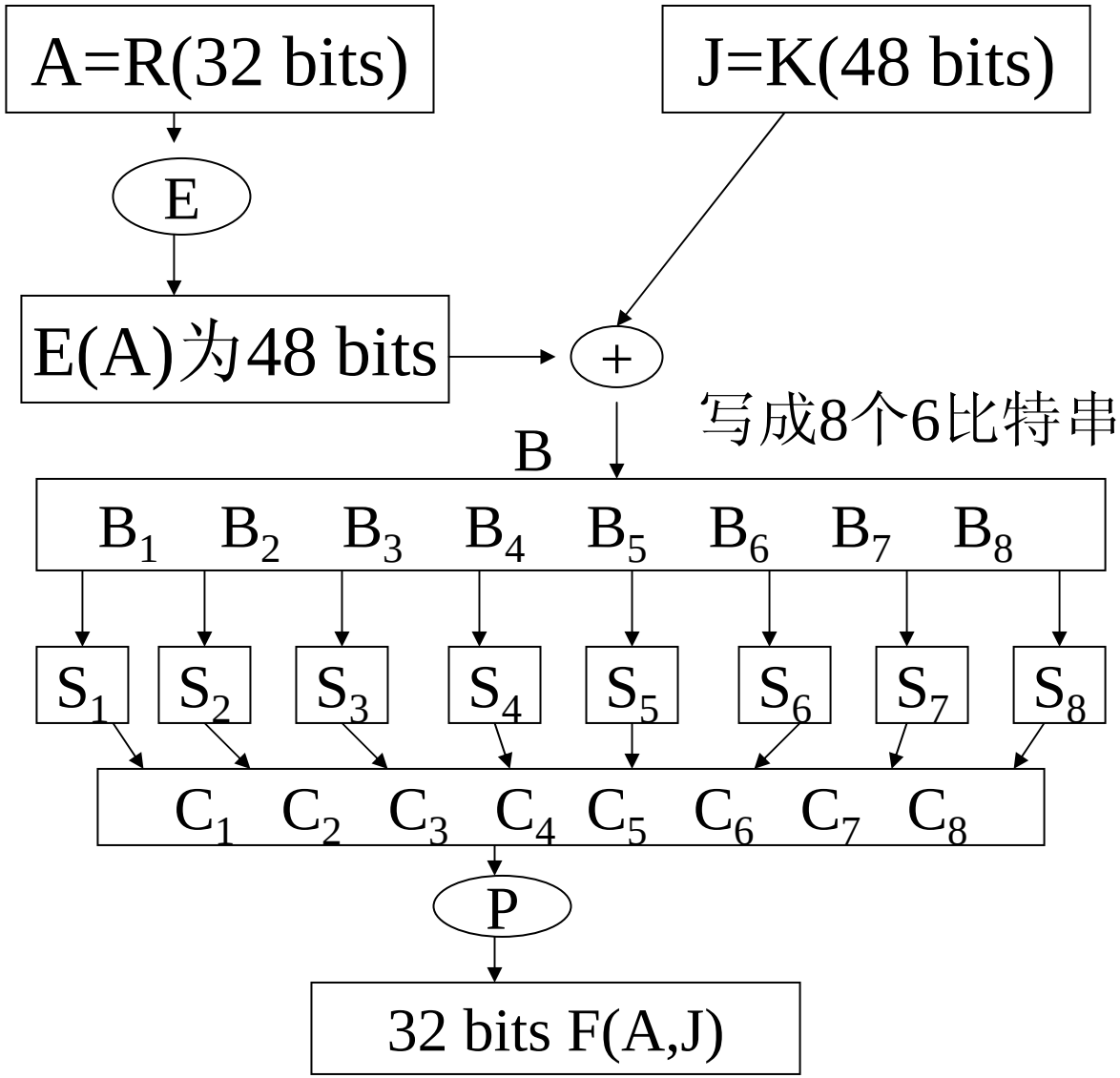


$L_i=R_{i-1}$

DES中的F函数图示

选择扩展运算

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


A=R(32 bits)
J=K(48 bits)
E
E(A)为48 bits
+
写成8个6比特串
B
B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8
S_1 S_2 S_3 S_4 S_5 S_6 S_7 S_8
C_1 C_2 C_3 C_4 C_5 C_6 C_7 C_8
P
32 bits F(A,J)

DES 的F函数

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

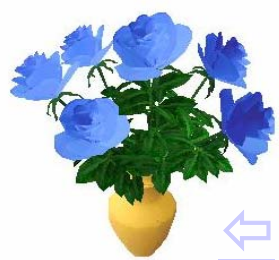
S_5	0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6	0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	3	4	3	2	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

S-Box

- 对每个盒，6比特输入中的第1和第6比特组成的二进制数确定的行，中间4位二进制数用来确定的列。　中相应行、列位置的十进制数4位二进制数表示作为输出。例如的输入为101001，则行数和列数的二进制表示分别是11和0100，即第3行和第4列，S2的第3行和第4列的十进制数为3，用4位二进制数表示为0011，所以的输出为0011。

Permutation

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



从密钥K计算子密钥

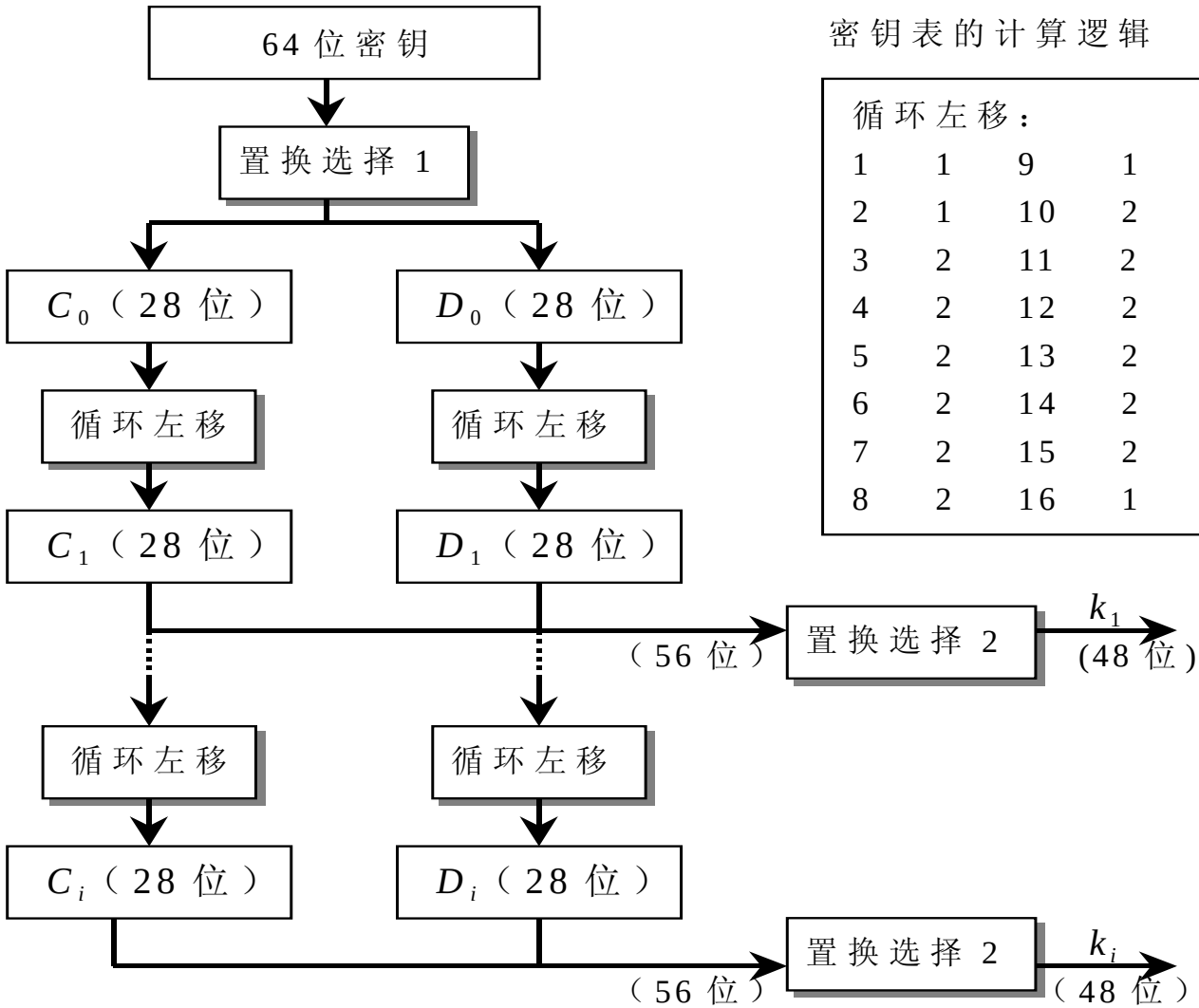
实际上，**K**是长度为**64**的位串，其中**56**位是密钥，**8**位是奇偶校验位（为了检错），在密钥编排的计算中，这些校验位可略去。

给定**64**位的密钥**K**，放弃奇偶校验位（**8**，**16**，…，**64**）并根据固定置换**PC-1**,来排列**K**中剩下的位。我们写

PC-1(K)=C_0D_0

其中C_0由**PC-1**（**K**）的前**28**位组成；D_0由后**28**位组成。

子密钥的产生



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

置换选择1（PC-1）和置换选择2（PC-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

- F函数(S-Box)设计原理未知
- 密钥长度的争论
- DES的破译
- 弱密钥与半弱密钥

S-Box问题

- **1976年美国NSA提出了下列几条S盒的设计准则：**
  1. S盒的每一行是整数0，…，15的一个置换
  2. 没有一个S盒是它输入变量的线性函数
  3.改变S盒的一个输入位至少要引起两位的输出改变
  4.对任何一个S盒和任何一个输入X，S（X）和S(X⊕001100）至少有两个比特不同（这里X是长度为6的比特串）
  5.对任何一个S盒，对任何一个输入对e,f属于{0,1}, S(X)≠ S(X⊕11ef00)
- **6.** 对任何一个S盒，如果固定一个输入比特，来看一个固定输出比特的值，这个输出比特为0的输入数目将接近于这个输出比特为1的输入数目。

密钥长度

- 关于**DES**算法的另一个最有争议的问题就是担心实际**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 $2^{56} \approx 10^{17}$ 个
- 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。

- 在CRYPTO’93上，Session和Wiener给出了一个非常详细的密钥搜索机器的设计方案，这个机器基于并行运算的密钥搜索芯片，所以16次加密能同时完成。此芯片每秒能测试5000万个密钥，用5760个芯片组成的系统需要花费10万美元，它平均用1.5天左右就可找到DES密钥。
- 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的破译

- 时间-存储权衡(Time-memory trade-off)分析方法
- 差分分析(Differential Cryptanalysis)方法
- 线性分析(Linear Cryptanalysis)分析方法

时间-存储权衡分析方法

- 选择明文攻击
- 不依赖于DES的“结构”，只与DES的输入、输出长度和密钥长度有关
- 是一种混合方法，在选择明文攻击中以时间换取空间，时间复杂度 $T = O(2^{112/3})$ 空间复杂度为 $S = O(2^{112/3})$

差分密码分析

- 选择明文攻击
- 基本思想是通过分析特定明文差对结果密文差的影响来获得可能性最大的密钥。
- 主要适用于攻击迭代密码体制
- 虽然对破译16-轮DES效果并不好，但用来破译轮数低的DES则非常成功。例如：8-轮DES在普通PC机上几分钟即可破译成功

线性分析方法

- 已知明文攻击
- 基本思想是通过寻找一个给定密码算法的有效的线性近似表达式来降低密钥的熵，从而破译密码系统
- 可用 2^{21} 个已知明文破译8-轮DES
- 可用 2^{47} 个已知明文破译16-轮DES

弱密钥与半弱密钥

- 弱密钥：$E_K \bullet E_K = I$，DES存在4个弱密钥
- 半弱密钥：$E_{K1} = E_{K2}$，至少有12个半弱密钥