信息安全实验第一部分——密码学实验

实验 2 对称密码学实验

Data Encryption Standard

北京邮电大学信息安全中心　　周亚建

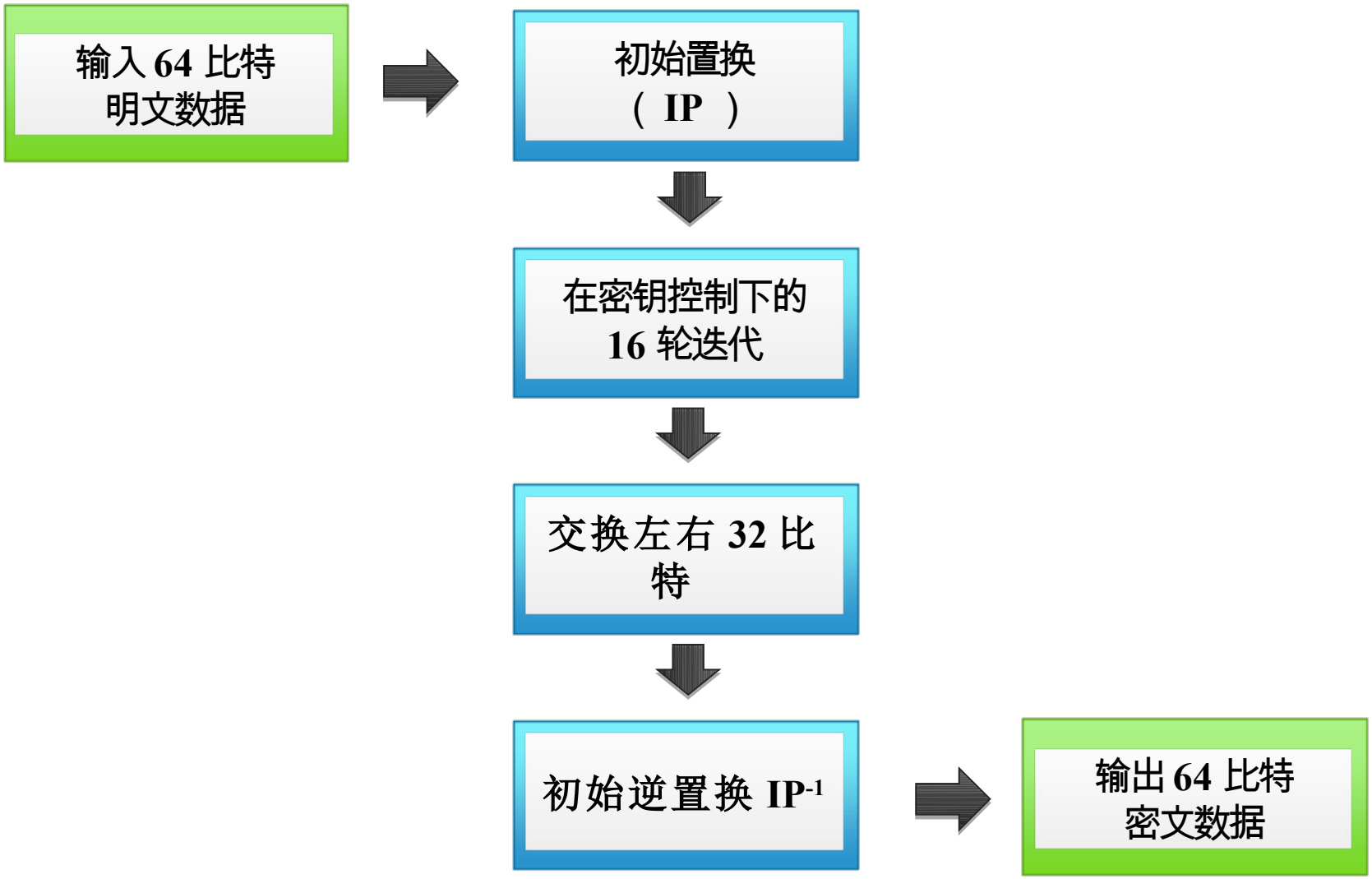
zhouyajian@gmail.com

对称密码学实验教学目的

深入理解 DES 算法的原理

掌握 DES 算法的实现方法

DES 加密过程


输入 64 比特
明文数据
初始置换
（ IP ）
在密钥控制下的
16 轮迭代
交换左右 32 比
特
初始逆置换 IP-1
输出 64 比特
密文数据

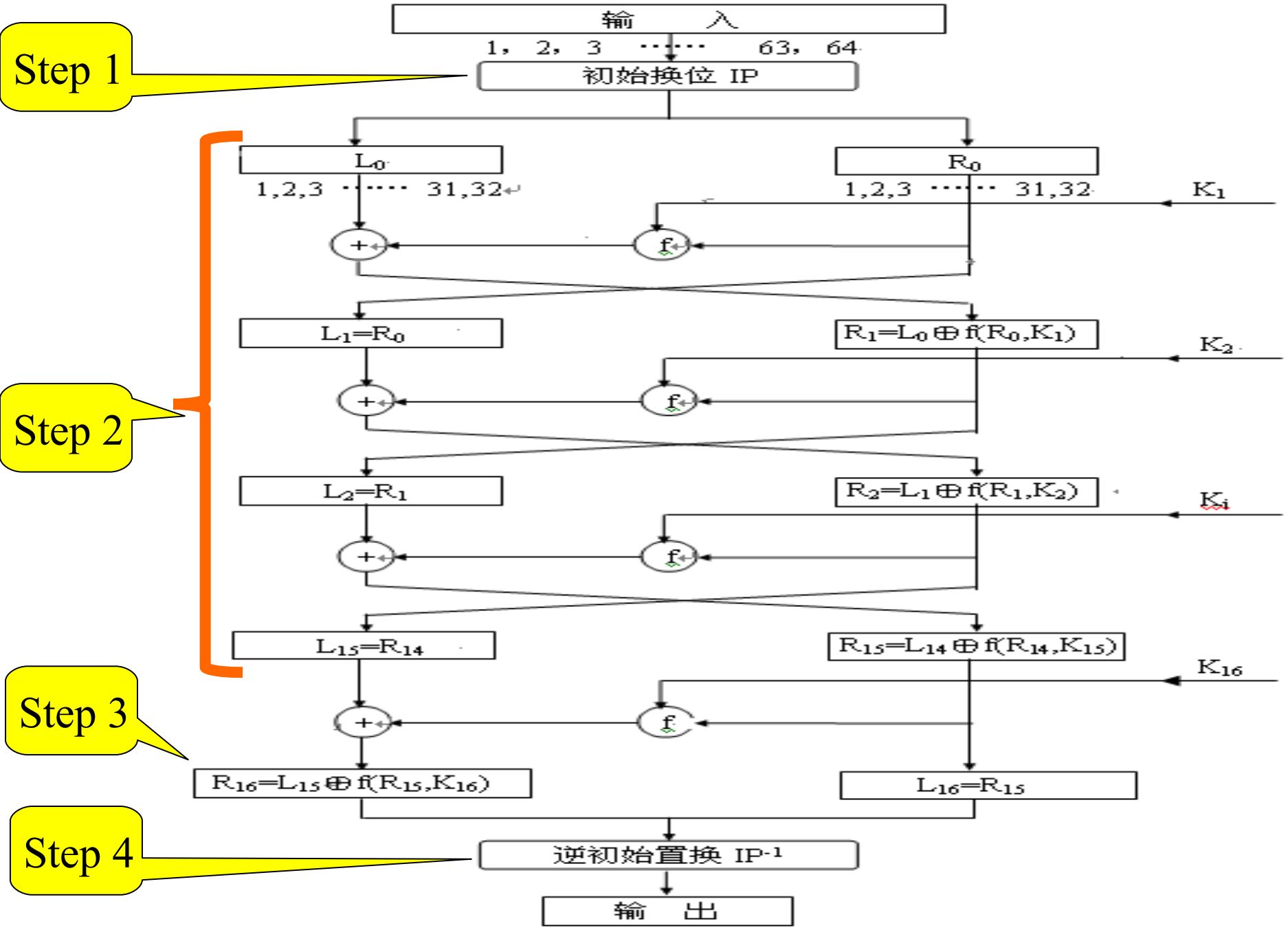


图 1-1 DES 加密/解密流程

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

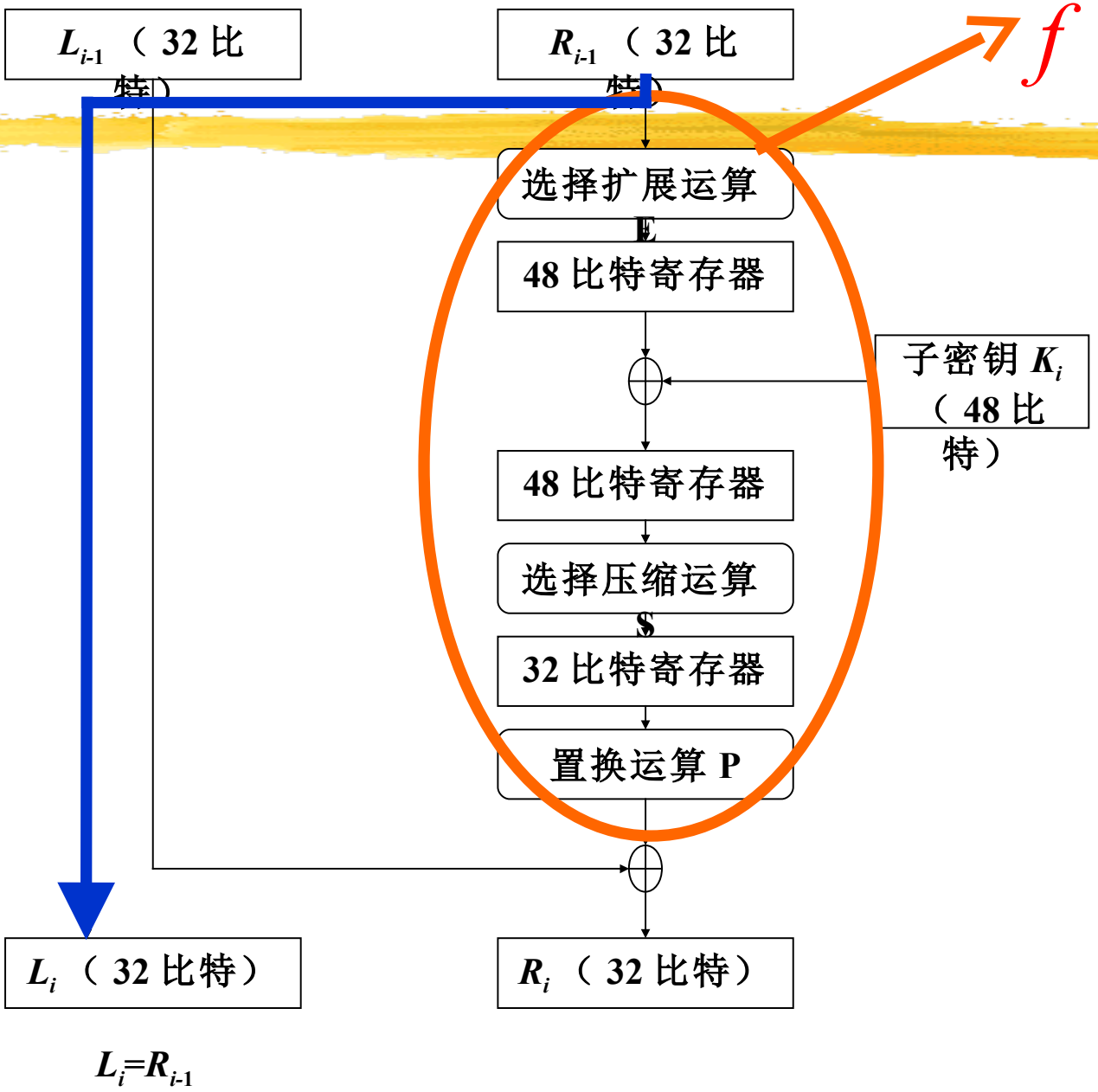
初始置换 IP 和初始逆置换 IP^{-1}

```
const static char Initial_Permutation[64] = {
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};

void Initial_Permutation(char *Message, bool*Left, bool*Right)
{
bool temp[64];
int i, j;
// 把 Message 转化成 bit 形式
for(i=0; i<8; i++)
for(j=0; j<8; j++)
temp[i*8+j] = (Message[i]>>j)&0x01;
for(i=0; i<32; i++)  Left[i]=temp[Initial_Permutation[i]-1];
for(; i<64; i++)       Right[i-32]=temp[Initial_Permutation[i]-1];
}
```

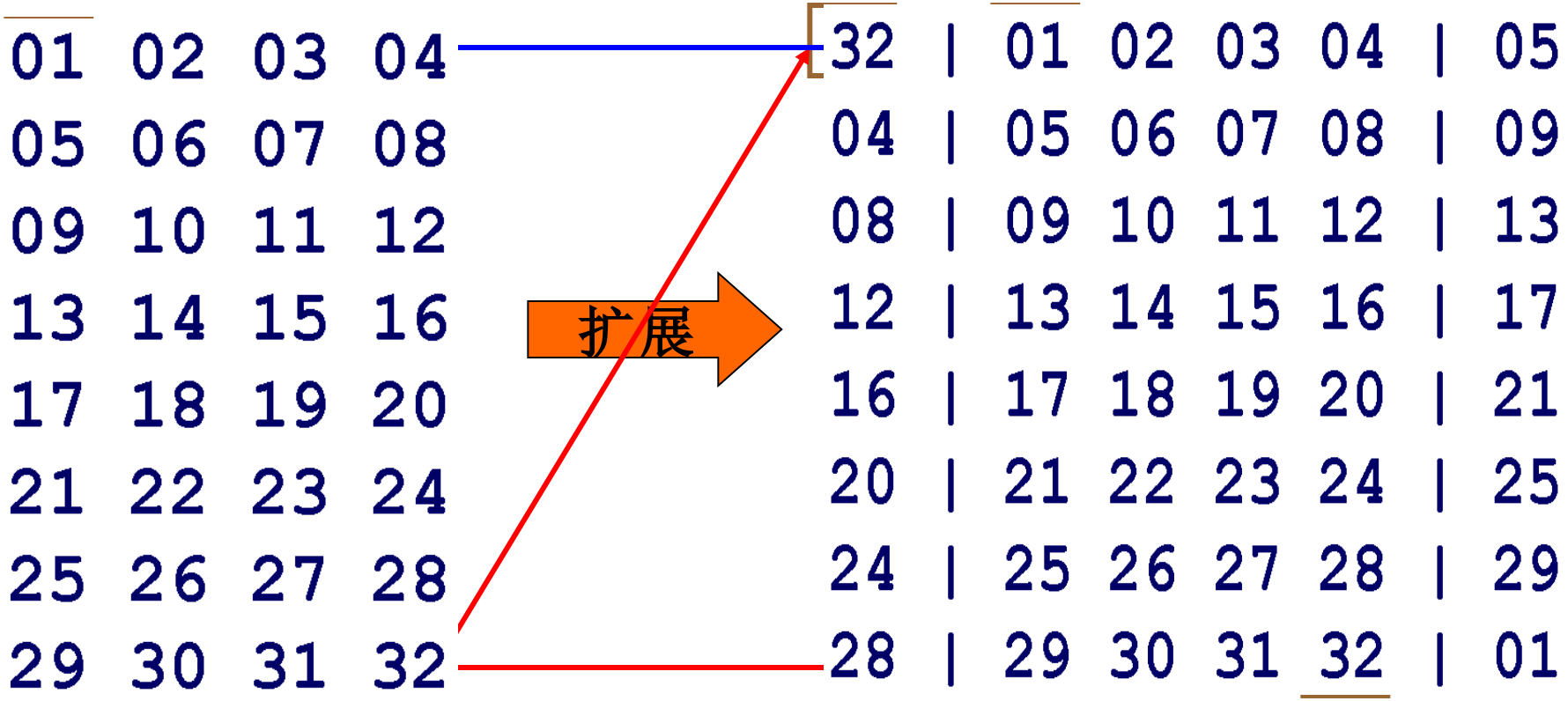
初始置换 IP 和初始逆置换 IP^{-1}

```
unsigned char Final[] =
{
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 };
void Final_ Permutation (bool*left , bool*right , char*result )
{
memset(result, 0, 8);
bool temp1[64];
//left 和 right 组合成 temp1
memcpy(temp1, left, 32);
memcpy(temp1+32, right, 32);
bool temp2[64];
for(int i=0; i<64; i++) { // 由末置换得到temp2
temp2[i]=temp1[Final[i]-1];
// 将temp2 转化成char 型
result[i/8] |= temp2[i]<<(i%8); }
}
```

DES 的一轮迭代


L_{i-1}（32 比特）
R_{i-1}（32 比特）
f
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）
$L_i=R_{i-1}$

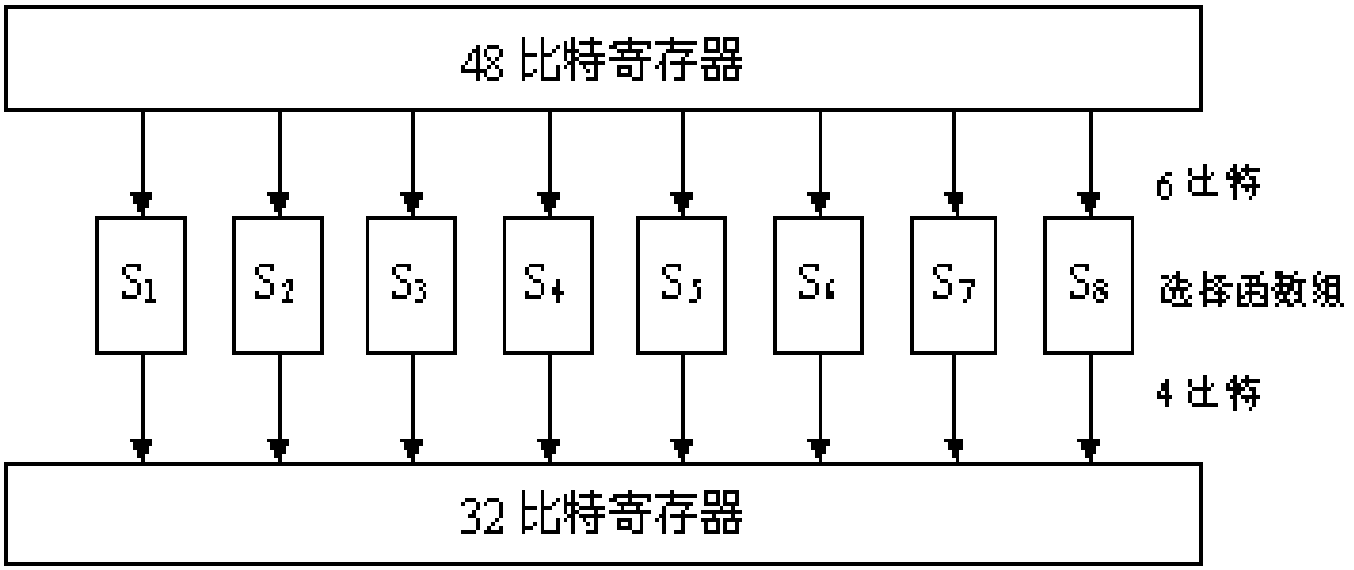
扩展置换 E - 盒——32 位扩展到 48 位



扩展置换 E - 盒——32 位扩展到 48 位

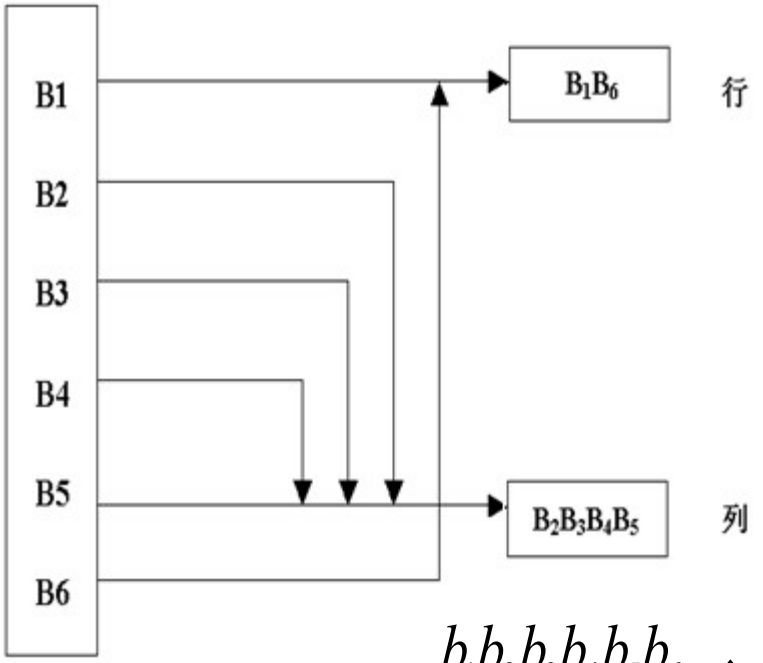
```
unsigned char Expansion[] =
{
32, 1, 2, 3, 4, 5, 4, 5, 6, 7, 8, 9, 8, 9, 10, 11, 12, 13, 12, 13, 14, 15, 16, 17,
16, 17, 18, 19, 20, 21, 20, 21, 22, 23, 24, 25, 24, 25, 26, 27, 28, 29,
28, 29, 30, 31, 32, 1
};

void Expand_Right(bool*Input , bool*Output)
{
  for(int i =0; i<48; i++)  Output[i]=Input[Expansion[i]-1];
}
```

压缩替代 S- 盒—— 48 位压缩到 32 位



选择压缩运算 S

S- 盒的构造



行\列	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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

$$\frac{b_1b_2b_3b_4b_5b_6}{110011} \Rightarrow \frac{\text{行：}\ b_1b_6 = 11_2 = 3}{\text{列：}\ b_2b_3b_4b_5 = 1001_2 = 9} \Rightarrow \frac{S_6\text{-盒子}\ \ 3\text{行}9\text{列}}{\text{值：}\ 14=1100}$$

S- 盒

```
unsigned char S_Box[8][64] =
{
/* S1 */
{ 14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7,
0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8,
4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0,
15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13},
                                ……
/* S8 */
{13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7,
1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2,
7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8,
2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11}
};
```

S- 盒

```
void S_function(bool* Input, bool*Output)
{
unsigned int x,y;
char z;
bool* in=Input;
bool* out=Output;
for(int j=0; j<8; j++, in+=6, out+=4)
{
x = (in[0]<<1) + in[5];   // 第1 位和第6 位
y = (in[1]<<3) + (in[2]<<2) + (in[3]<<1) + in[4];   //2-4 位
z = S_Box[j][x*16+y];   // 由x y 确定的数z
for(int i=0; i<4; i++)  out[i] = (z>>i) & 1;    //z 用4 个bit 表示
}
}
```

S- 盒的构造（ *continued* ）

DES 中其它算法都是线性的，而 S- 盒运算则是非线性的

S- 盒不易于分析，它提供了更好的安全性

所以 S- 盒是算法的关键所在

置换 p- 盒的构造

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

P 置换的目的是提供雪崩效应

明文或密钥的一点小的变动都引起密文的较大变化

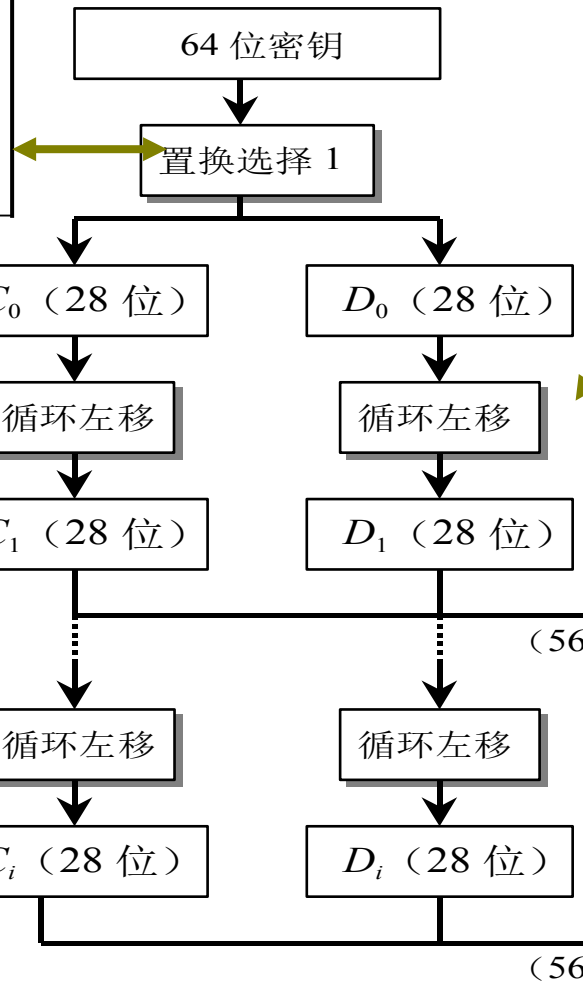
置换 p- 盒的构造

```
//P_ 盒置换
unsigned char P_Box[]=
{
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
};
void P_function(bool* a)
{
bool temp[32];
memcpy(temp,a,32);
for(int i = 0 ;i<32;i++)   a[i]=temp[P_Box[i]-1];
}
```

DES 中的子密钥的生成

PC-1

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



密钥表的计算逻辑

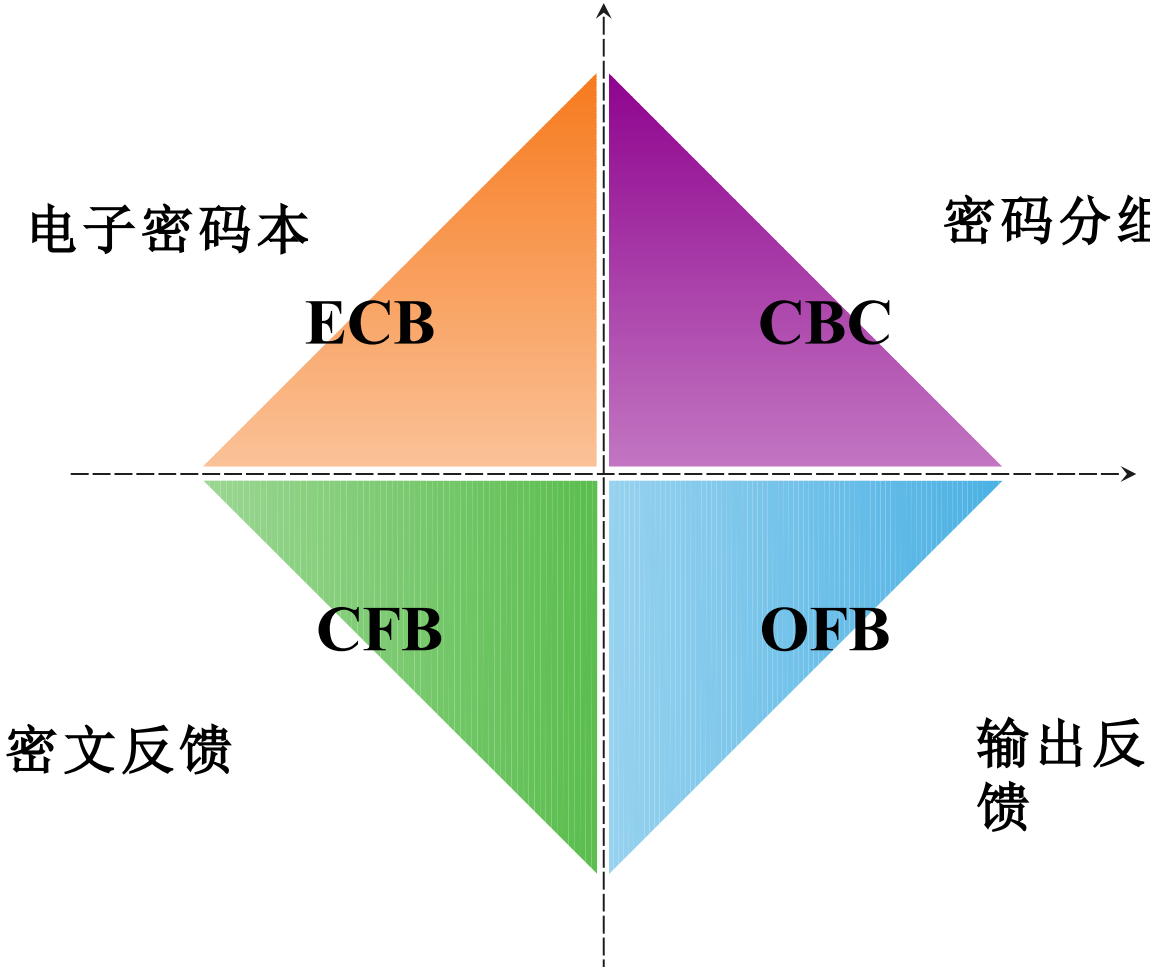
循环左移：

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

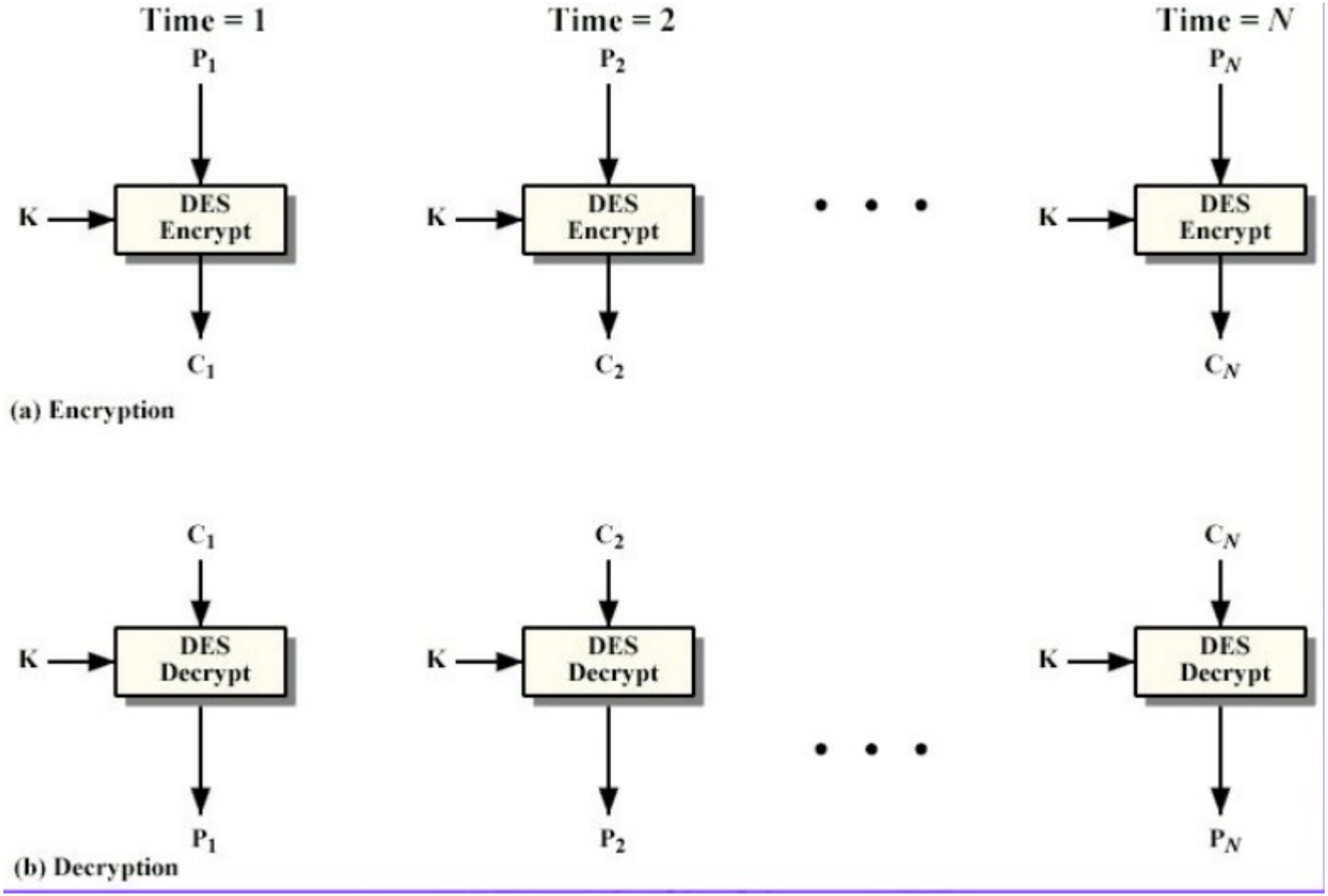
PC-2

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

DES 的四种工作模式



ECB 模式


Time = 1
Time = 2
Time = N
P_1
P_2
P_N
K
DES Encrypt
C_1
C_2
C_N
(a) Encryption
C_1
C_2
C_N
DES Decrypt
P_1
P_2
P_N
(b) Decryption

实验过程中需要思考的问题

1. 相对于古典密码，DES 是怎么提高抗攻击性的？

课后实验代码： LibTomCryp

LibTomCrypt is a library that provides various cryptographic algorithms in a highly modular and flexible manner.

The library is free for all purposes without any express guarantee it works.

实验涉及的函数 1

int des_setup(const unsigned char *key, int keylen, int num_rounds, des_key *skey)

函数名称：密钥生成函数

参数说明：

key 是一个指针，指向用户输入的初始密钥。

keylen 是输入密钥的长度，以字节为单位。

num_rounds 是加密轮数，当输入 0 时，使用算法默认的轮数。

skey 是一个指向结构体变量的指针，存储每轮使用的子密钥。

des_key 的定义

```
typedef struct des_key
{
    ulong32 ek[32], dk[32];
    // ek 存储加密时用的子密钥， dk 里面存储解密时用的子密钥
}des_key ；
```

结构体中用两个 32bit 的整数来存储一轮的 48bit 密钥，每一个 32bit 整数被分成 4 个 8bit，每个 8bit 的低 6 位 bit 存储密钥。

如果把 48bit 密钥分成 8 组，则这 8 组的顺序按存储的顺序从高到低分别为 1、3、5、7、2、4、6、8。这样作是为了加密时可以把扩展和查表运算结合进行。

实验涉及的函数 2

void des_ecb_encrypt(const unsigned char *pt, unsigned char *ct, des_key *key)

函数名称：加密函数

参数说明：

pt 是指向待加密的明文数组的指针

ct 是指向存储加密结果的指针

key 是调用密钥生成函数后存储有每一轮子密钥的结构体变量

加密成功时，返回 CRYPT_OK 。

实验涉及的函数 3

void des_ecb_decrypt(const unsigned char *ct, unsigned char *pt, des_key *key)

函数名称：解密函数

参数说明：

ct 是指向待解密的密文数组的指针

pt 是指向存储解密结果的指针

key 是调用密钥生成函数后存储每一轮子密钥的结构体变量

解密成功时，返回 **CRYPT_OK** 。

加密和解密时， **pt** 和 **ct** 可以指向同一块内存。

实验涉及的函数 4

int des_test(void)

函数名称：测试函数

这个函数用来对加密算法进行测试。函数体内部定义了对应的明文和密文数组，并且进行了多轮加密和解密。这个函数还可以用来测试函数的运行时间。

实验涉及的函数 5

int des_keysize(int *desired_keysize)

函数名称：密钥长度检验函数

参数说明：

desired_keysize 是使用者所想要的密钥长度

当密钥长度小于所需密钥长度时，返回值为 **CRYPT_INVALID_KEYSIZE** ，否则， **desired_keysize** 指向的变量被置为 **8** 。

实验报告要求

实验报告提交时间

2012 年 11 月 26 日 17:00 之前。

实验报告内容：

参见《第二次实验课实验报告要求》。

参考资料

《信息安全实验指导》 崔宝江，周亚建