第21章 网络安全(2)
对称密码学

南京大学计算机系 黄皓 教授

2007年12月4日 星期二

2007年12月11日 星期二

对称密码学

1. 基本概念 — 一个例子

- 问题
  - 两个朋友Alice和Bob想一起外出，但是他们定不下来是去电影院还是歌剧院。他们达成一个通过抛掷硬币来决定的协议：
  - Alice说：“你选择一面，然后我来抛”。他们约定：如果Bob选择的一面朝上则有Bob决定，否则有Alice决定。
  - 假想这两个朋友尝试在电话上执行这个协议，如果Alice对Bob说：“你选则一面，然后我来抛，并且告诉你是否你赢了”。

1. 基本概念 — 一个例子

- 单向函数
  - 对任意整数x，由x计算f(x)是容易的，而给出f(x)，要找出对应的原像x是不可能的，不管x是奇数还是偶数。
  - 不可能找出一对整数(x，y)，满足x≠y且f(x)=f(y)。

1. 基本概念 — 一个例子

- 协议
  - Alice和Bob已经同意:
    - 有一个单向函数f
    - f(x)中的偶数x代表“正面”，奇数x代表“背面”
  - Alice选择一个大随机数x并计算f(x)，然后通过电话告诉Bob f(x)的值;
  - Bob告诉Alice自己对x的奇偶性猜测;
  - Alice告诉Bob x的值;
  - Bob验证f(x)，并察看他所做的猜测是正确或错误。

1. 基本概念 — 一个例子

- 安全性
  - 首先，Alice无法找到不同的两个数x和y，其中一个是奇数而另一个是偶数，使其满足f(x)=f(y)。因此，Alice一旦通过电话告诉Bob，f(x)的值(第1步)，她也就向Bob就x的值做出了承诺，她无法再改变x的值。也就是说Alice已经完成了其掷硬币过程。
  - 第二，由于，已知f(x)，Bob不能判定出Alice所使用的x是奇数还是偶数，因周而他不得不把其猜测(第2步)真实地给出。
  - 这样，Alice可给出x的值，令Bob相信其猜测是否正确(第3步)。
  - 如果Bob利用Alice告诉的x对f(x)进行计算的结果(第4步)与Alice在第1步给出的结果一样，且Bob相信f所具有的性质，则Bob应该相信最终的输赢。

1. 基本概念 — 术语

- 消息被称为明文（plain text）。
- 用某种方法伪装消息以隐藏它的内容的过程称为加密(encryption, encipher)。
- 加了密的消息称为密文(cipher text)。
- 而把密文转变为明文的过程称为解密(decryption, decipher)。

1. 基本概念—术语

- 使消息保密的技术和科学叫做**密码编码学(cryptography)**。
- 从事此行的叫密码编码者（**cryptographer**）。
- 破译密文的科学和技术叫做**密码分析学**（**cryptanalysis**）。
- 从事密码分析的专业人员叫做**密码分析者**（**cryptanalyst**）。
- **密码学**包括密码编码学和密码分析学两者。现代的密码学家通常也是理论数学家。

1. 基本概念—密码学的其它作用

- **鉴别** 消息的接收者应该能够确认消息的来源；入侵者不可能伪装成他人。
- **完整性** 消息的接收者应该能够验证在传送过程中消息没有被修改；入侵者不可能用假消息代替合法消息。
- **抗抵赖** 发送者事后不可能虚假地否认他发送的消息。

1. 基本概念—算法和密钥

- 密码算法也叫密码，是用于加密和解密的数学函数。通常情况下，有两个相关的函数：一个用作加密，另一个用作解密。
- 明文用M（消息），密文用C表示，加密函数E作用于M得到密文C，用数学表示为：

$$E(M)=C.$$

- 相反地，解密函数D作用于C产生M

$$D(C)=M.$$

- 先加密后再解密消息，原始的明文将恢复出来，下面的等式必须成立：

$$D(E(M))=M$$

1. 基本概念—受限制的算法

- 如果算法的保密性是基于保持算法的秘密，这种算法称为受限制的算法。
- 如果有人无意暴露了这个秘密，所有人都必须改变他们的算法。

1. 基本概念—现代密码学

- 现代密码学用密钥解决了这个问题，密钥用K表示。
- 密钥K的可能值的范围叫做密钥空间。
- 加密和解密运算都使用这个密钥，加/解密函数现在变成:

  $E(K_e,\ M) = C$

  $D(K_d, C) = M$

  $D(K_d, E(K_e, M)) = M$

- 如果$K_d = K_e$， 则密码算法为**对称密码算法**。
- 如果$K_d \neq K_e$，则密码算法为**非对称密码算法**。
- 对于非对称密码算法。 $D(K_e, E(K_e, M)) \neq M$
- **所以加密密钥可以公开，又称为公开密钥算法。**

2. 古典密码算法

- 在计算机出现前，密码学由基于字符的密码算法构成。不同的密码算法是字符之间互相代换或者是互相之间换位，好的密码算法是结合这两种方法，每次进行多次运算。
- 现在事情变得复杂多了，但原理还是没变。重要的变化是算法对比特而不是对字母进行变换，实际上这只是字母表长度上的改变，从26个元素变为2个元素。大多数好的密码算法仍然是代替和换位的元素组合。

密码体制

- P: 所有可能的明文组成的有限集;
- C: 所有可能的密文组成的有限集;
- K: 所有可能的密钥组成的有限集;

对任意的$k \in K$，都存在
一个加密法则$e_k \in E$和**相应的**解密法则$d_k \in D$，满足:

对任意的e_k: $P \rightarrow C$，d_k: $C \rightarrow P$，明文$x \in P$，均有

$$d_k(e_k(x)) = x$$

2.1 移位密码

- 令$P=C=K=Z_{26}$。对$0 \leqslant k \leqslant 25$，$x,y \in Z_{26}$，定义

$$e_k(x) = x+k \bmod 26$$

$$d_k(y) = y-k \bmod 26$$

- 著名的凯撒密码就是一种移位密码，k=3;

ABCDEFGHIJKLMNOPQRSTUVWXYZ

DEFGHIJKLMNOPQRSTUVWXYZABC

- 密钥的数量为25

代换密码（单表代替）

- 令$P=C=K=Z_{26}$。K由在有限集{0,1,…,25}上所有的置换π组成。对任意的K，定义：

$$e_{\pi}(x)= \pi(x)$$

$$d_{\pi}(y)= \pi^{-1}(y)$$

- 密钥的数量为26!-1

字母频率表

字母	概率	字母	概率	字母	概率	字母	概率
A	0.082	H	0.061	O	0.075	W	0.023
B	0.015	I	0.070	P	0.019	X	0.001
C	0.028	J	0.002	**Q**	0.001	Y	0.020
D	0.043	K	0.008	R	0.060	Z	0.001
E	0.127	L	0.040	S	0.063		
F	0.022	M	0.024	T	0.091		
G	0.020	N	0.067	U	0.028		

单表代替密码的缺点

- 在单表代替下字母的频度、重复字母模式、字母结合方式等统计特性除了字母名称改变以外，都未发生变化，依靠这些不变的统计特性就能破译单表代换;

2.2 多表代替密码

- Vigenere密码是由法国密码学家Blaise de Vigenere于1858年提出的一种密码，它是一种以移位代换为基础的周期代换密码。

- 设m是一个整数。定义$P=C=K=(Z_{26})^m$。对任意的密钥$K=(k_1,k_2,\ldots, k_m)$，定义

$$e_k(x_1,x_2,\ldots, x_m)=(x_1+k_1, x_2+k_2,\ldots, x_m+k_m)$$

$$d_k(y_1,y_2,\ldots, y_m)=(y_1-k_1, y_2-k_2,\ldots, y_m-k_m)$$

多表代替密码的例子

例1 设m=6, K=cipher,

明文串:

this cryptosystem is not secure

x=(19,7,8,18,2,17,24,15,19,14,18,24,18,19,4,12,8,18,13,14,19,18,4,2,20,1,4)。

密钥:

k=(2,8,15,7,4,17)

19	7	8	18	2	17	24	15	19
2	8	15	7	4	17	2	8	15
21	15	23	25	6	8	0	23	8
14	18	24	18	19	4	12	8	18
7	4	17	2	8	15	7	4	17
21	22	15	20	1	19	19	12	9
13	14	19	18	4	2	20	1	4
2	8	15	7	4	17	2	8	15
15	22	8	25	8	19	22	25	19

密文串为：VPXZGIAXIVWPUBTTMJPWIZITWZT

多表代替密码的优点

- 在多表代换下，原来明文中的统计特性通过多个表的平均作用而被隐蔽了起来。多表代换密码的破译要比单表代替密码的破译难得多。

多表代替密码的破解（1）

- 1863年，普鲁士军官F.Kasiski发明了通过分析密文中的字母重复的情况来确定周期多表代替密码的准确周期的方法。
- 例：密钥：dog，明文：to be or not to be

t	o	b	e	o	r	n	o	t	t	o	b	e
d	o	g	d	o	g	d	o	g	d	o	g	d
w	c	h	h	c	x	q	c	z	w	c	h	h

多表代替密码的破解（2）

- 在密文中字符串wchh重复了两次，其间个为9个字符。
- 这个距离是密钥长度的倍数，可能的密钥长度是1, 3, 9。
- 判定了长度之后就可以用频率分析法来破译各个单表代替密码了。
- 多表代替密码的破解的原因：密钥的长度太短。

密钥字长度的确定— 重合指数法(index of coincidence)

- 设**x**=$x_1x_2\cdots\cdots x_n$，**x**的重合指数定义为**x**中两个随机字母相同的概率，记为$I_c(x)$。

$$I_c(x) = \frac{\sum_{i=0}^{25}\binom{f_i}{2}}{\binom{n}{2}} = \frac{\sum_{i=0}^{25} f_i(f_i-1)}{n(n-1)} = \sum_{i=0}^{25}\frac{f_i}{n}\bullet\frac{f_i-1}{n-1} \approx \sum_{i=0}^{25} p_i^2 = 0.065$$

- 假定$Y=y_1y_2\ldots y_n$是多表密码算法加密后的密文，将Y分成d个密文子串：

  $Y_1=y_1y_{d+1}y_{2d+1}$, $Y_2=y_2y_{d+2}y_{2d+2}$ … $Y_d=y_dy_{2d}y_{2d}\cdots$

- 如果d为密钥字的长度， 则Y_i的重合指数 ≈ 0. 065
- 如果d不是密钥字长度，则Y_i的由完全随机的字母组成，所以重合指数 $I_c(z_i)=26/(1/26)^2 \approx 0.038$

密钥字的确定 —

重合互指数法(Mutual index of coincidence)

- 假定$x=x_1x_2\ldots x_n$，$y=y_1y_2\ldots y_m$，x 和y的互指数定义为从x中随机取一个字母和从y中随机取的一个字母相同的概率，记为：$MI_c(x,y)$。
- 如果在x和y中a,b,….,z出现的次数分别为$f_0,f_1,\ldots, f_{25}$和 $g_0, g_1, \ldots, g_{25}$,则

$$MI_c(x,y) = \sum_{i=0}^{25} \frac{f_i}{n} \bullet \frac{g_i}{m}$$

密钥字的确定 —

重合互指数法(Mutual index of coincidence)

- 假定$Y=y_1y_2\ldots y_n$是多表密码算法加密后的密文，$k=(k_1, k_2, \ldots, k_d)$是密钥字，将Y分成d个密文子串：$Y_1, Y_2, \ldots, Y_d$。考虑$Y_i$中的一个随机字母和$Y_j$中的一个随机字母，两个字母都是A和B的概率分别是

$$p_{-k_i} p_{-k_j}, p_{1-k_i} p_{1-k_j}$$

- 我们可以得到：

$$MI_c(Y_i, Y_j) = \sum_{h=0}^{25} p_{h-k_i} p_{h-k_j} = \sum_{h=0}^{25} p_h p_{h+ki-k_j}$$

- 可以看出，**$MI_c(x,y)$ 的估计值只依赖于（k_i-k_j）mod 26**，称这个差为**Y_i**与**Y_j**的相对位移。试验表明相对位移不等于**0**时，这些估计值在**0.031**到**0.045**之间，相对位移等于**0**时，估计值为**0.065**。

密钥字的确定 —

重合互指数法(Mutual index of coincidence)

l	MI_c	l	MI_c	l	MI_c	l	MI_c
0	0.065	4	0.044	8	0.034	12	0.039
1	0.039	5	0.033	9	0.034	13	0.043
2	0.032	6	0.036	10	0.038		
3	0.034	7	0.039	11	0.045		

- 我们用这个表来推测Y_i与Y_j的相对*位移* $l=k_i-k_j \pmod{26}$)
- 考虑长度为d 的密钥字$e_0=(0,0,\ldots 0)$, $e_1=(1,1,\ldots,1)$，……。用它们加密Y_j，分别记为Y^0, Y^1, ……。
- **计算$MI_c(Y_i, Y_j^g)$, 当$g=l$ 时， MI_c接近于0.065。**
- 设$k=(k_1,k_2,\ldots,k_m)$为密钥字;
- 固定k_1, 猜测k_i-k_1的差;
- 最后猜测k_1。
- **冯登国，密码分析学，清华大学出版社，1.4节。**

2.3 置换密码

- 置换：定义在有限集X上的置换为一个双射函数π: X→X。 π^{-1}定义为π的逆函数

$$\pi(x)=x', \quad \pi^{-1}(x')=x$$

- 令m为一整数。 $P=C=(Z_{26})^m$，K由定义在集合{1,2,…,m}上的置换组成。对于任意的密钥π（置换），定义加密
  - $e_\pi(x_1,x_2,\ldots,x_m)=(x_{\pi(1)}, x_{\pi(2)},\ldots, x_{\pi(m)})$
  - $d_\pi(y_1,y_2,\ldots,y_m)=(y_{\pi^{-1}(1)}, y_{\pi^{-1}(2)},\ldots, y_{\pi^{-1}(m)})$

Enigma

- 直到第一次世界大战结束为止，所有密码都是使用手工来编码的，就是铅笔加纸的方式。
- 考虑到不能多次重复同一 种明文到密文的转换方式，和 民用的电报编码解码不同，加密人员 并不能把转换方式牢记于心。转 换通常是采用查表的方法，所查表又每日不同，所以解码速度极慢。

Enigma(1)

- 解密一方当时正值春风得意之时，几百年来被认为坚不可破的维 吉耐尔(Vigenere)密码和它的变种也被破解。
- 而无线电报的发明，使得 截获密文易如反掌。无论是军事方面还是民用商业方面都需要一种可 靠而又有效的方法来保证通讯的安全。

Enigma(2)

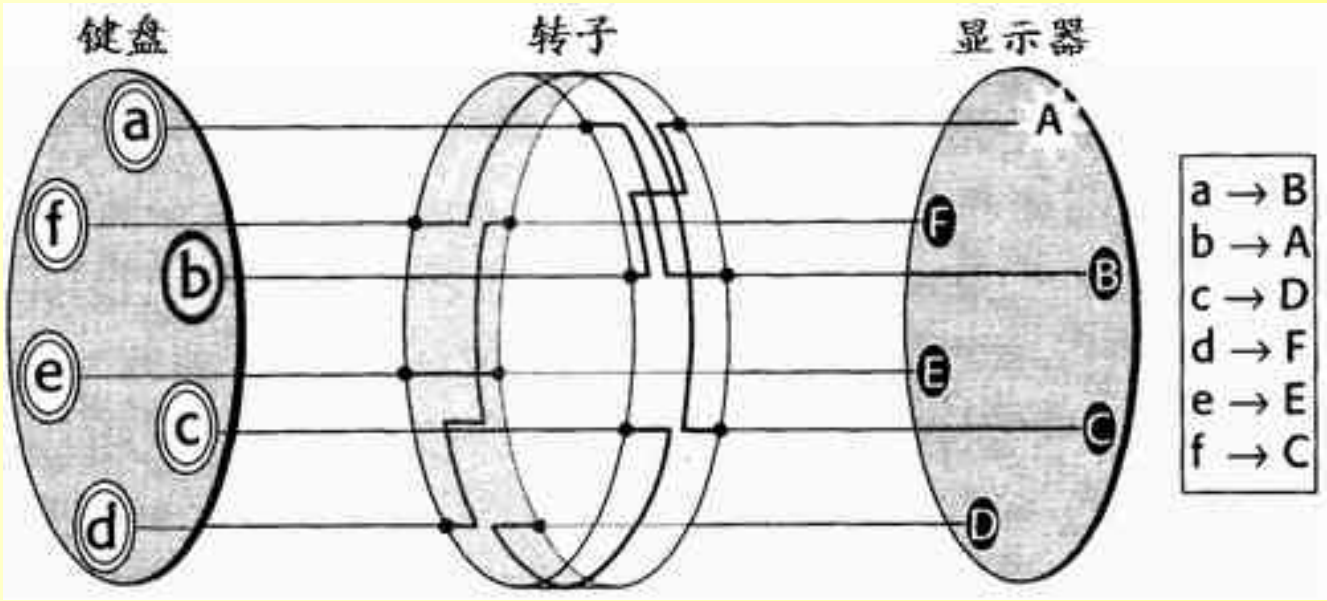
- **1918**年，德国发明家亚瑟·谢尔比乌斯**(Arthur Scherbius)**和他的朋 友理查德·里特**(Richard Ritter)**创办了谢尔比乌斯和里特公司。这是一 家专营把新技术转化为应用方面的企业，很象现在的高新技术公司。
- 谢尔比乌斯负责研究和开发方面，紧追 当时的新潮流。他的一个想 法就是要用二十世纪的电气技术来取代那种过时的铅笔加纸的加密方 法。

Enigma(3)

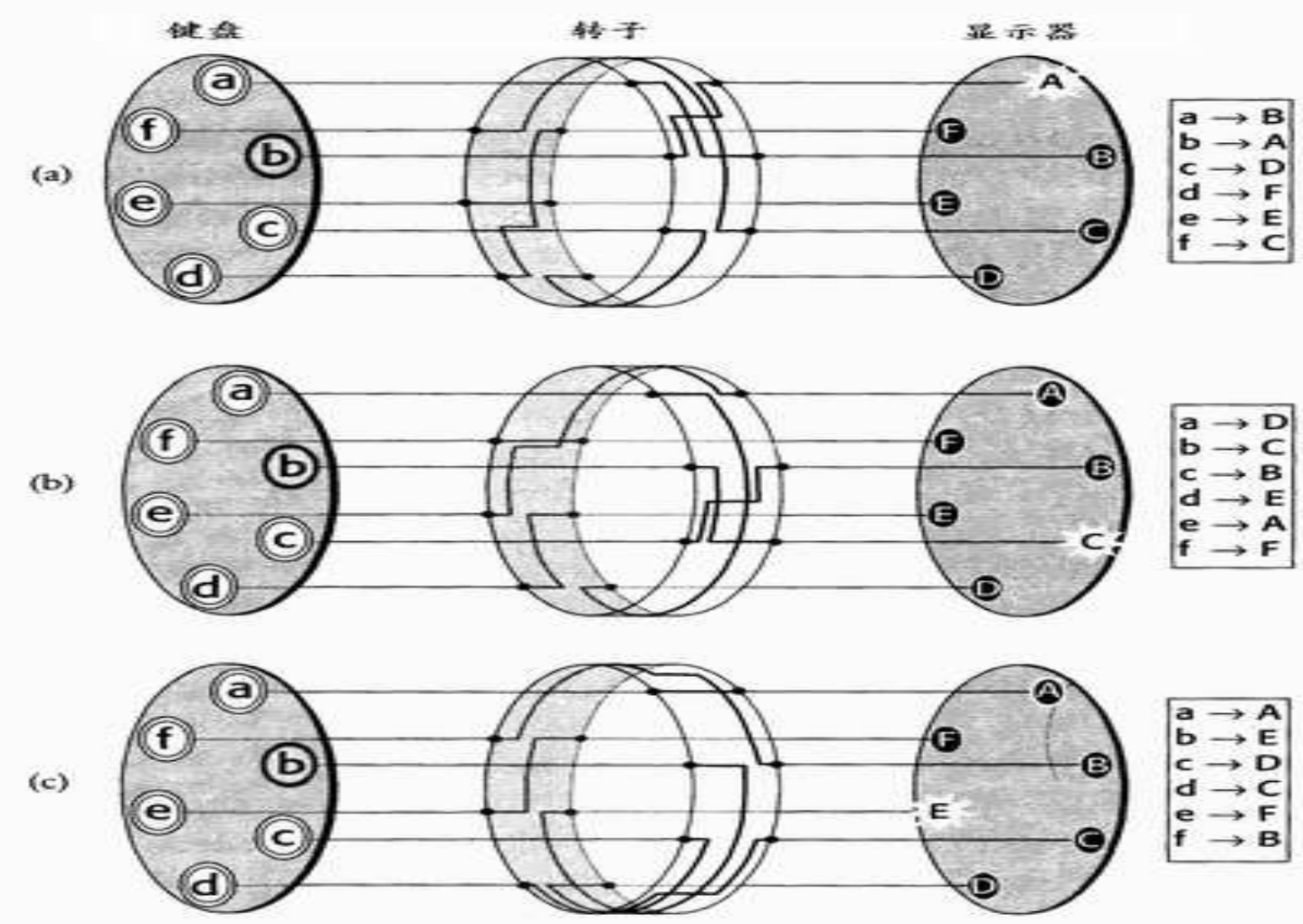
- 谢尔比乌斯发明的加密电子机械名叫ENIGMA，在以后的年代里，它将被证明是有史以来最为可靠的加密系统之一。
- 而对这种可靠性的盲目乐观，又使它的使用者遭到了灭顶之灾。

Enigma(4)

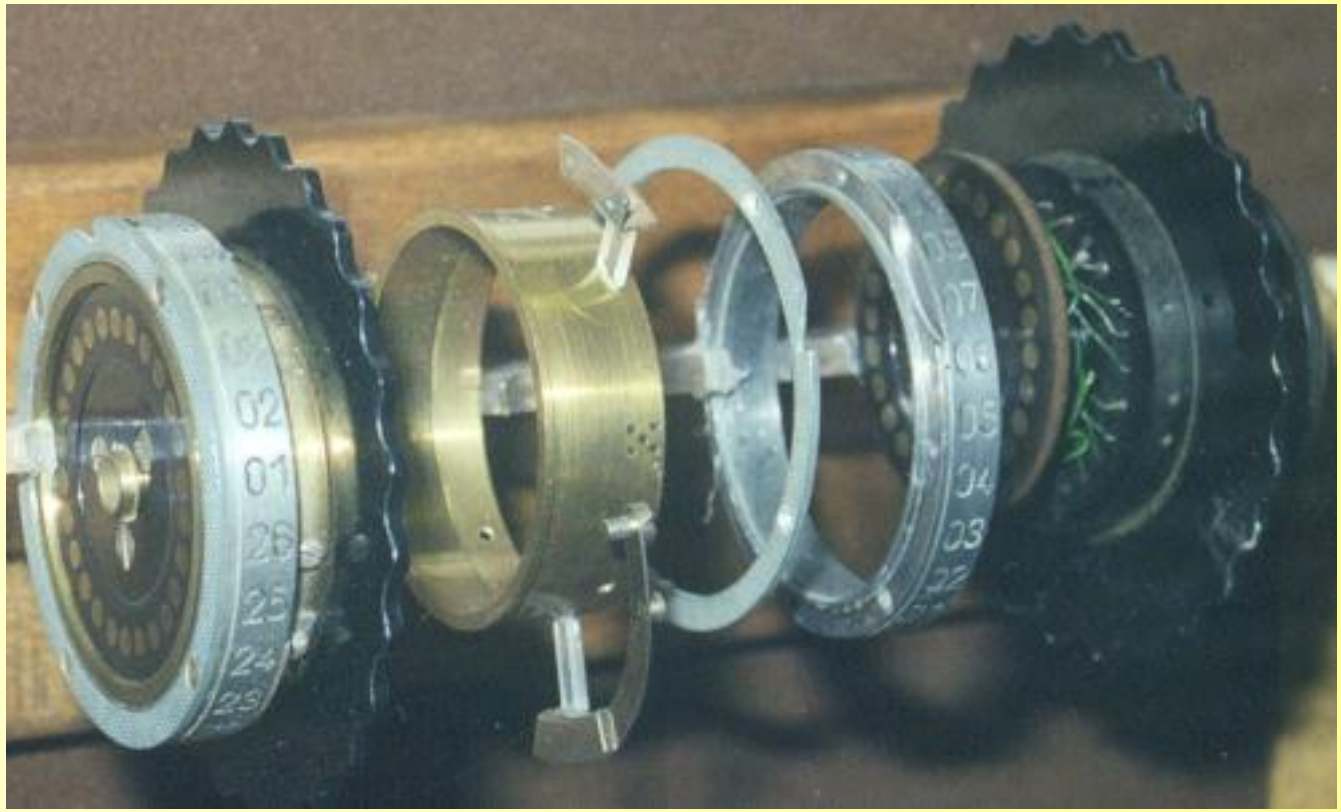
- ENIGMA它可以被分解成相当简单的几部分。下 面的图是它的最基本部分的示意图，我们可以看见它的三个部分：键 盘、转子和显示器。



Enigma(5)



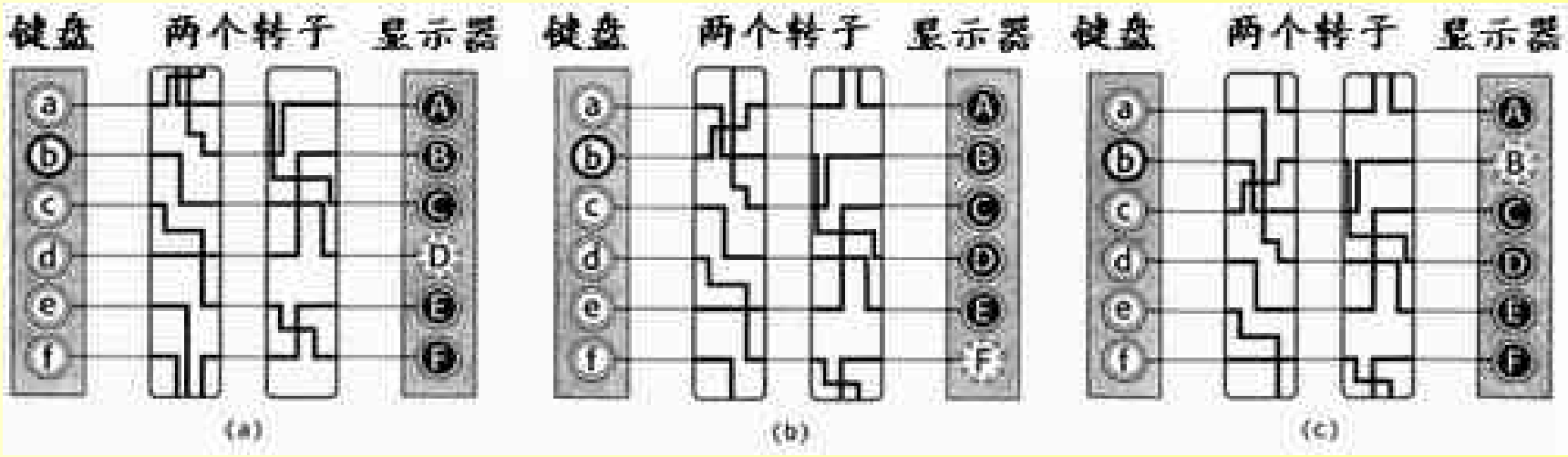
Enigma(6)



照片左方是一个完整的转子，右方是转子的分解，我们可以看到安装在转子中的电线

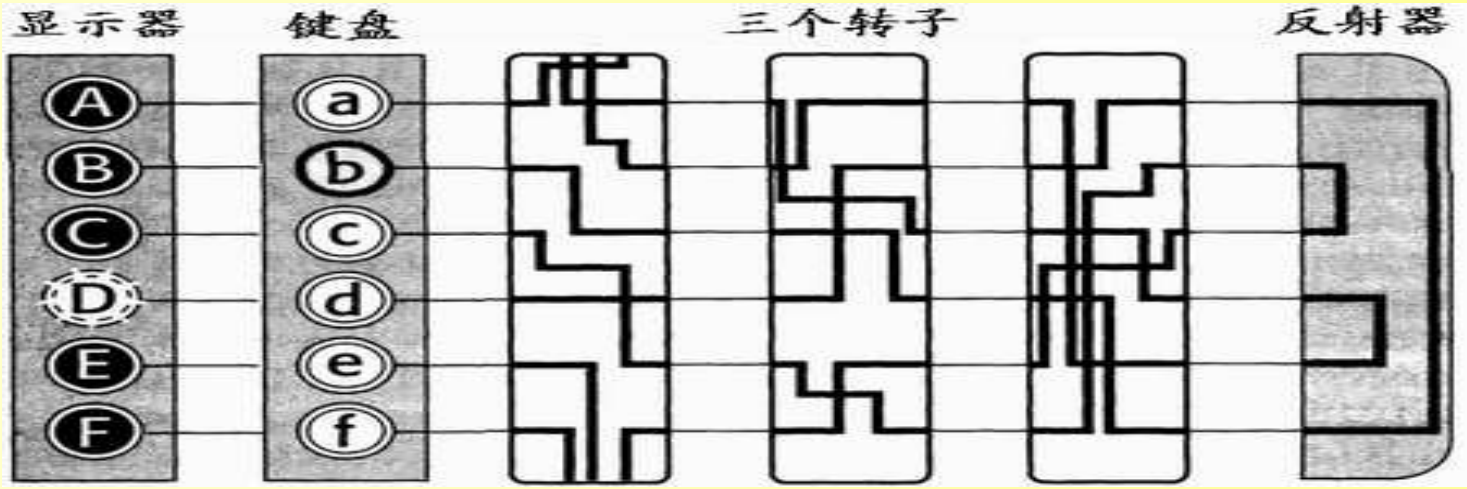
Enigma(7)

- 当第 一个转子转动整整一圈以后，它上面有一个齿拨动第二个转子，使得 它的方向转动一个字母的位置。



Enigma(7)

- 在此基础上谢尔比乌斯十分巧妙地在三个转子的一端加上了一个反射器，而把键盘和显示器中的相同字母用电线连在一起。反射器和 转子一样，把某一个字母连在另一个字母上，但是它并不转动。这么一个固定的反射器它并不增加可以使用的编 码数目。它和解码联系起来了。



Enigma(8)

- 发信人首先要调节三个转 子的方向，使它们处于26*26*26 = 17576个方向中的一个（事实上转子的初始方向 就是密匙，这是收发双方必须预先约定好的）
- 依次键入明文， 并把闪亮的字母依次记下来
- 然后就可以把加密后的消息用比如电报 的方式发送出去。
- 当收信方收到电文后，使用一台相同的ENIGMA， 按照原来的约定，把转子的方向调整到和发信方相同的初始方向上;
- 依次键入收到的密文，并把闪亮的字母依次记下来，就得到了明 文。

Enigma(9)

- 当然谢尔比乌斯还可以再多加转子，但是我们看见每加一个转子 初始方向的可能性只是乘以了26。尤其是，增加转子会增加ENIGMA 的体积和成本。
- 谢尔比乌斯希望他的加密机器是便于携带的;
- 首先他把三个转子做得可以拆卸下来互相交换，这样一来初 始方向的可能性变成了原来的六倍。 26*26*26*6 = 105,456

Enigma(10)

- 下一步谢尔比乌斯在键盘和第一转子之间增加了一个连接板。这 块连接板允许使用者用一根连线把某个字母和另一个字母连接起来，这样这个字母的信号在进入转子之前就会转变为另一个字母的信号。 这种连线最多可以有六根（后期的ENIGMA具有更多的连线），这样 就可以使 6 对字母的信号互换，其他没有插上连线的字母保持不变。 当然连接板上的连线状况也是收发信息的双方需要预先约定的。
- 连接板上两两交换6对字母的可能性数目非常巨大，有100391791500种
- 密钥
  - 连接板的连接：A/L-P/R-T/D-B/W-K/F-O/Y
  - 转子的顺序：2,3,1
  - 转子的初始方向：Q-C-W

Enigma(11)

- 调整好ENIGMA，现在操作员可以开始对明文加密了。但是我们看到每天只有一个密钥，如果这一天的几百封电报都以这个密钥加密发 送的话，暗中截听信号的敌方就会取得大量的以同一密钥加密的信息， 这对保密工作来说不是个好兆头。我们记得在简单替换密码的情况下， 如果密码分析专家能得到大量的密文，就可以使用统计方法将其破解。

Enigma(12)

- 尽管不知道对ENIGMA是否可以采用类似的统计方法，德国人还是 留了个心眼。他们决定在按当日密钥调整好ENIGMA机后并不直接加密要发送的明文，首先发送的是一个新的密钥。
- 设置新密钥时，连接板的连线顺 序和转子的顺序并不改变，和当日通用的密钥相同；转子的初 始方向将按新密钥被改变。
- 操作员首先按照上面所说的方法按当日密钥调整好 ENIGMA，然后随机地选择三个字母，比如说PGH。他把PGH在键盘上 连打两遍，加密为比如说KIVBJE（注意到两次PGH被加密为不同的形式， 第一次KIV，第二次BJE，这正是ENIGMA的特点，它是一种复式替换密码）。然后他把KIVBJE记在电文的最前面。接着他重新调整三个转子的 初始方向到PGH，然后才正式对明文加密。

Enigma(13)

- 汉斯—提罗·施密特(Hans-Thilo Schimdt) 于1888年出生在柏林的 一个中产阶级家庭里，一次大战时当过兵打过仗。根据凡尔赛条约， 战败后的德国进行了裁军，施密特就在被裁之列。退了伍后他开了个 小肥皂厂，心想下海从商赚点钱。结果战后的经济萧条和通货膨胀让 他破了产。此时他不名一文，却还有一个家要养。

Enigma(14)

- 鲁道夫给他的二弟在密码处(Chiffrierstelle)找了 个位置。这是专门负责德国密码通讯的机构——ENIGMA的指挥中心， 拥有大量绝密情报。
- 汉斯—提罗把一家留在巴伐利亚，因为在那里生活 费用相对较低，勉强可以度日。就这样他一个人孤零零地搬到了柏林， 拿着可怜的薪水，对大哥又羡又妒，对抛弃他的社会深恶痛绝。

Enigma(15)

- 接下来的事情可想而知。如果把自己可以轻松搞到的绝密情报出 卖给外国情报机构，一方面可以赚取不少自己紧缺的钱，一方面可以 以此报复这个抛弃了他的国家。
- 1931年11月8日，施密特化名为艾斯克 (Asche)和法国情报人员在比利时接头，在旅馆里他向法国情报人员提 供了两份珍贵的有关ENIGMA操作和转子内部线路的资料，得到一万马 克。靠这两份资料，盟国就完全可以复制出一台军用的ENIGMA机。

Enigma(16)

- **1929年1月，波兹南大学数学系主任兹德齐斯罗·克里格罗夫斯基 (Zdzislaw Kryglowski)教授开列了一张系里最优秀的数学家的名单，在这 张名单上，有以后被称为密码研究“波兰三杰”的马里安·雷杰夫斯基 (Marian Rejewski)，杰尔兹·罗佐基(Jerzy Rozycki)和亨里克·佐加尔斯 基 (Henryk Zygalski)。**
- **雷杰夫斯基深知“重复乃密码大敌”。在ENIGMA密码中，最明显 的重复莫过于每条电文最开始的那六个字母——它由三个字母的密钥 重复两次加密而成。德国人没有想到这里会是看似固若金汤的ENIGMA 防线的弱点。**

Enigma(17)

- 雷杰夫斯基每天都会收到一大堆截获的德国电报，所以一天中可 以得到许多这样的六个字母串，它们都由同一个当日密钥加密而成。 比如说他收到四个电报，其中每封电报的开头的六个字母为

	1	2	3	4	5	6
第一封电报:	L	O	K	R	G	M
第二封电报:	M	V	T	X	Z	E
第三封电报:	J	K	T	M	P	E
第四封电报:	D	V	Y	P	Z	X

Enigma(18)

- 对于每封电报来说，
  - 第一个字母和第四个字母
  - 第二个字母和第五个字母
  - 第三个字母和第六个字母

  都是分别 由同一个字母加密而来。

Enigma(19)

- 第一个字母：ABCDEFGHIJKLMNOPQRSTUVWXYZ
- 第四个字母：　___P_____M_RX_____________

如果雷杰夫斯基每天可以得到充分多的电报，他就可以把上面这 个关系表补充完整

- 第一个字母：ABCDEFGHIJKLMNOPQRSTUVWXYZ
- 第四个字母：FQHPLWOGBMVRXUYCZITNJEASDK

Enigma(20)

- 雷杰夫斯基对这样的 表格进行了仔细观察。从字母A开始看，它被对应成F；而F在此表中 又被对应成W，接下去它被对应成A，我们又回到了最先开始的字母， 于是就有了一个循环的字母圈 A→F→W→A。

- 如果考虑所有的字母， 雷杰夫斯基就能写出关于此对应表的所有的循环圈：

A→F→W→A	3个字母的循环圈
B→Q→Z→K→V→E→L→R→I→B	9个字母的循环圈
C→H→G→O→Y→D→P→C	7个字母的循环圈
J→M→X→S→T→N→U→J	7个字母的循环圈

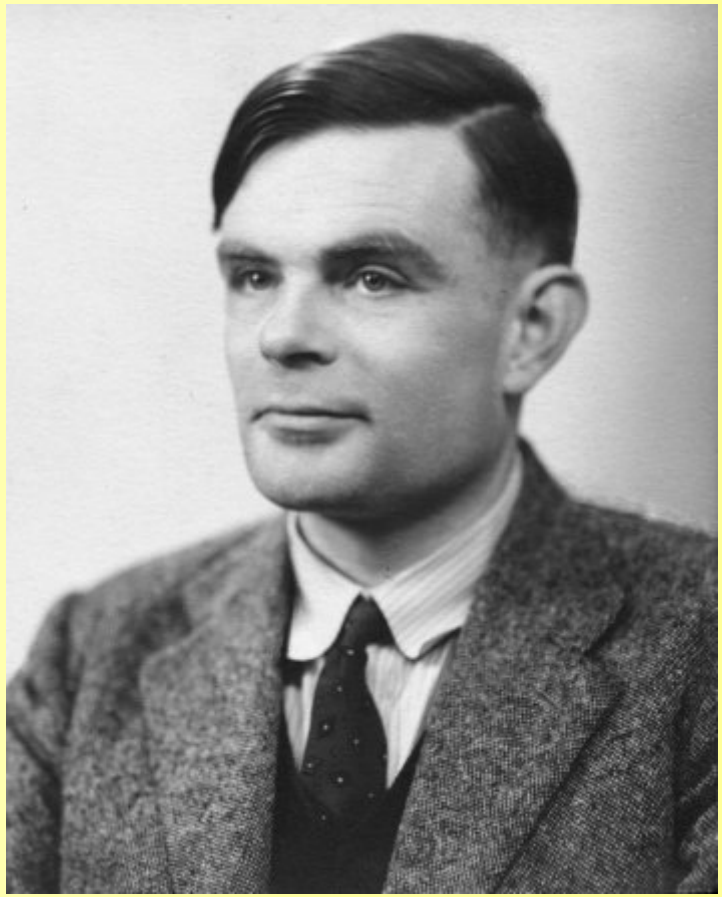
Enigma(21)

- 虽然这些循环圈是由当日密钥，也就是转子的位置，它们的初始方向以及连接板上字母置换造成的
- 但是每组循环圈的 个数和每个循环圈的长度，却仅仅是由转子的位置和它们的初始 方向决定的，和连接板上字母交换的情况无关！

Enigma(22)

- 首先要取得足够的当日电文来构 造字母对应表并且写出字母循环圈;
- 然后根据循环圈的数目和它们的 长度从记录表中检索出相对应的转子位置和初始方向;
- 这就是当日的 密钥（连接板的情况还未知）。
- 循环圈的个数和长度可以看作是这个 密钥的“指纹”——通过建立密钥“指纹”档案，雷杰夫斯基就能及 时地把当天的密钥找出来。

Enigma(23)



Enigma(24)

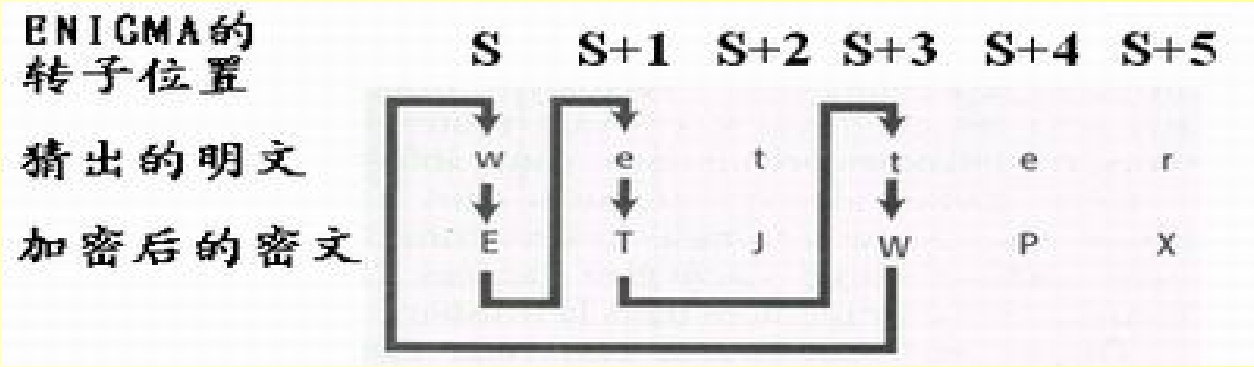
- 但是这是ENIGMA使用中的一个重大弱点，德国人很可能会 发觉这一点并取消这种重复，这样就会使英国密码分析专家的破译手 段变得毫无用处。图灵的任务就是要找到另一种不必利用重复密钥的 破译方法。
- 在分析了以前大量德国电文后，图灵发现许多电报有相当固定的 格式，他可以根据电文发出的时间、发信人、收信人这些无关于电文内容的信息来推断出一部分电文的内容。
- 比方说，要是在六点零五分截获了一份德国电报， 它里面八成有Wetter这个词，也就是德文中的“天气”。根据在此之 前德国人天气预报电文的死板格式，图灵甚至能相当准确地知道这个 词具体在密文的哪个位置。
- 这就使得图灵想到了用“候选单词”这一 方法来破译ENIGMA电文。

Enigma(25)

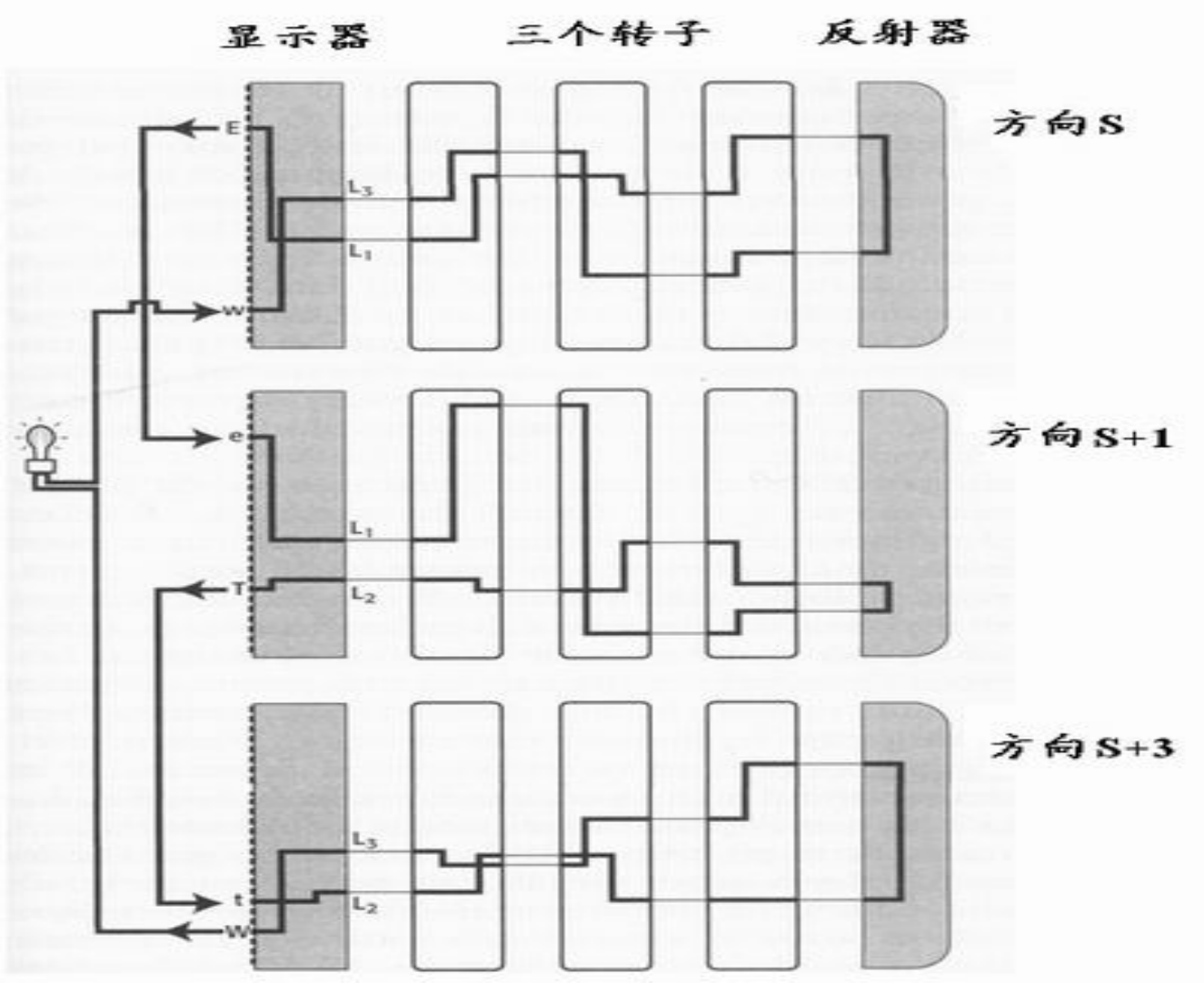
- 如果在一篇密文中，图灵知道Wetter这个词被加密成了ETJWPX，那么剩下的任务就是要找到将Wetter加密成ETJWPX的初始设置。
- 但是雷杰夫斯基的天才思想告诉图灵，必须把转子方向变化造 成的问题和连接板交换字母造成的问题分开来考虑。如果他能够， 他就可以最多只用尝试105456次（6种转子放置方法乘以17576种转 子初始方向）便可找到正确的转子设置。

Enigma(26)

- 图灵找到了这样的特性。这里就存 在着一个字母循环圈。



Enigma(27)



- 假设连接板上有关的交换字母的连线是下 面这样的（三台ENIGMA机上的都一样E←→L1，T←→L2　W←→L3 当然这里的L1、L2和L3都还是未知的。

- 现在假设字母w被输入第一台ENIGMA，它先通过连接板变成了L3， 然后通过三个转子经过反射器，再通过三个转子返回连接板；因为我 们根据候选单词知道w此时会被加密成E，所以没有经过接线板前它一 定是和E对应的L1；L1经过接线板变成E后，直接成了第二台ENIGMA 的输入。提醒一下，第二台ENIGMA的转子方向是S+1，所以根据候选 单词知道e此时会被加密成T，我们来看看具体是怎么回事。从第一台 ENIGMA来的e通过连接板变成了L1，再通过转子和反射器回来变成了 连接板上和字母T对应的L2；通过连接板后变成了T，然后这个T又变 成第三台ENIGMA机上的输入t。第三台ENIGMA机的转子方向是S+3， 这个传送过来的t会被加密成E，具体的情况和上面第一第二台上的类 似。我们发现现在三台ENIGMA机的线路组成了一个闭合回路，如果 在里面加上一个灯泡，它就会亮起来。**这个闭合回路事实上就是那 个字母循环圈的形象化。**

- 2001年4月21日，雷杰夫斯基、罗佐基和佐加尔斯基纪念基金在波 兰华沙设立，基金会在华沙和伦敦设置了纪念波兰数学家的铭牌。

- 2001 年7月，基金会在布莱切利公园安放了一块基石，上面刻着丘吉尔的名 言：“在人类历史上，从未有如此多的人对如此少的人欠得如此多。”

3. 分组密码的原理

- 分组密码将明文序列分成等长的分组，对每一组用同一加密算法和同一密钥进行加密。
- 优点：
  - 容易被标准化
  - 加密解密容易实现同步
- 缺点：
  - 算法庞大
  - 安全性难以证明

分组密码的设计准则—安全性原则

- 影响密码算法安全的主要因素是：混乱与扩散

  Shannon C E. Communication theory of secrecy systems, Bell System Technology Journal, 1949, 28, 656-715.

分组密码的设计准则—安全性

- 扩散(diffusion)：明文的统计结构被扩散消失到了密文的长程统计特性中；让明文的每个数字影响许多密文的数字；例如

$$Y_n = \sum_{i=1}^{k} m_{n+i} \quad (\mathrm{mod}\ 26)$$

- 扰乱(confusion)：使得密文的统计特性与加密密钥之间的关系尽量复杂。可以用复杂的代换算法来达到这个目的。

分组密码的设计准则—安全性

- 抵抗现有的所有攻击
  - 抗差分分析
  - 抗线性分析
  - 安全强度的稳定性

分组密码的设计准则—实现原则

- 软件实现原则
  - 子块长度自然适应软件编程8，16，32，…
  - 尽量避免比特置换。
  - 使用标准处理器的指令:加法、乘法、移位。
- 硬件实现原则
  - 加密和解密实现的相似性，同一个器件即可以用来加密又可以用来解密。

分组密码的设计准则 — 有效性

- 应使得密钥最大限度地起到安全作用
- 有效性差的例子:
  - 2n比特的密钥k_1, k_2解密方法$Y=(x \oplus k_1) \oplus k_2$ 等效于密钥n比特密钥$k= k_1 \oplus k_2$加密$Z=x \oplus k$

代换 – 置换网络SPN
— Substitution Permutation Network

- 设l,m,Nr为整数,
- $\pi_s: \{0,1\}^l \rightarrow \{0,1\}^l$为置换
- $\pi_p: \{0,1,\ldots, l\cdot m\} \rightarrow \{0,1,\ldots, l\cdot m\}$为置换
- $P=C=\{0,1\}^{l\cdot m}$;
- $k=(k^1,k^2,\ldots, k^{Nr+1})$

- $x=(x_1,x_2,\ldots x_{l\cdot m})$
  $=x_{(1)}\| x_{(2)}\|\ldots\|x_{(m)}$
- $x(i)=(x_{(i-1)l+1},\ldots, x_{i\cdot l})$

- $w^0=x$
- for r=1 to Nr-1 do
  - $u^r=w^{r-1} \oplus K^r$
  - for i=1 to m do
    - $v^r_{(i)} = \pi_s(u^r_{(i)})$
  - $w^r=(v^r_{\pi_p(1)}, v^r_{\pi_p(2)},\ldots, v^r_{\pi_p(lm)},))$
- $u^{Nr}=w^{Nr-1} \oplus K^{Nr}$
- for i=1 to m do
  - $v^{Nr}_{(i)} = \pi_s(u^{Nr}_{(i)})$
- $y=v^{Nr} \oplus K^{Nr}$

乘积密码

- 设S=(P,P,K,E,D)
- S×S=(P,P,K×K,E,D)
  - $e_{(k1,k2)}(x)=e_{k2}(e_{k1}(x))$
  - $d_{(k1,k2)}(x)=d_{k1}(d_{k2}(x))$
- 如果S×S = S，则称S为幂等的，即对于任意的$k_1 \times k_2 \in K \times K$,都存在$k \in K$,使得

$$e_{(k1,k2)}(x)=e_k(x))$$

- 如果S为幂等的，则乘积密码S×S没有提高安全性；如果S不是幂等的，则多次迭代可能提高安全性。
- 许多古典密码，如移位密码、代换密码、 Vigenere密码和置换密码都是幂等的。

Feistel 密码

- L_0, R_0
- $L_1=R_0$, $R_1 = L_0 \oplus F(k_1,R_0)$;
- $L_2=R_1$, $R_2 = L_1 \oplus F(k_2,R_1)$;

……

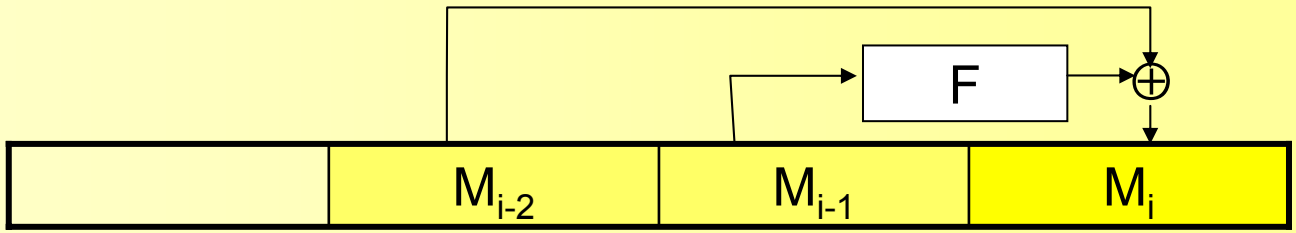
- $L_i=R_{i-1}$, $R_i = L_{i-1} \oplus F(k_i,R_{i-1})$;

……

- $L_{n-1}=R_{n-2}$; $R_{n-1}=L_{n-2} \oplus F(k_{n-1},R_{n-2})$;
- $L_n=R_{n-1}$, $R_n = L_{n-1} \oplus F(k_n,R_{n-1})$;

Feistel 密码是乘积密码

- $m_0=L_0$, $m_1=R_0$,
- **for i=2 to n do**
  - **$m_i=m_{i-2} \oplus F(k_{i-1}, m_{i-1})$**
- **密文$=L_n \| R_n=m_n \| m_{n+1}$**



4. 对称密钥算法DES

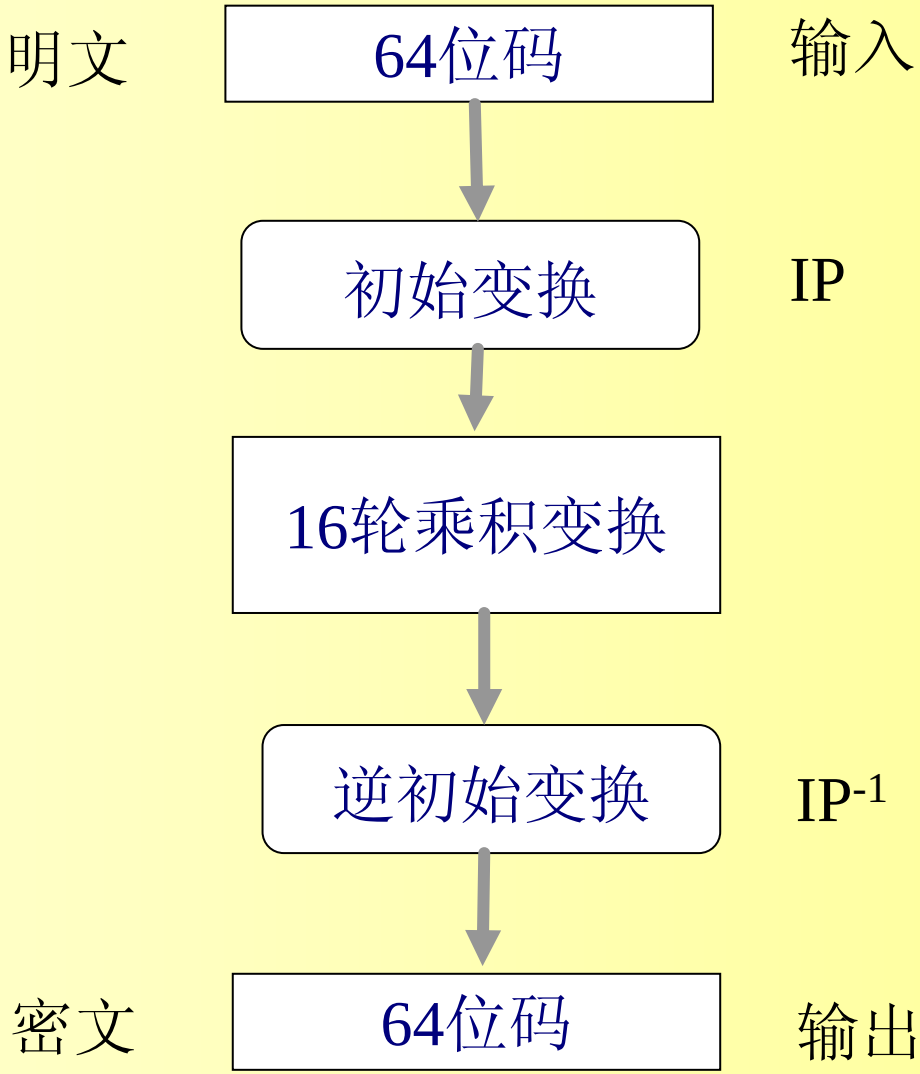
4.1 DES加密算法的背景

- 发明人：美国IBM公司W. Tuchman 和 C. Meyer 1971-1972年研制成功。
- 产生：美国国家标准局（NBS)1973年5月到1974年8月两次发布通告，公开征求用于电子计算机的加密算法。经评选从一大批算法中采纳了IBM的LUCIFER方案。
- 标准化：DES算法1975年3月公开发表，1977年1月15日由美国国家标准局颁布为数据加密标准（Data Encryption Standard），于1977年7月15日生效。
- 标准的条款中规定每五年对标准重新审查一次。1983年DES被认证了一次，1987年经过争论DES在获准使用到1992。1992年仍然没有DES的替代方案，NIST决定在延长DES 5年，并在这5年中考虑替代方案。1997年NIST发起了推选用于保护敏感的无密级的信息的加密算法的活动，最终RIJNDAEL算法胜出成为AES。

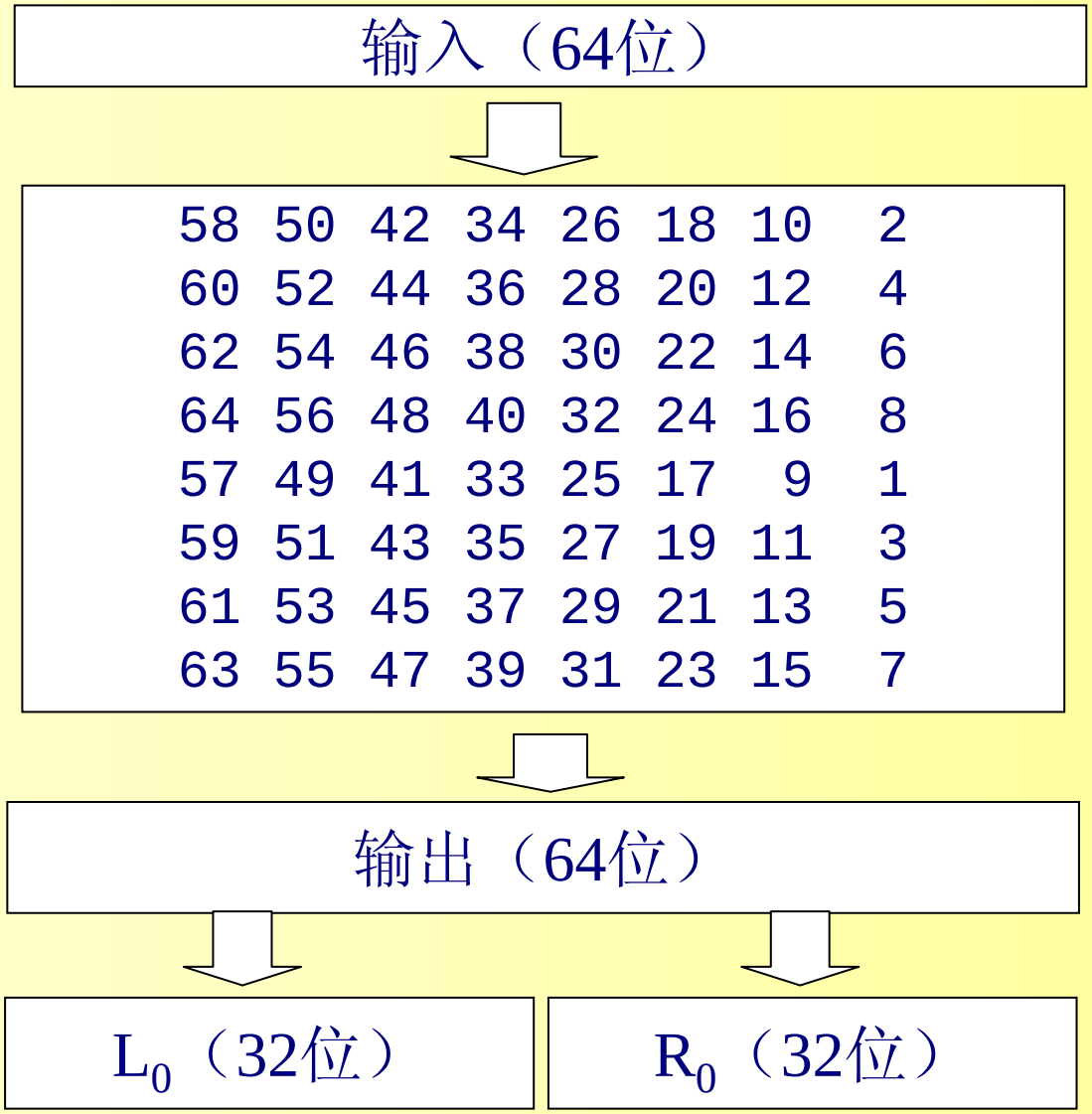
4.2 DES算法

1. 输入64位明文M;
2. 初始变换IP：$IP(M)=L_0 \| R_0$
3. 乘积变换:
   - $L_i=R_{i-1}$，$R_i=L_{i-1} \oplus f(R_{i-1}, K_i)$，　$i=1,\ldots16$
4. 初始变换逆IP^{-1}，　$IP^{-1}（L_{16} \| R_{16}）$

4.2 DES算法



初始变换IP


输入（64位）
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
输出（64位）
L_0（32位）
R_0（32位）

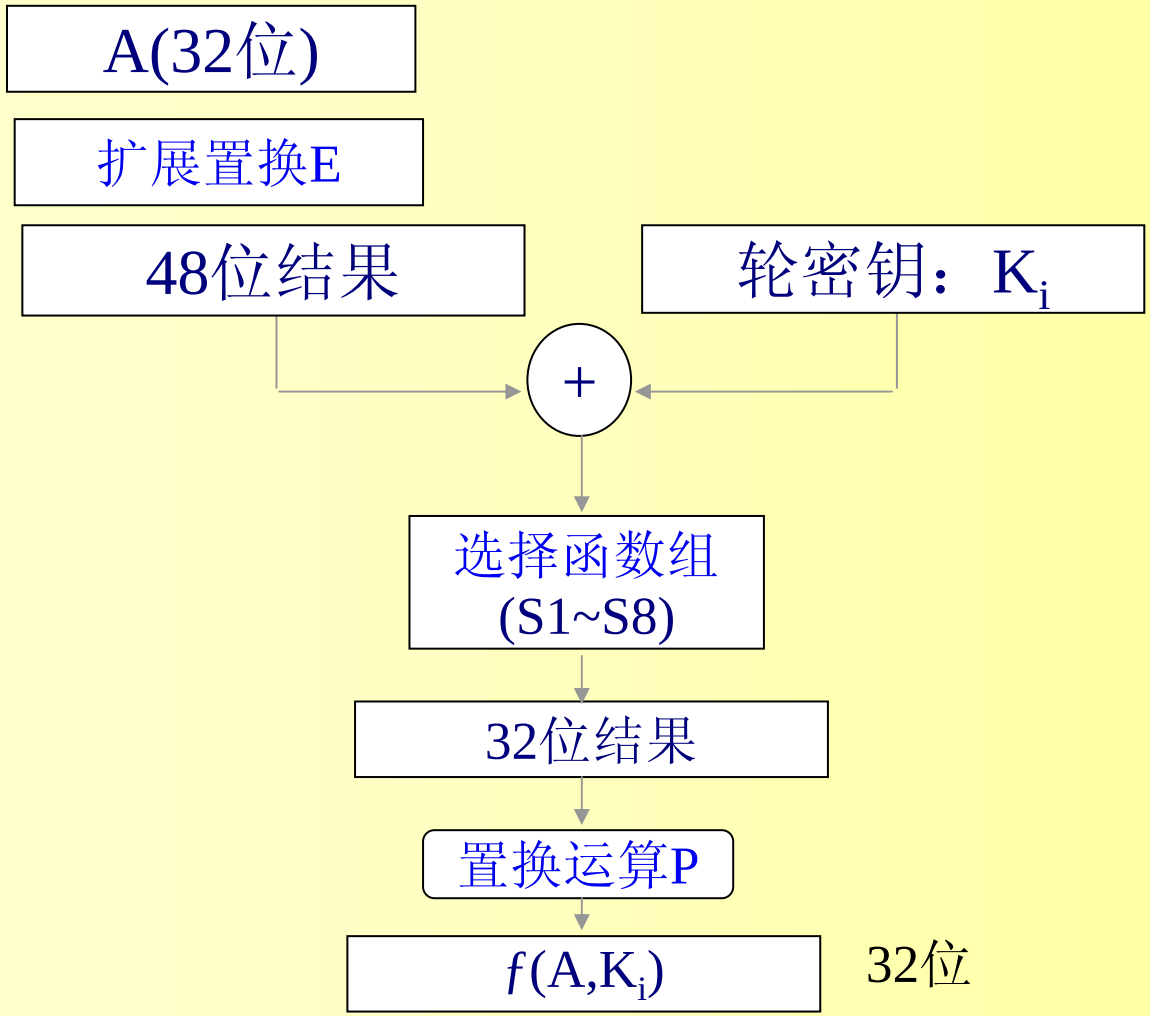
逆初始变换IP^{-1}

置换码组　输入（64位）

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

输出（64位）

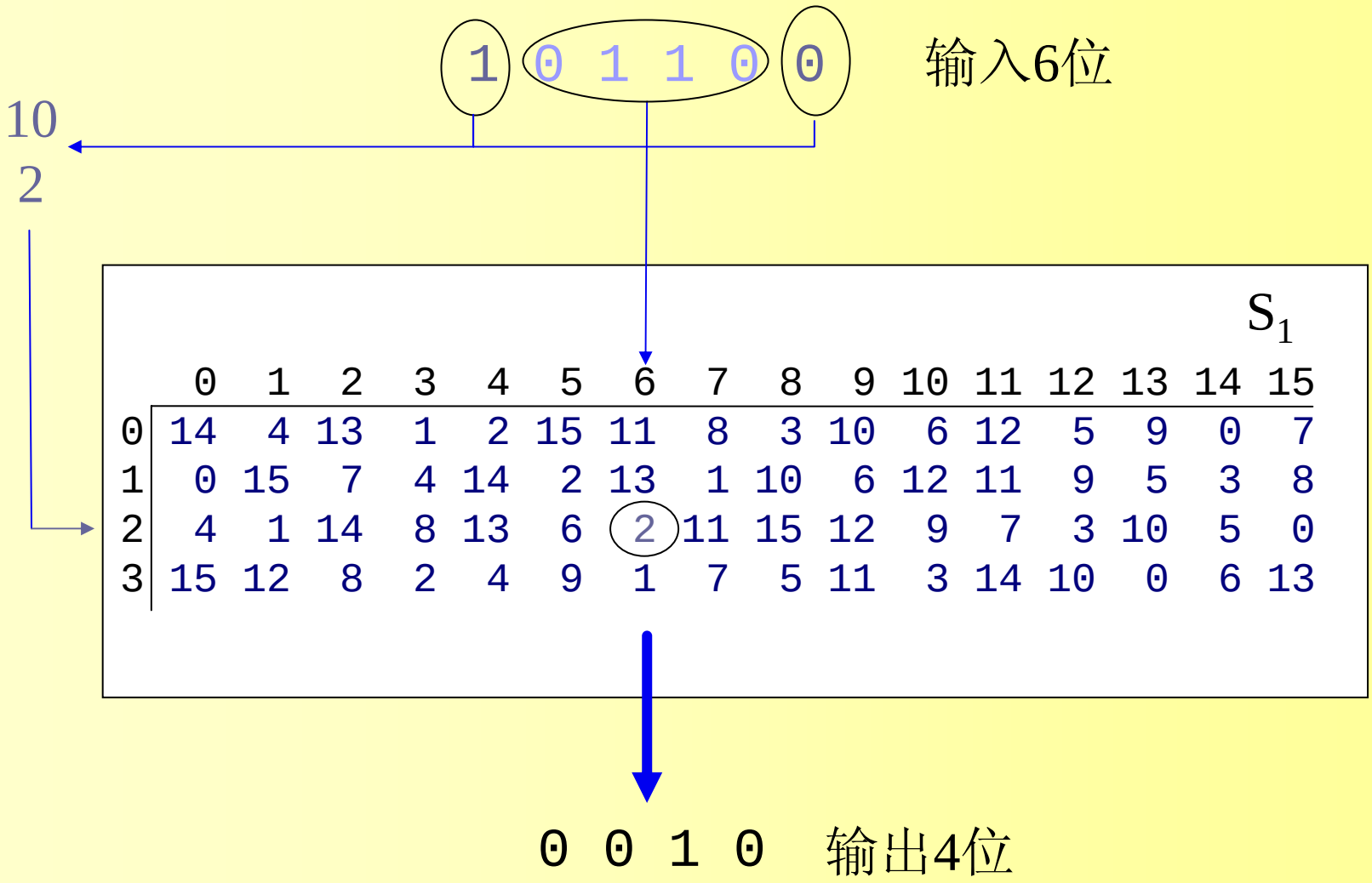
加密轮函数f(A,K_i)



扩展运算E


A
32位
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
扩展运算E
扩展运算E的结果
48位

选择运算


1 0 1 1 0 0
输入6位
10
2
S_1
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
3 15 12 8 2 4 9 1 7 5 11 3 14 10 0 6 13
0 0 1 0 输出4位

置换运算 P

选择函数的输出 （32位）

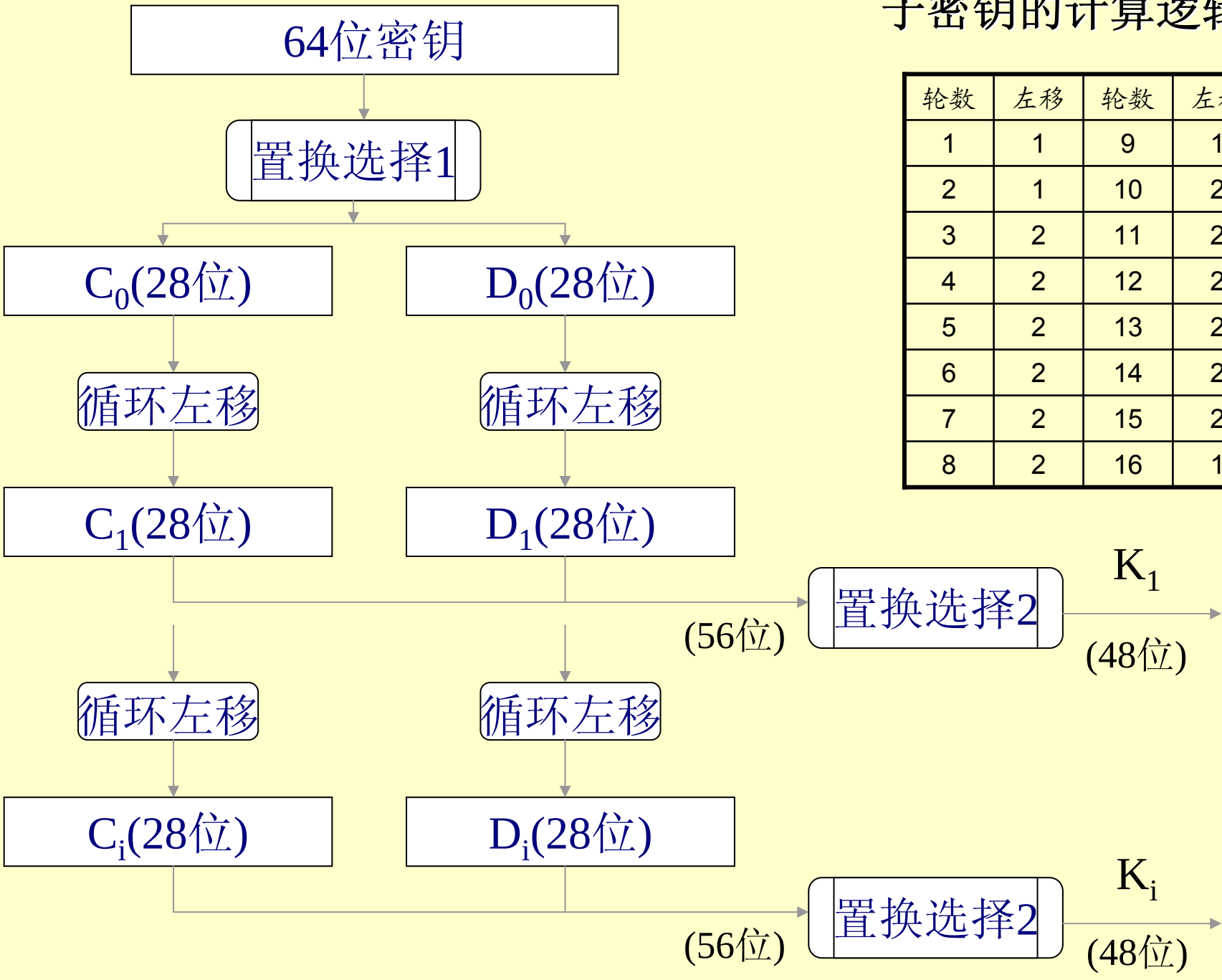
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

置换P

加密函数的结果 （32位）

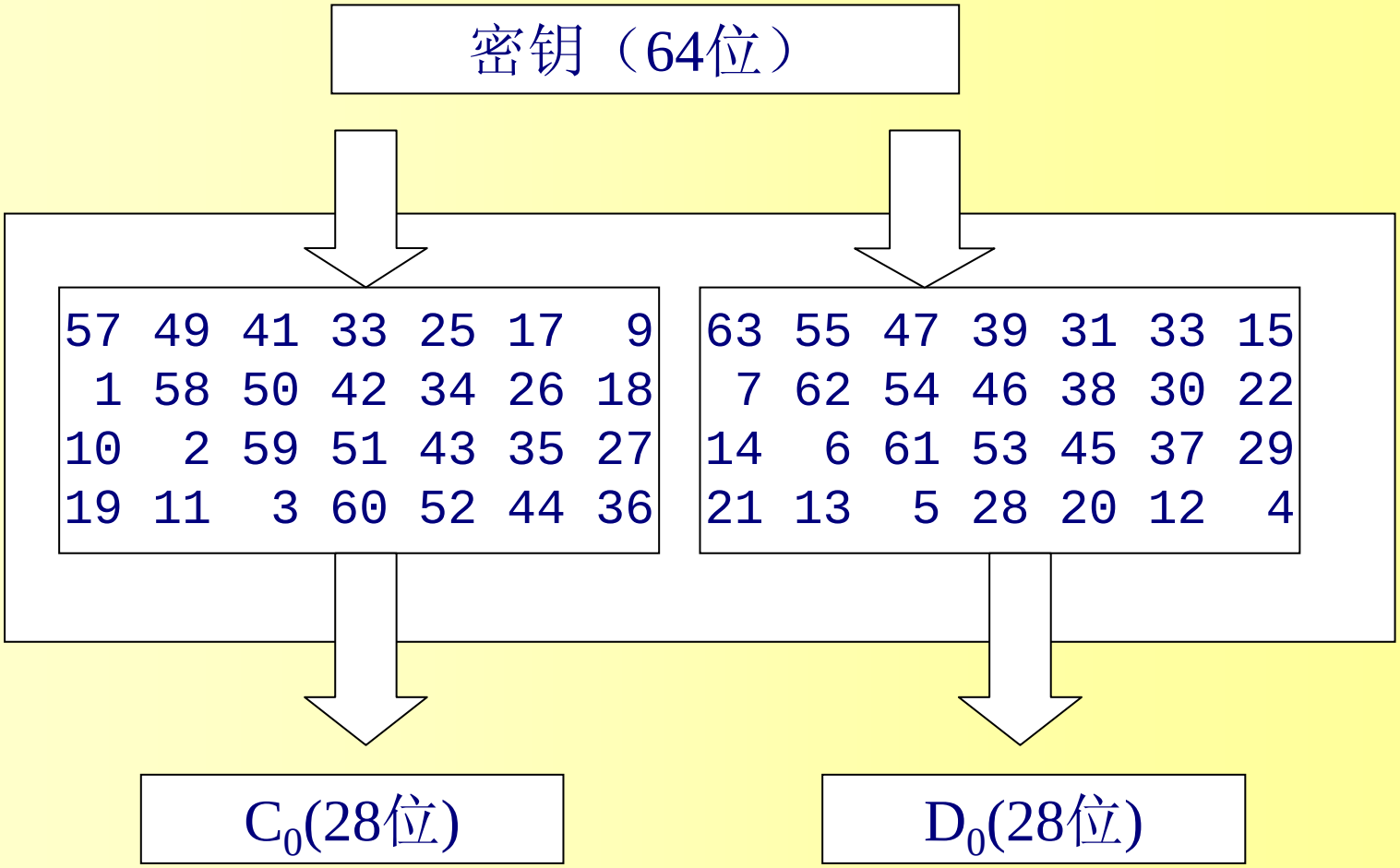
- F(A, K)， $k= k_1k_2k_3k_4k_5k_6k_7k_8$
  - $E(A) = E_1E_2E_3E_4E_5E_6E_7E_8$
  - $k_1 \oplus E_1 \| k_2 \oplus E_2) \| \ldots\ldots \| k_8 \oplus E_8$
  - $S_1(k_1 \oplus E_1) \| S_2(k_2 \oplus E_2) \| \ldots\ldots \| S_8(k_8 \oplus E_8)$
  - $P(S_1(k_1 \oplus E_1) \| S_2(k_2 \oplus E_2) \| \ldots\ldots \| S_8(k_8 \oplus E_8))$
- $F(A, K) = P(S(E(A) \oplus K))$

子密钥的计算逻辑

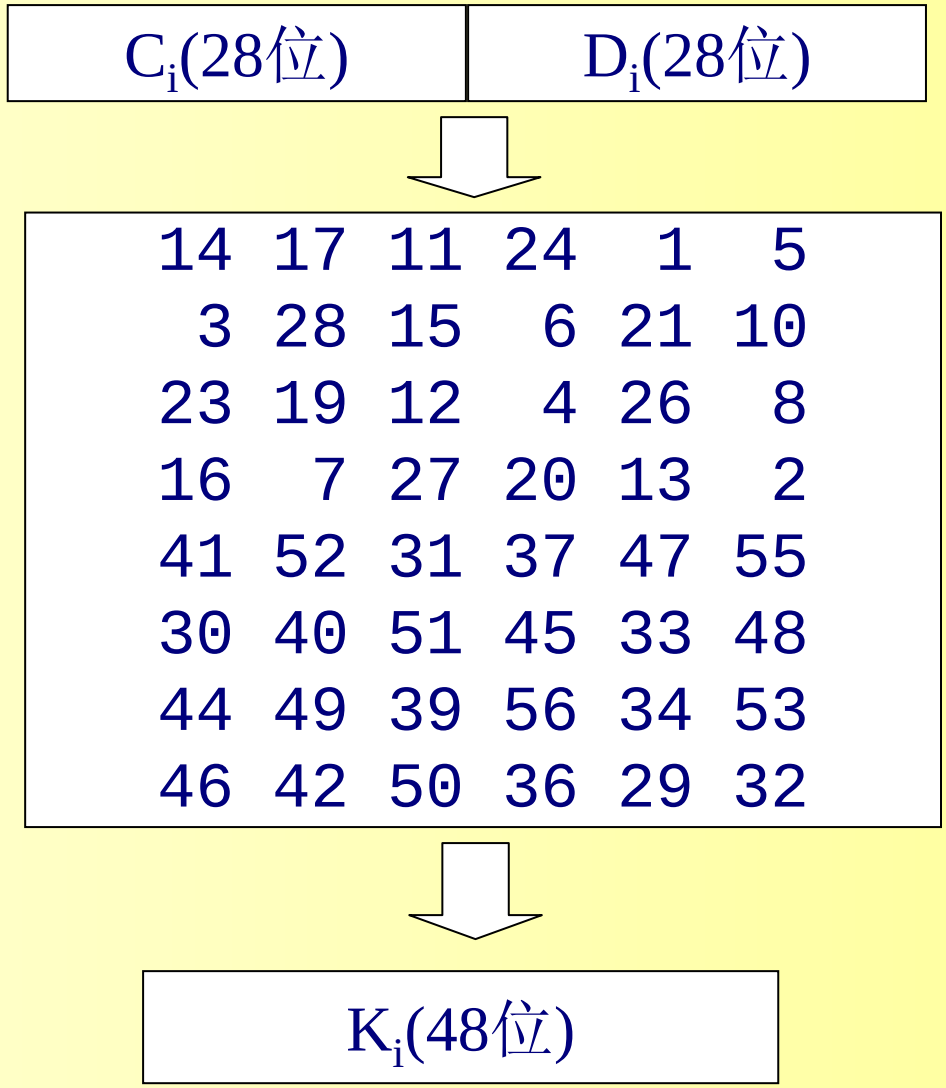


轮数	左移	轮数	左移
1	1	9	1
2	1	10	2
3	2	11	2
4	2	12	2
5	2	13	2
6	2	14	2
7	2	15	2
8	2	16	1

置换选择1


密钥（64位）
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 33 15
7 62 54 46 38 30 22
14 6 61 53 45 37 29
21 13 5 28 20 12 4
C0(28位)
D0(28位)

置换选择2


C_i(28位)
D_i(28位)
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
K_i(48位)

4.3 分组密码的统计测试原理

- 统计测试的目的：断定算法产生的输出是否在统计上难以与真随机数据区分开来。
  - 数据变换的有效性测试原理
  - 算法对明文的扩散性测试原理
  - 密钥更换的有效性测试原理

数据变换的有效性测试原理

- 频数检验：测试密文的“0”，“1”平衡性
- 若输出的密文是随机的，则首先应该有较好的“0”，“1”平衡性。
  - 设待测分组密码算法的分组长度为n比特
  - 待测密文文件含有F个密文分组
  - 统计这F个分组中汉明重量为i的分组个数，记为F_i，
  - 与其期望数$E_i=C_n^i \times F/2^n$进行X^2拟合检验，
  - 将计算结果与自由度为n，显著性水平为5%的X^2阈值相比较，来判断F_i是否符合二项分布B(n, 1/2)。

数据变换的有效性测试原理

- 跟随性检验：跟随性检验测试分组中相邻比特的出现情况。
  - 设待测分组密码算法的分组长度为n比特;
  - 对分组取分组中相邻元素的模2加;
  - 然后统计模2加后的分组中重量为i的分组个数，记为G_i;
  - 与其期望数$E_i=C_n^i \times F/2^{n-1}$进行拟合 X^2检验;
  - 将计算结果与自由度为n-1，显著性水平为5％的 X^2阈值相比较，来判断G_i是否符合二项分布B(n-1, 1/2)。

数据变换的有效性测试原理

- 明密文独立性测试： 明密文独立性测试主要是测试密文是否有不依赖于明文统计特性的性质。
  - 主要考虑两方面的测试：一方面，如果明文具有某种明显的统计特性，算法具有较好的明密文独立性，则明文与其对应的密文的距离应是随机的；
  - 另一方面，如果明文是随机的，则明文与其对应的密文的距离也应是随机的。
  - 设待测分组密码算法的分组长度为n比特，明文文件含有F个分组，则其对应的密文也含有F个分组。
  - 设明文分组集为$P=\{p_0, p_1, \ldots\ldots, p_{F-1}\}$，密文分组集为$C=\{c_0, c_1, \ldots\ldots, c_{F-1}\}$，记录相应的明密文距离$D=W(p_i+c_i)$，$0 \leqslant i \leqslant F-1$。
  - 统计D中汉明重量为i的分组数，记为H_i。
  - $E_i=C_n^i \times F/2^n$进行拟合X^2检验。将计算结果与自由度为n，显著性水平为5％的X^2阈值相比较，来判断H_i是否符合二项分布B(n，1/2)。

算法对明文的扩散性测试原理

- 从数据变换的有效性考虑，一个分组密码算法对明文的变化应是敏感的，即明文的雪崩现象。根据分组密码测度中的**严格雪崩准则，改变明文分组的任一比特，应导致密文分组中大约一半比特的变化**。
- 设待测分组密码算法的分组长度为n比特，
- 设明文分组 $P=\{p_0, p_1, \ldots\ldots, p_{n-1}\}$, $p_i \in \{0,1\}$
- 设密文分组$C=\{c_0, c_1, \ldots\ldots, c_{n-1}\}$, $c_i \in \{0,1\}$
- 在固定密钥的情况下，每次改变P中的某个p_i，则有
- $P_i=\{ p_0, p_1, \ldots, p_i \oplus 1, \ldots, p_{n-1}\}$，$p_i \in \{0，1)$，$0 \leqslant i \leqslant n-1$
- 得到其相应的密文
- $C_i=\{c^i_0, c^i_1, \ldots\ldots, c^i_{n-1}\}$，$c^i_k \in \{ 0，1)$，$0 \leqslant k \leqslant n-1$, $0 \leqslant i \leqslant n-1$
- 计算C与C_i之间的汉明距离$d_i= W(C+C_i)$，$0 \leqslant i \leqslant n-1$
- 根据随机性要求，密文之间的距离分布应符合二项分布B(n, 1/2)。

密钥更换的有效性测试原理

- 从密钥更换的有效性考虑，一个分组密码算法对密钥的变化应是敏感的，即密钥的雪崩现象。
- 根据分组密码测度中的严格雪崩准则，改变密钥中任一比特，应导致密文分组中大约一半比特的变化。

密钥更换的有效性测试原理

- 设待测分组密码算法的分组长度为n比特，密钥长度为m比特。
- 设明文分组　$P=\{p_0, p_1, \ldots\ldots, p_{n-1}\}$, $p_i \in \{0,1\}$
- 设密文分组　$C=\{c_0, c_1, \ldots\ldots, c_{n-1}\}$, $c_i \in \{0,1\}$
- 密钥　$K=\{k_0, k_1, \ldots\ldots, k_{m-1}\}$, $k_i \in \{0,1\}$
- 在固定明文的情况下，每次改变K中的某一位，则有
- $K_i=\{ k_0, k_1, \ldots, k_i \oplus 1, \ldots, k_{m-1}\}$，$0 \leqslant i \leqslant m-1$
- 得到其相应的密文
- $C_i=\{c^i_0, c^i_1, \ldots\ldots, c^i_{n-1}\}$，$c^i_k \in \{0, 1)$，$0 \leqslant k \leqslant n-1$, $0 \leqslant i \leqslant m-1$
- 计算C与C_i之间的汉明距离$d_i= W(C+C_i)$，$0 \leqslant i \leqslant m-1$
- 根据随机性要求，密文之间的距离分布应符合二项分布B(n, 1/2)。

4.4 DES的安全性 — 关于S-盒的设计准则

- 这个问题涉及美国国家安全局(NSA)，他们修改了IBM的S-盒。虽然修改后的S－盒满足IBM原先关于S-盒的设计原则，但是，人们仍因此而怀疑NSA在S-盒中嵌入了陷门，使得NSA借助于这个陷门及56比特的短密钥可以解密DES。

DES的安全性 — 关于S-盒的设计准则

- 在1990年以前， S-盒的设计准则一直没有公布，直到差分密码分析方法发表后才公布。公布的S-盒设计准则是:
  - S-盒的每一行是0-15的一个排列;
  - 没有一个S-盒接近其输入的线性函数;
  - 如果固定输入的最左边，最右边的位固定，变换中间4位，每个可能的4位输出能得到一次。
  - 如果两个输入仅中间2位不同，则输出至少有2位不同;
  - 如果两个输入仅差1位，则输出至少有2位不同;
  - 如果两个输入前2位不同，后2位已知，则输出不同;
  - 具有非零，相同差分的32对输入中，至多有8对具有相同的输出差分;

DES的安全性 — 关于56比特的短密钥

- IBM最初提交的方案中密钥是112比特，但是，公布的DES却是56比特，坚持使用56比特短密钥是NSA的意见。这就使得人们更加相信DES的陷门之说。

- 分布式穷举攻击：1997年1月28日，美国RSA数据安全公司在Intemet上开展了一项“秘密密钥挑战”的竞赛，悬赏一万美元，破解一段DES密文。科罗拉多州的程序员R. Verser设计了一个可以通过互联网分段运行的密钥搜索程序，组织了一个称为DESCHALL的搜索行动，成千上万的的志愿者加入到计划中。第96天，即竞赛公布后的第140天，1997年6月17日晚上10点39分，美国盐湖城Inetz公司职员M. Sanders成功地找到了密钥

- 专用搜索机：1998年5月，美国电子边境基金会(EFF，Electronic Frontier Foundation)宣布，他们用一台价值20万美元的计算机改装成专用设备，56小时就破译了DES。这样，更加直接地说明了56比特密钥太短了，彻底宣布了DES的终结。

- Wiener 报道了使用流水线的技术达到
  - 每秒5000万个密钥搜索速率的芯片的设计;
  - 1993年的价格计算，10万美元的模块包含5760个密钥搜索芯片;
  - DES的搜索结果:

造价	搜索时间
$100,000	35小时
$1,000,000	3.5小时
$10,000,000	21分钟

DES的安全性 — DES的对称性

- DES算法具有对称：如果用表示按位取补，则：

$$\overline{DES_k(z)} = DES_{\bar{k}}(z)$$

- 这种对称性，使穷举攻击密钥搜索量可以减少一半。

DES的安全性 — 弱密钥

- 如果对于初始密钥k，生成的子密钥有$k_1=k_2=\ldots=k_{16}$，则称k是弱密钥。显然，$DES_k(DES_k(x))=x$。
- DES 有4个弱密钥:

01	01	01	01	01	01	01	01
1F	IF	1F	1F	1F	IF	1F	1F
E0	E0	E0	E0	E0	E0	E0	E0
FE	FE	FE	FE	FE	FE	FE	FE

DES的安全性 — 半弱密钥

- 如果k1、k2 满足

$$DES_{k2}(DES_{k1}(x)) = DES_{k1}(DES_{k2}(x))$$

则称k1、k2是互逆的半密钥

- DES 有12个半密钥

01 FE 01 FE 01 FE 01 FE

FE 01 FE 01 FE 01 FE 01

1F E0 1F E0 1F EO IF E0

E0 IF E0 1F E0 1F E1 IF

01 E0 0I E0 0I E0 0I E0

……

DES的安全性 — 代数性质

- 设E是一个密码算法，如果对任意的密钥k1, k2，都存在密钥k3，使得对任意的明文x 有:

$$E_{k1}(E_{k2}(x)) = E_{k3}(x)$$

则称E是闭合的。

- 显然，对闭合的加密算法企图通过二重加密来增强体制的安全性是无效的。
- 设(P，C，E，D，K)是一个密码体制，如果加密算法E是闭合的，且C=P，D=E，则$\{ E_k \mid k \in K \}$在复合加密运算下是一个群。
- 1992年，K. W. Campbell，M. J. Wiener证明了DES不是一个群。
- 正因以上结论，**二重DES，三重DES有一定的价值**，其穷举攻击的安全性虽有所改进，然而并非2倍，3倍的密钥量那么坚强。

Double-DES and Triple-DES

- $C = DES(K_2, DES(K_1, M))$ $\quad K = K_1K_2$
- $C = DES(K_1, DES^{-1}(K_2, DES(K_1, M)))$ $\quad K = K_1K_2K_1$
- $C = DES(K_3, DES(K_2, DES(K_1, M)))$ $\quad K = K_1K_2K_3$

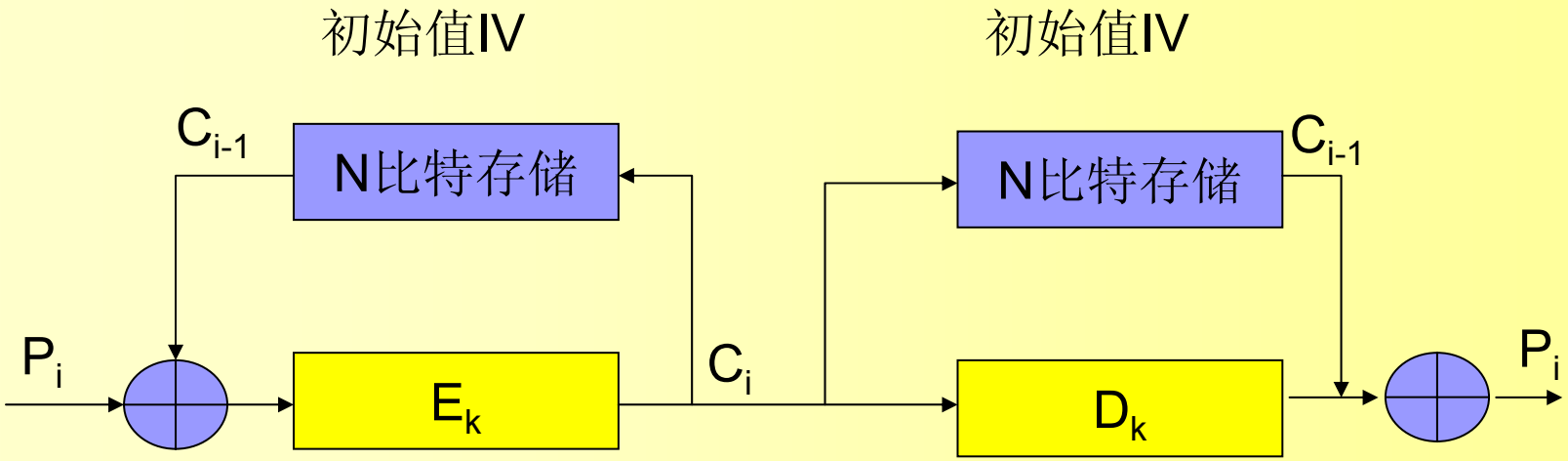
4.6 分组密码的操作方式

- 电子密码本ECB(Electronic Code Book)
- 密码分组链接方式CBC (Cipher Block Chaining)
- 密码反馈方式CFB (Cipher FeedBack mode)
- 输出反馈方式 OFB (Output FeedBack mode)

电子密码本ECB工作方式的问题

1	2	3	4	5	6	7	8	9	10	11	12	13
时间标记	发送银行		接收银行	储户姓名						储户帐号		存款金额

CBC（密码分组链接）



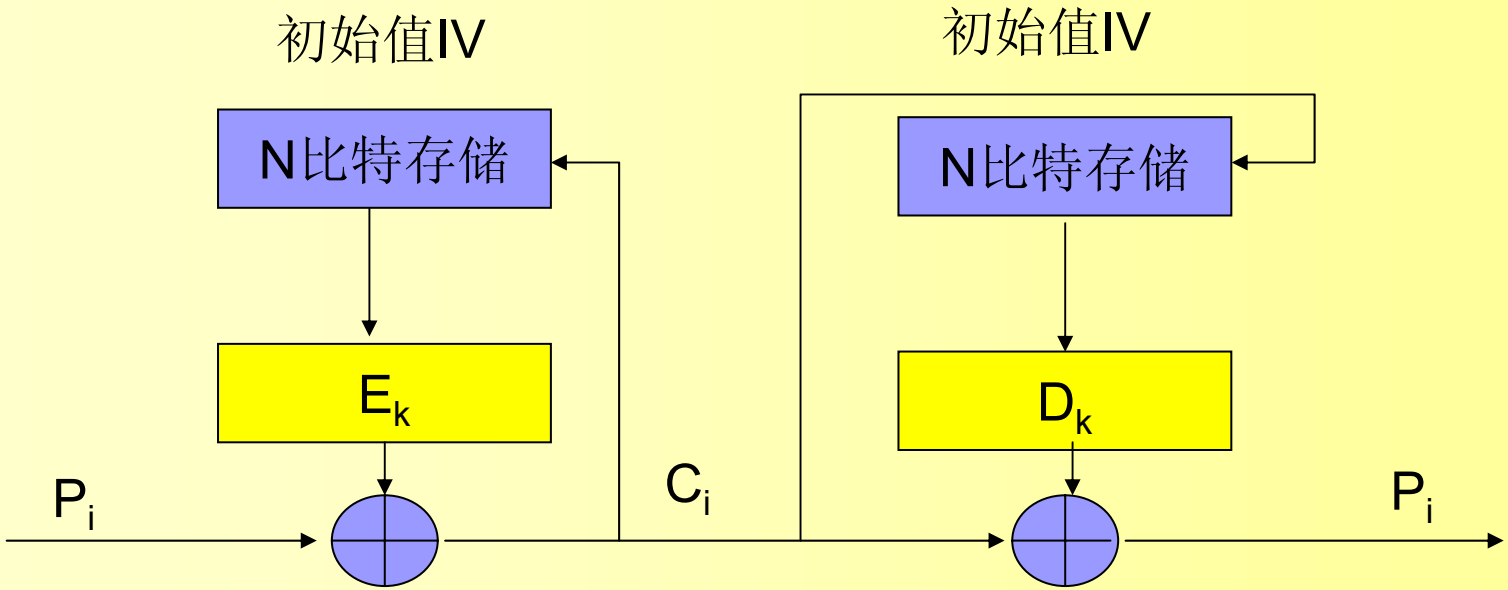
$C_i=E_k(P_i \oplus C_{i-1})$

$C_1=E_k(P_1 \oplus IV)$

$P_i= C_{i-1} \oplus D_k(C_i)$

1. 明文的1位错误将影响本分组开始向后的所有明文分组
2. 扩散：密文分组的1位错误将影响该分组和下一个分组
3. 同步：密文位流中增减1位将影响影响所有后继分组

CFB (CipherFeedBack 密码反馈)



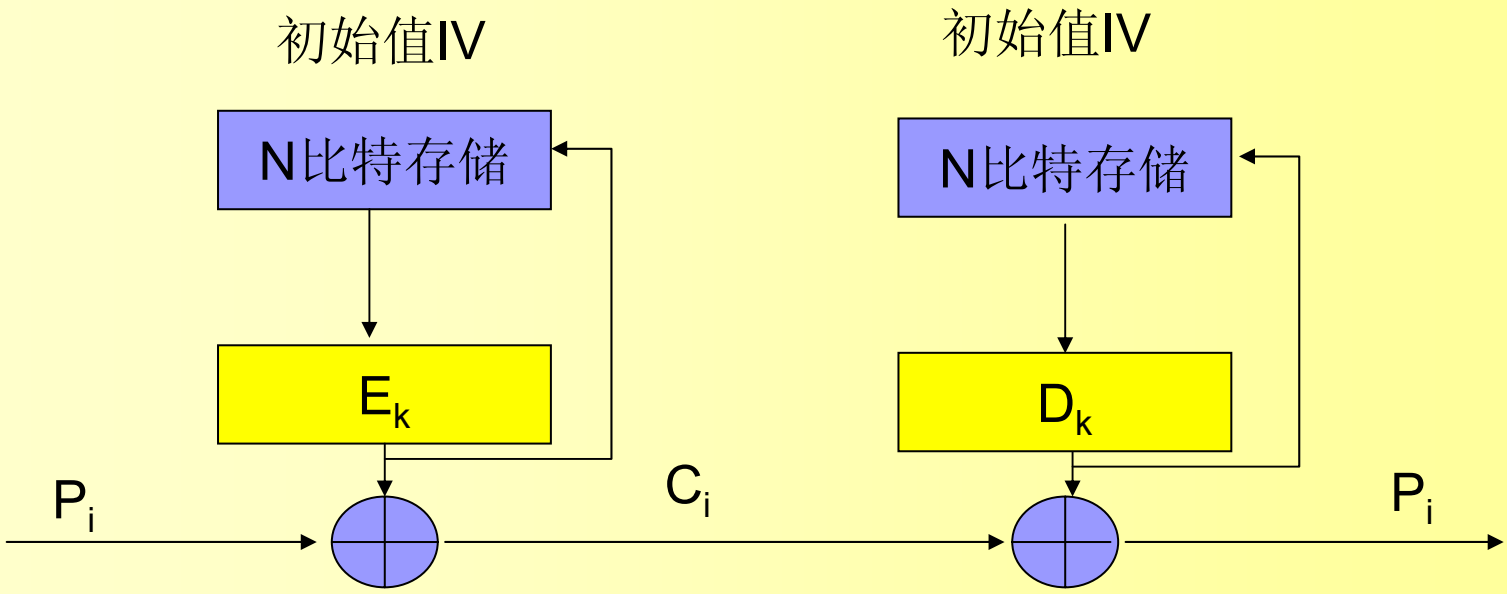
$C_i = P_i \oplus E_k(C_{i-1})$

$C_1 = P_1 \oplus E_k(IV)$

$P_i = C_i \oplus E_k(C_{i-1})$

1. 明文的一个错误将影响后面所有的密文及解密过程中的逆。
2. 密文的1位错误将影响密文的一个位，但影响后面的分组，直到密文从移位寄存器的另一端移出。

OFB (OutputFeedBack 输出反馈)

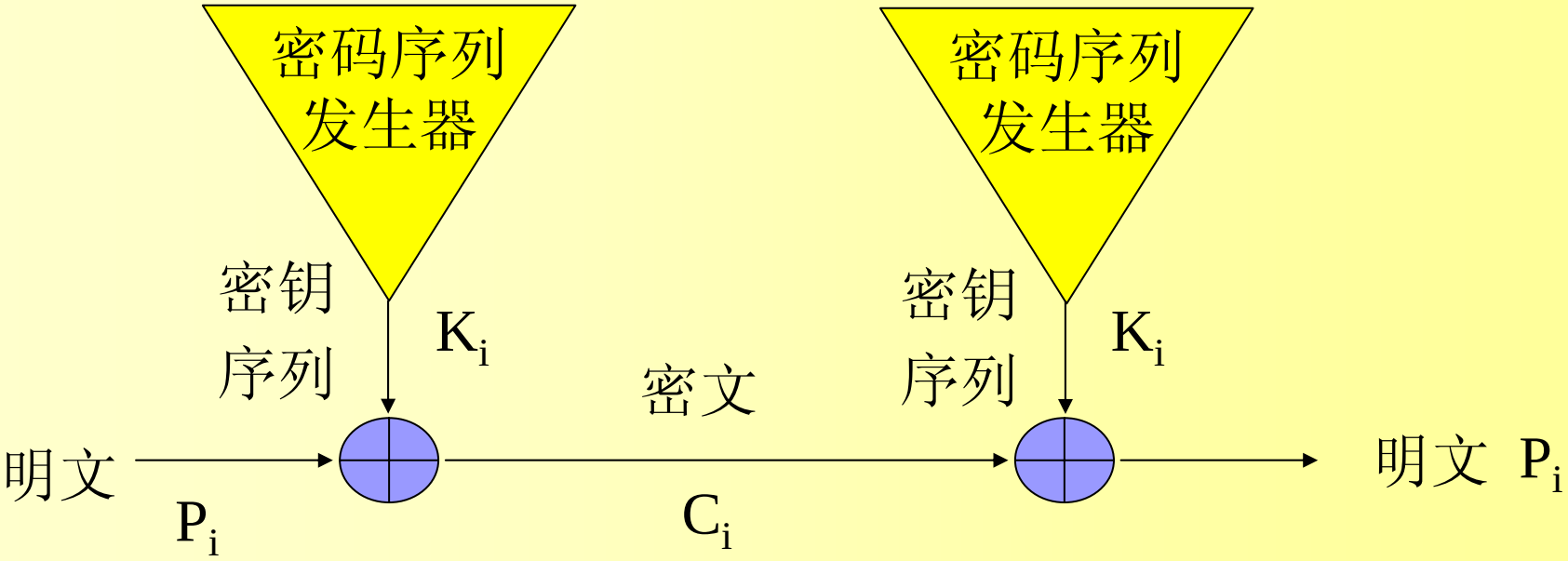


$C_i=P_i \oplus S_i$, $S_i=E_k(S_{i-1})$,
$S_0 = IV$

$P_i=C_i \oplus S_i$ $S_i=E_k(S_{i-1})$

1. OFB模式没有错误扩散。密文中单个位错误只引起恢复明文的单个位错误。这一点对数字化模拟传输特别有用。
2. 不同步是致命的，OFB必须有检测和恢复同步的机制。

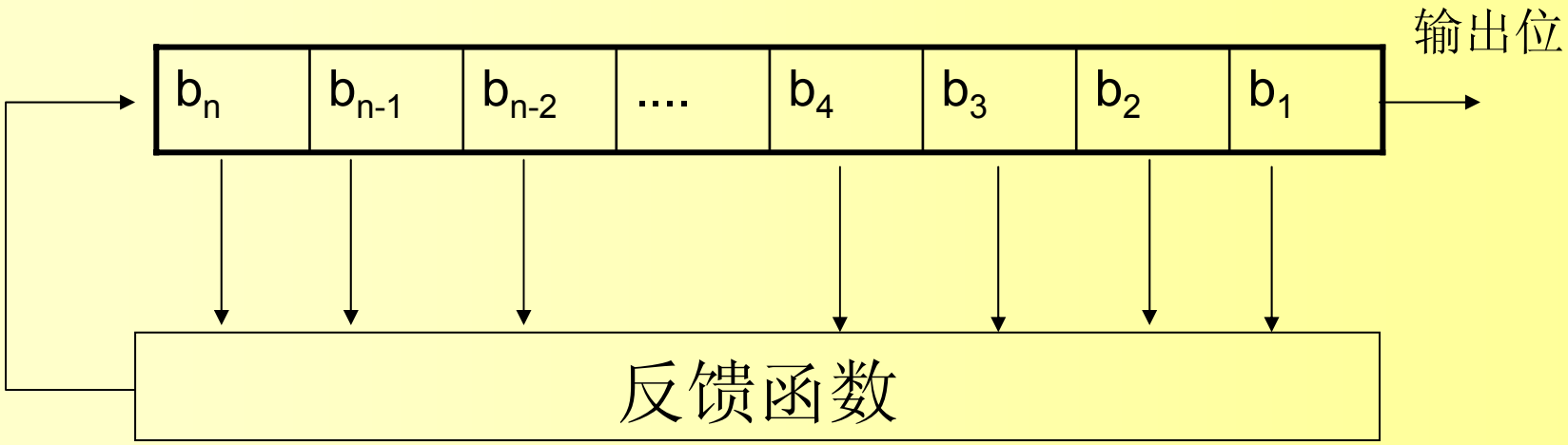
序列密码算法



- $C_i = P_i \oplus K_i$
- $P_i = C_i \oplus K_i$

反馈移位寄存器

feedback shift register



5. 其他分组密码算法

5.1 AES

- 1997年美国NIST发起了一场推选用于保护敏感的无密级的信息的加密算法的活动
  - 评估分为三大项：安全性、成本、算法和实现特性
- 1998年NIST宣布选定了15个候选算法并提请全世界的密码学界协助分析这些候选算法。
- 1999年8月20日选定了MARS、RC6、RIJNDAEL、Serpent、Twofish等5个算法作为参加决赛的算法。
- 2000年10月2日， RIJNDAEL最终获胜
  - RIJNDAEL由比利时密码学家Joan Daemen 和Vincent Rijmen设计的。

- RIJNDAEL算法是一个迭代型密码
  - 分组长度和密钥长度都可变;
  - 各自可以是128、192、256比特;
  - RIJNDAEL的明文分组成为状态，所有的操作都在状态间进行;
  - 状态可以用以字节为元素的矩阵表示;
  - 矩阵有4行，列数N_b=分组长度 / 32。

RIJNDAEL

- RIJNDAEL的计算部件
  - 加密钥A_K
  - 字节代替$S_{u,v}$
  - 行移位R_C
  - 混合H_M
  - $T_0 = A_0$
  - $T_i = A_i H_M R_C S_{u,v}$， i=1,……, N-1
  - $T_N = A_N R_C S_{u,v}$
- $AES = T_N \bullet T_{N-1} \bullet \bullet \bullet T_1 \bullet T_0$

AES — 加密钥(AddRoudKey)

a_{ij} 为GF(2^8)中的元素，

a_{ij} ＋ k_{ij}为GF(2^8)中的运算

$$a=(a_{ij})_{4\times \mathrm{Nb}}\begin{pmatrix} a_{11} & \cdots & a_{1Nb} \\ \vdots & \ddots & \vdots \\ a_{41} & \cdots & a_{4Nb} \end{pmatrix},\quad \mathrm{kj}=(k_{ij})_{4\times \mathrm{Nb}}\begin{pmatrix} k_{11} & \cdots & k_{1Nb} \\ \vdots & \ddots & \vdots \\ k_{41} & \cdots & k_{4Nb} \end{pmatrix}$$

$$Aj(a)=(a_{ij})+k_i=\begin{pmatrix} a_{11}+k_{11} & \cdots & a_{1Nb}+k_{Nb} \\ \vdots & \ddots & \vdots \\ a_{41}+k_{41} & \cdots & a_{4Nb}+k_{Nb} \end{pmatrix}$$

AES — S-盒(ByteSub)

- 逆元素变换：在有限域 $GF(2^8)=F_2[x]/(m(x))$ 进行

$$t(a)=\begin{cases}0, a=0\\ a^{-1}, a\neq 0\end{cases}$$

- 仿射变换：$a \rightarrow L_{u,v}(a)$，在环 $GF(2^8)=F_2[x]/(x^8+1)$ 进行

$$S_{u,v}(a(x)) = (u(x)\ (a_{ij}^{-1} \bmod m(x)) + v(x) \bmod x^8+1)_{4\times Nb}$$

- **ByteSub在不同的数学结构上进行，保证了AES的非线性。**

有限域

- 具有相同阶数的有限域是同构的。
- 对于每一个素数p^n幂，恰好有一个有限域，用$GF(p^n)$。
- 设p是素数
  - { 0, 1, 2, …, p-1 } 按照模p加法和乘法构成域,记为F_p。
  - F_p上的多项式全体构成环。
  - 设 f(x)是F_p上的多项式，则$F_p[x] / f(x)$ 构成环。
  - 设 f(x)是上的n次不可约多项式，则$F_p[x] / f(x)$ 构成域$GF(p^n)$ 。

- 设p=2
  - 寻找上的8次不可约多项式f(x).
  - 8位二进制串，按照$F_2[x] / f(x)$上的运算构成域$F_{2^8}= GF(2^8)$。
- $GF(2^8)$上加法定义为按位异或;
- $GF(2^8)$上定义乘法如下:
- $(a_7a_6a_5a_4a_3a_2a_1a_0)\ (b_7b_6b_5b_4b_3b_2b_1b_0) =(c_7c_6c_5c_4c_3c_2c_1c_0)$
  - $a(x)= a_7x^7+a_6x^6+a_5x^5+a_4x^4+a_3x^3+a_2x^2+a_1x+a_0$
  - $b(x)= b_7x^7+b_6x^6+b_5x^5+b_4x^4+b_3x^3+b_2x^2+b_1x+b_0$
  - $c(x) = c_7x^7+c_6x^6+c_5x^5+c_4x^4+c_3x^3+c_2x^2+c_1x+c_0$
  - $m(x)=x^8+x^4+x^3+x+1$
  - $c(x) = a(x)\ b(x)\ \mod\ f(x)$
  - f(x)是不可约多项式，
- $GF(2^8)$按照这样的定义的加法和乘法构成域。
- 如果选择$m(x) = x^8+1$, $GF(2^8)$按照这样的定义的加法和乘法构成环。

AES —列混合变换(MixColumn)

- 将状态矩阵的一个列看成是GF(2^8)上的多项式:

  $t(x) = t_3+x^3+t_2x^2+t_1x+t_0$

- 列混合变换表示为

  $a(x) = \{03\}x^3+ \{01\}x^2+ \{01\}x+\{02\}$

  $d(x) = a(x)\ t(x) \mod (x^4+1)$

$$\begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} t_0 \\ t_1 \\ t_2 \\ t_3 \end{pmatrix}$$

记M=$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}$,T为状态矩阵

$H_M(T) = MT$

AES —行移位变换（ShiftRaw）

- 第0行不动;
- 第1行循环左移1个字节;
- 第2行循环左移2个字节;
- 第3行循环左移3个字节;

5.2 IDEA

- International Data Encryption Algorithm
- 瑞士联邦工学院的来学佳、James Massey研制的分组密码
- 128位密钥，64位分组。
- 采用8轮迭代，每轮迭代中有三种运算：1)逐位异或，2)模65536相加，3) 模65537乘法。

5.3 Blowfish

- Bruce Schneier设计。
- 密钥长度从32 bit到448 bit。
- 在32位的微处理器上加密数据，每个字节用18个时钟周期。
- S盒依赖于密钥。
- 密码分析比较困难。

5.4 RC5

- Ron Rivest 研制的对称分组密码
- 适合在微处理器上运行
- 适应在不同字长的机器上执行（16，32，64）
- 可变的循环次数（0到255）
- 可变长度的密钥（0到2040 bit）

- RC5-32/12/16：RC5-字长/循环次数/密钥字节长度