计算机系统信息安全

南京大学计算机系　　黄皓 教授

2013年9月17日

开场白 —— 一个电话骗局

中央电视台《今日说法》报道

一个电话欺骗（中央电视台今日说法）

第1幕 — 在火车站

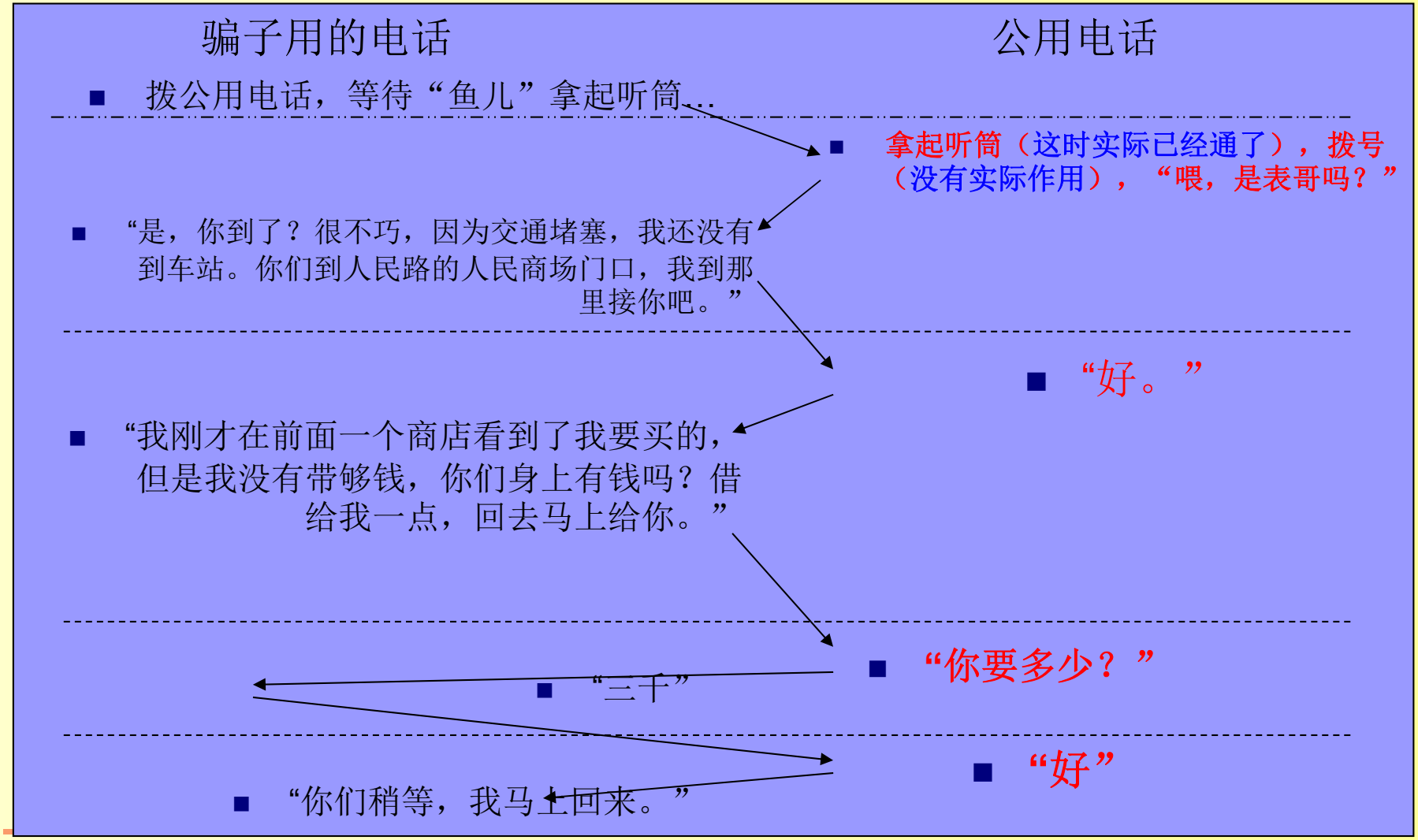
- 一个计划到上海的给孩子看病的旅客到达了上海站。
- 他到公用电话处，拿起电话听筒，拨打在上海的亲戚的电话。
- 旅客：“喂，是表哥吗？”
- 听筒传出：“是，你到了？很不巧，因为交通堵塞，我还没有到车站。你们到人民路的人民商场门口，我到那里接你吧。”
- 旅客：“好。”

一个电话欺骗（中央电视台今日说法）

第2幕 — 在人民商场门口

- 旅客在人民商场的门口等待。
- 一个中年人向正在等待的旅客走来。
- “我是你表哥委托来接你们的。”
- 交谈后，在准备回表哥家时：
  - 中年人：“我刚才在前面一个商店看到了我要一直想买的非常紧俏的相机，今天好不容易看到了，但是我没有带够钱，你们身上有钱吗？借给我一点，回去马上给你。”
  - “你要多少？”
  - ”三千“
  - “好”
  - 你们稍等，我马上回来。
- 几个警察迅速抓住了这个快速离开的中年人。
- 旅客过去跟警察说：“这是我表哥的朋友，你抓错人了吧？”

- 骗子设法知道公用电话的号码。
- 不断拨这个公用电话的号码。
- 拨通后等待别人：拿听筒、不听是否拨号音就直接拨电话的人。
- 骗子可以听到，“鱼儿”的拨号的键盘音。等待他/她的开始说话。



课程内容

1. 对称密码学
2. 非对称密码学
3. 密码协议
4. 公开密钥基础设施
5. 安全模型
6. 安全操作系统
7. 安全数据库
8. IP安全
9. Web安全
10. 安全电子交易
11. 安全邮件
12. 网络防火墙技术
13. 入侵检测技术
14. 安全评估技术

参考书

1. 讲稿和各章讲稿所列的参考文献
2. Wenbo Mao(毛文波)，现代密码学理论与实践，电子工业出版社，2004年7月。
3. **王立斌，黄征译，计算机安全学—安全的艺术与科学，电子工业出版社，2005年5月。**
4. Allen Harper, Shon Harris等著， 杨明军，韩智文，程文俊译， 灰帽黑客，清华大学出版社，2012年11月，第1版。

第1章　对称密码学

参考文献

1. Wenbo Mao(毛文波)，现代密码学理论与实践，电子工业出版社，2004年7月。
2. 吴文玲，冯登国，张文涛，分组密码的设计与分析，清华大学出版社，2009年10月。
3. 冯登国，密码分析学，清华大学出版社，2000年8月。
4. Bruce Chneier, 应用密码学，机械工业出版社，2000年1月。

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再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”。

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)。

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”

① 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；

② Bob告诉Alice自己对x的奇偶性猜测；

③ Alice告诉Bob x的值；

④ Bob验证f(x)，并察看他所做的猜测是正确或错误。

只能是猜测，因为f(x)是单向向函数，无法从函数值计算自变量。

Alice只能如实告诉，因为Alice无法找到另一个值y(奇偶性与x不同)，f(y)=f(x)。

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来计算对f(x) (第4步)，并与Alice在第1步发送的结果一样，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（M）=C
D_K（C）=M
D_K（E_K（M））=M

E_{K1}（M）=C
D_{K2}（C）=M
D_{K2}（E_{K1}（M））=M

1. 基本概念—对称算法和非对称算法

- **对称算法** 加密密钥能够从解密密钥中推算出来，反过来也成立。
- **公开密钥算法** 公开密钥算法用作加密的密钥不同于用作解密的密钥，而且解密密钥不能根据加密密钥计算出来。

1. 基本概念—密码分析

- 密码分析学是在不知道密钥的情况下。恢复出明文的科学。
- 对密码进行分析的尝试称为攻击。
- 密码分析的一个基本假设：密码分析者已有密码算法及其实现的全部详细资料。在实际的密码分析中并不总是有这些详细信息的——应该如此假设。如果其他人不能破译算法，即便了解算法如何工作也是徒然，如果连算法的知识都没有，那就肯定不可能破译它。

（1）唯密文攻击

- *密码分析者有一些消息的密文*
  - 这些消息都用同一加密算法加密
  - 密码分析者的任务是恢复尽可能多的明文
  - 或者最好是能推算出加密消息的密钥来
  - 已知：$C_1=E_K（P_1）$，$C_2=E_K（P_2）$，……，
  - 推导出：P_1，P_2，…，

（2）已知明文攻击

- 密码分析者不仅可得到一些消息的密文，而且也知道这些消息的明文。
  - 分析者的任务就是用加密信息推出用来加密的密钥或导出一个算法，此算法可以对用同一密钥加密的任何新的消息进行解密。
  - 已知：P_1，$C_1=E_k（P_1）$，P_2，$C_2=E_k（P_2）$，…，P_i，$C_i=E_k（P_i）$
  - 推导出：密钥k，或从$C_{i+1}= E_k（P_{i+1}）$推出P_{i+1}的算法。

（3）选择明文攻击

- 分析者不仅可得到一些消息的密文和相应的明文，而且他们也可选择被加密的明文。
  - 这比已知明文攻击更有效。因为密码分析者能选择特定的明文块去加密，那些块可能产生更多关于密钥的信息，分析者的任务是推出用来加密消息的密钥或导出一个算法，此算法可以对用同一密钥加密的任何新的消息进行解密。

（4）选择密文攻击

- 密码分析者能选择不同的被加密的密文，并可得到对应的解密的明文，例如密码分析者存取一个防窜改的自动解密盒，密码分析者的任务是推出密钥。
  选择：C_1，$P_1=D_k（C_1）$，C_2，$P_2=D_k（C_2）$，…，C_i，$P_i=D_k（C_i）$，
  推导出： k。

- 最好的算法是那些已经公开的，并经过世界上最好的密码分析家们多年的攻击，但还是不能破译的算法。
- 美国国家安全局对外保持他们的算法的秘密，但他们有很好的密码分析家在内部工作，他们互相讨论他们的算法，通过执著的审查发现他们工作中的弱点。

1. 基本概念— 密码学目标

- 机密性
- 完整性
- 认证
- 不可抵赖性、公证性

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

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

密码体制：(P, C, K, E, D)

2.1 移位密码

- 令P=C=K=Z_{26}。对0≤ k≤ 25，x,y,k ∈Z_{26}，定义

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

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

这里a对应0记为a^f、b对应1记为b^f、……、z对应25记为z^f；
两个字母的运算$x+k \bmod 26$表示$x^f+k^f \bmod 26$所对应的字母。

- 著名的凯撒密码就是一种移位密码，k=3;

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
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

- 移位密码的密钥数量为25。

2.2 代换(substitution)密码

单表代换

- 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$26!-1\approx 7.21 * 10^{26}$
- 著名的DES算法的密钥数量为$2^{56} \approx 7.21 * 10^{16}$

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): cipher

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

- 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$I_c(x)$。n是总的字母数，f_i是第i个字母出现的次数，i=0,…,25。
- 如果取出的两个字母都是第i个字母，有种$\binom{f_i}{2}$可能。

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

- 假定Y=$y_1y_2\ldots y_n$是多表密码算法加密后的密文，如果猜测密钥的长度是d，则将Y分成d个密文子串：

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

- 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 $I_c(z_i)=26/26^2\approx 0.038$

密钥字的确定 — 重合互指数法

- 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

- 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：$Y_1, Y_2, \ldots, Y_d$,。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

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

这里假定在英文文件中，字母A出现的概率是p_0, B出现的概率是p_1, ……，z出现的概率是p_{25}，$p_{-1}=p_{25}$, $p_{-2}=p_{24}$, …… 。上式中的加减法运算取模 mod 26。

密文字母A是由密钥k_i加密得到的，字母编号0, 那么相应的明文字母编号就是$-k_i$。因此密文字母A出现的概率是p_{-ki},密文字母B出现的概率是p_{-kj}

- 我们可以得到：

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

- 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

- 固定Y_i，考虑长度为d的密钥字 $e_i = (i, i, \ldots\ldots, i)$, $i=0,\ldots25$。
- 用密钥字e_i加密Y_j，分别得到Y_j^i，$i=0,\ldots\ldots,25$。
- 计算 $MI_c(Y_i, Y_j^k)$, $0 \le k \le 25$。
- 当k等于Y_i与Y_j的相对误差的时候， $MI_c(Y_i, Y_j^k)$的值接近于0.065。
- 重合互指数为0.065的K值就是Y_i与Y_j的相对误差。

密钥字的确定 — 重合互指数法

- 设$k=(k_1,k_2,\ldots,k_m)$为密钥字；
- 固定Y_1 (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。

作业

- 编写一个破解程序，完成下列工作之一：
  1. 猜测密钥长度，用明文字母频率破解密钥。
  2. 用重合指数法破解密钥字长度。
  3. 用重合互指数法破解密钥。

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
  - $e_\pi(x_1,x_2,\ldots, x_m)=(x_{\pi(1)}, x_{\pi(2)},\ldots, x_{\pi(m)})$
  - $d_\pi(y_1,y_2,\ldots, y_m)=(y_{\pi-1(1)}, y_{\pi-1(2)},\ldots, y_{\pi-1(m)})$

- 置换密码与替代密码的区别
  - 替代密码的明文字母集合与密文字母集合往往是不同的。
  - 置换密码明文字母集合与密文字母集合完全相同。

Enigma —— 诞生

- 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 — 诞生

- 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 — 诞生

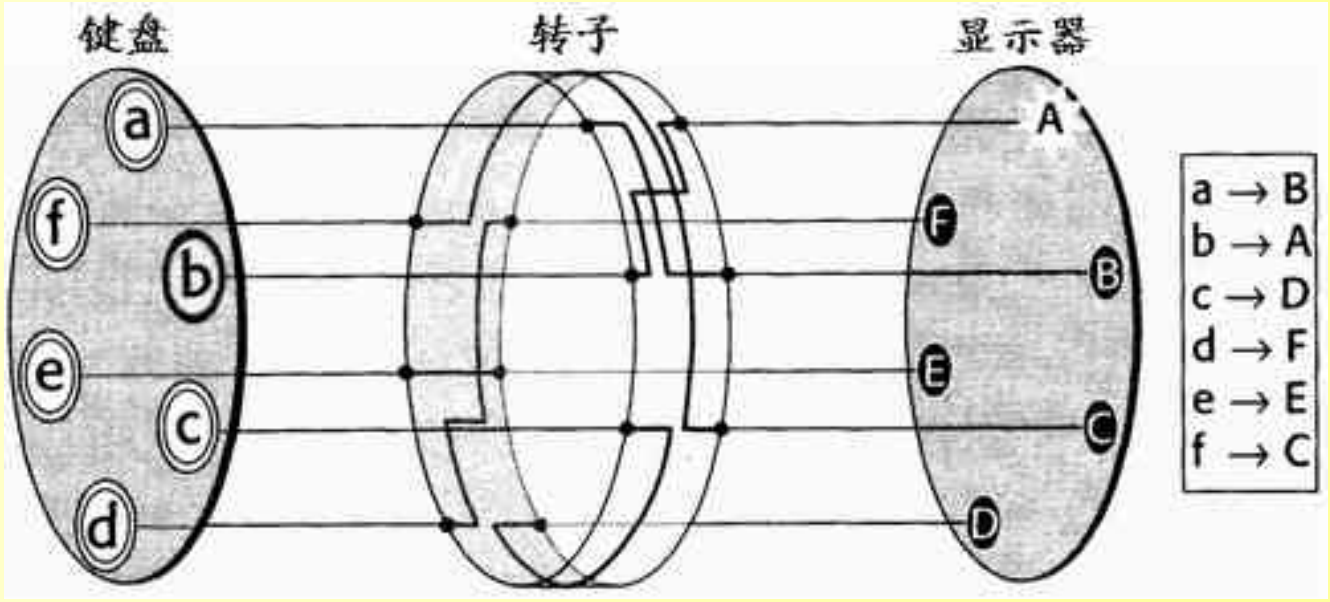
- **1918**年，德国发明家亚瑟·谢尔比乌斯**(Arthur Scherbius)**和他的朋 友理查德·里特**(Richard Ritter)**创办了谢尔比乌斯和里特公司。这是一 家专营把新技术转化为应用方面的企业，很象现在的高新技术公司。
- 谢尔比乌斯负责研究和开发方面，紧追 当时的新潮流。他的一个想 法就是要用二十世纪的电气技术来取代那种过时的铅笔加纸的加密方 法。

Enigma — 诞生

- 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。

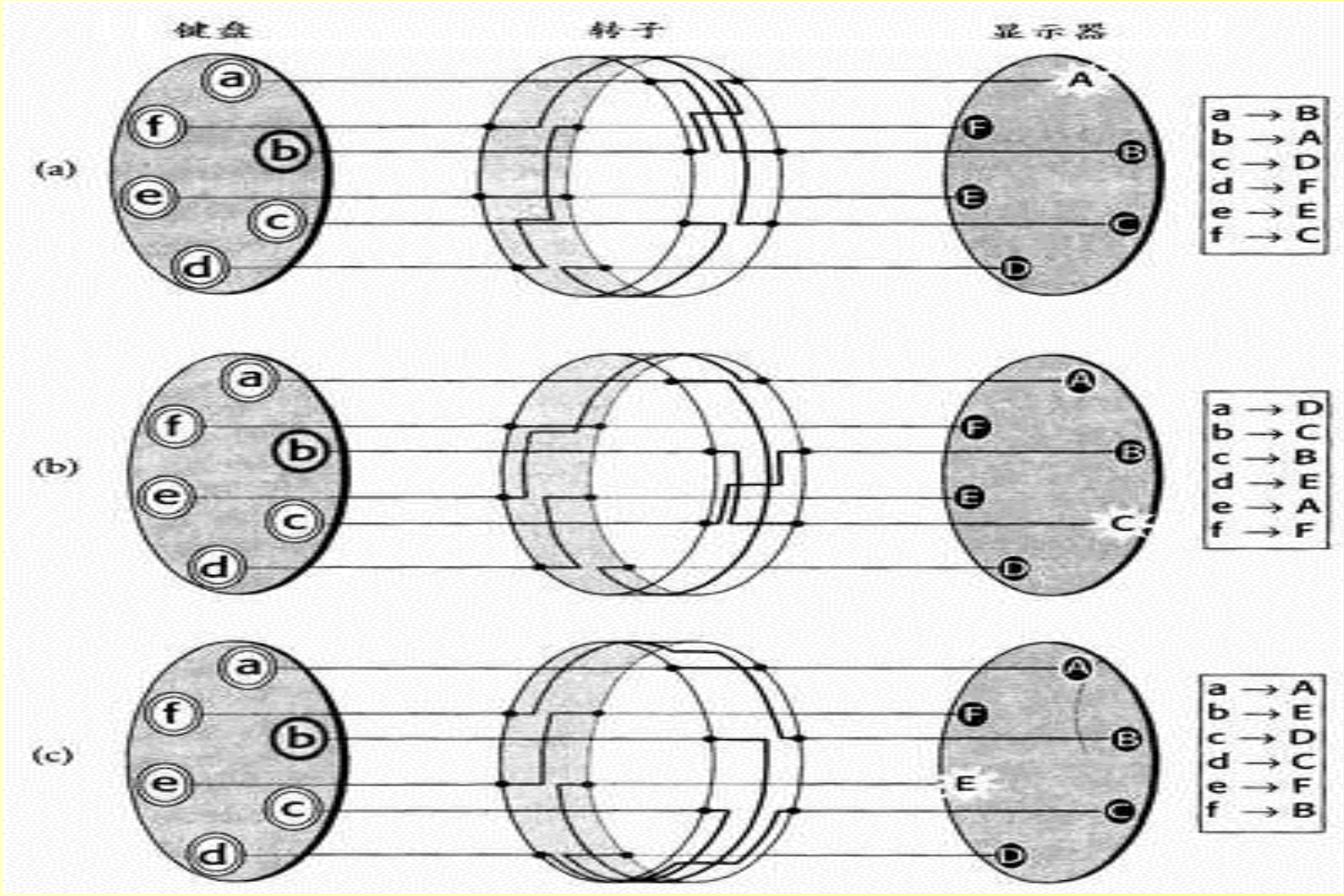
Enigma — 原理

- 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。

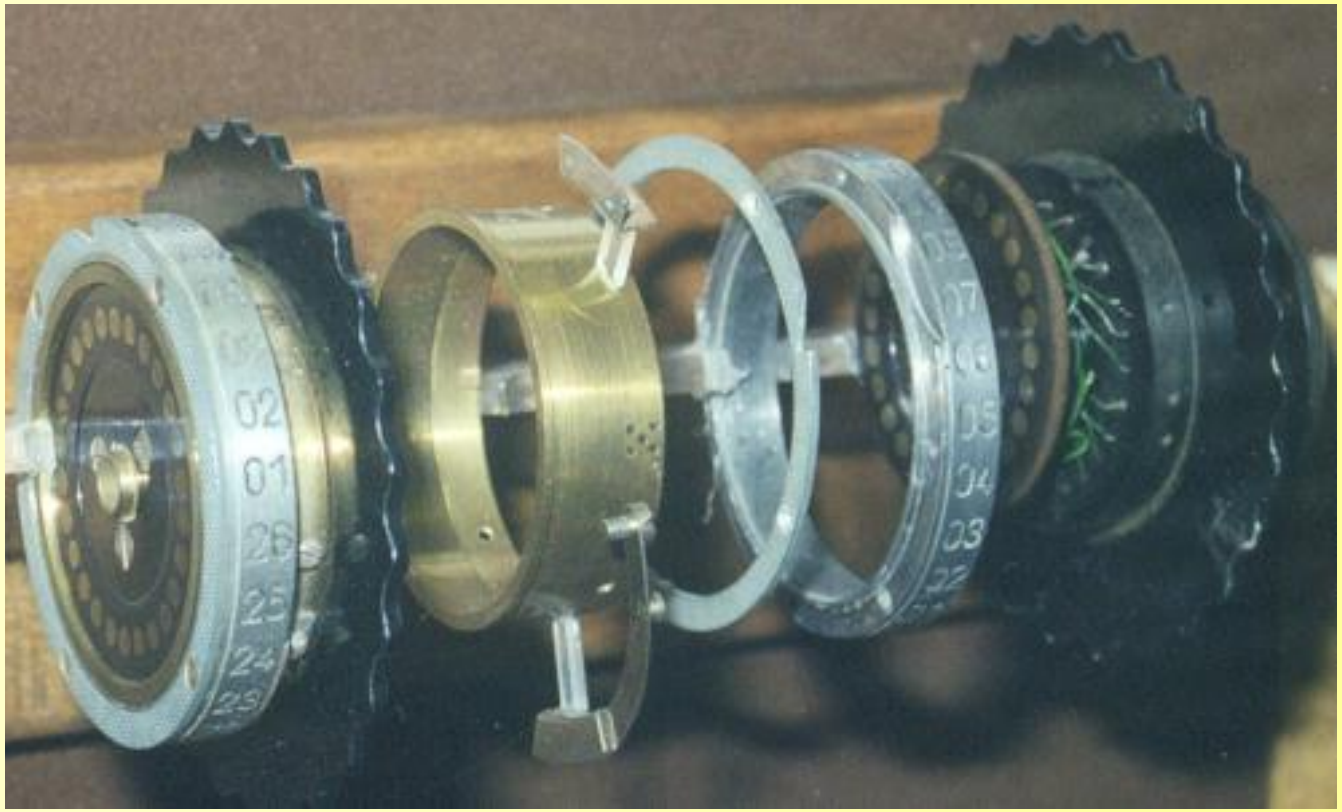


Enigma — 原理

不用的转子或转子在不同的位置给出不同的替代。



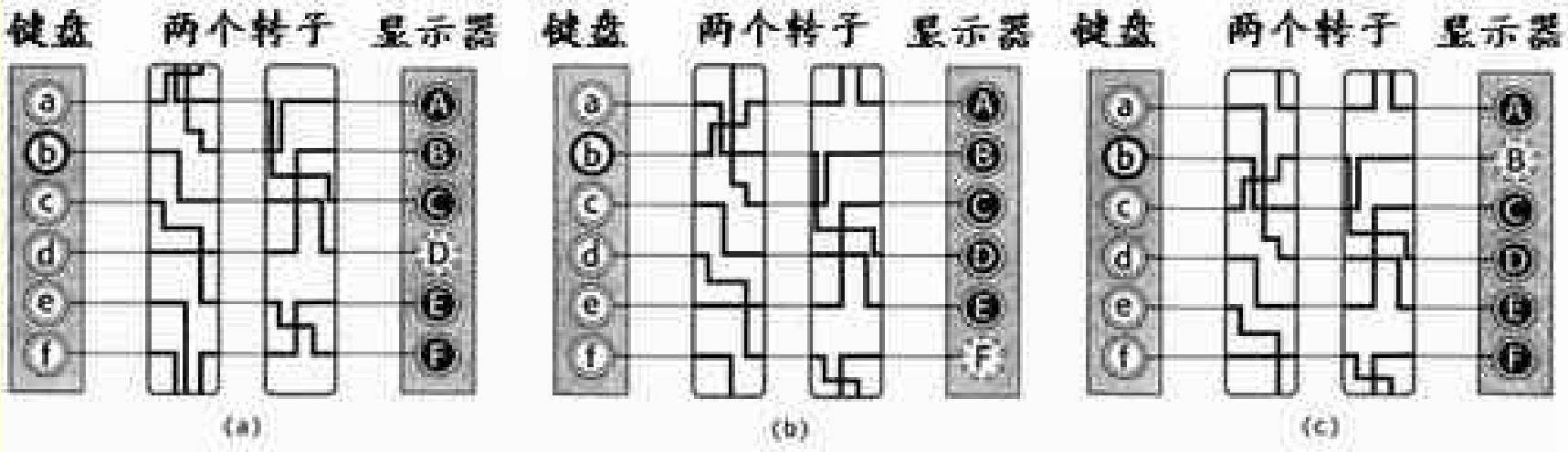
Enigma — 原理



- 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

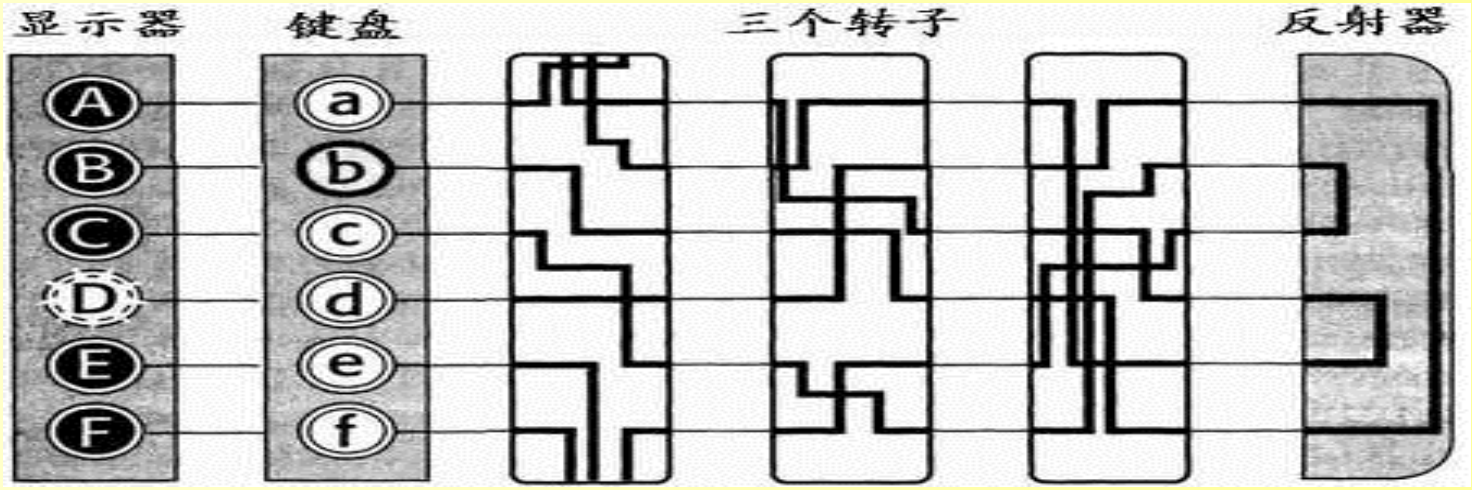
Enigma — 原理

- 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 — 原理

- 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：按下密文字母，明文字母会亮。



Enigma — 加密与解密过程

加密

1. 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钥，这是收发双方必须预先约定好的）
2. 依次键入明文， 并把闪亮的字母依次记下来作为密文字母。
3. 然后就可以把加密后的消息用比如电报的方式发送出去。

解密

1. 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。
2. 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。

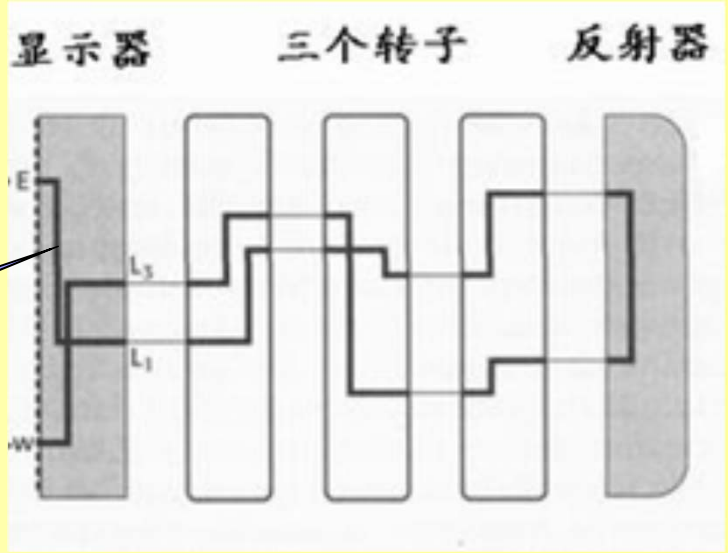
Enigma — 密钥

- 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 — 密钥

- 键盘和第一转子之间增加了一个连接板
  - 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6对（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**

连接板，可以选择若干对互换连接。



Enigma — 密钥 —会话密钥

- 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 — 密钥 —会话密钥

- 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。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；然后将转子的初 始方向按照新密钥来设置。
- 操作方法
  - （1）操作员首先按照上面所说的方法按当日密钥调整好 ENIGMA
  - （2）然后随机地选择三个字母，比如说PGH。
  - （3）把PGH在键盘上 连打两遍，加密为比如说KIVBJE（ 第一次KIV，第二次BJE）。
  - （4）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。
- 发送的报文

$E_{MasterKey}(\text{Session Key} \,||\, \text{Session Key}) \,||\, E_{SessionKey}(\text{Text})$

Enigma — 被攻击的原因（1）— 管理

- 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 — 被攻击的原因（1）— 管理

- 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 — 被攻击的原因（1）— 管理

- 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 — 波兰人的破解（使用方法）

- 在此以前，密码分析人员通常是语言天才，精通对语言方面特征的分析。但是既然**ENIGMA**是一种机械加密装置，波兰总参二局密码处就考虑到，是否一个具有科学头脑的人更适合于它的破译工作呢？
- **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 — 波兰人的破解

- 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

$E_{MasterKey}$(Session Key || Session Key)

Enigma — 波兰人的破解

- 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 — 波兰人的破解

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

- 一天中的MasterKey是固定的，所有的第i个字母都是用同一个字母映射表， i = 1,…6。
- 加密相同的SessionKey得到相同“前6个字母”。
- 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：	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
第四个字母：	F	Q	H	P	L	W	O	G	B	M	V	R	X	U	Y	C	Z	I	T	N	J	E	A	S	D	K

Enigma — 波兰人的破解

- 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

注意：一天中针对前6个字母分别考虑。

Enigma — 波兰人的破解

- 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 — 波兰人的破解

- 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。

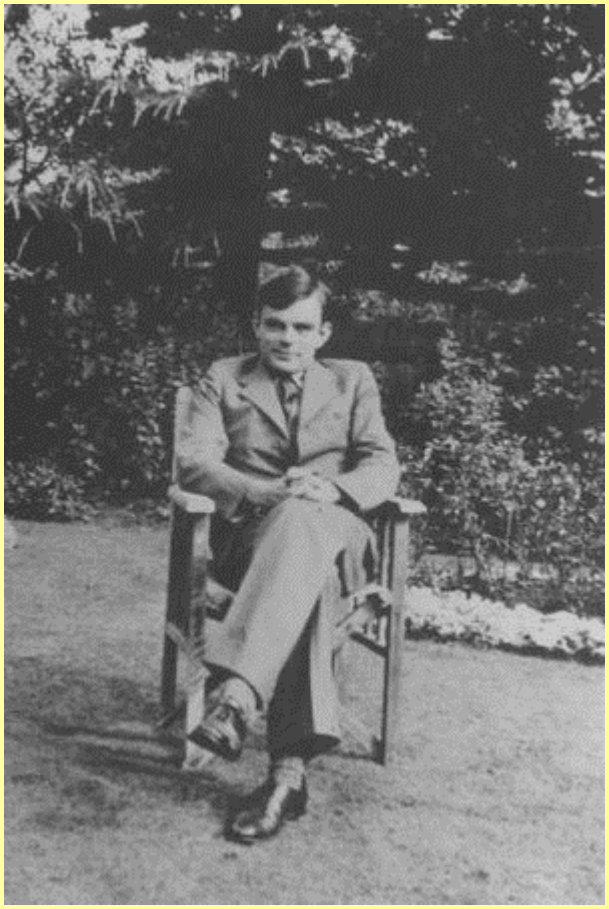
Enigma —图灵的破解

- 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。

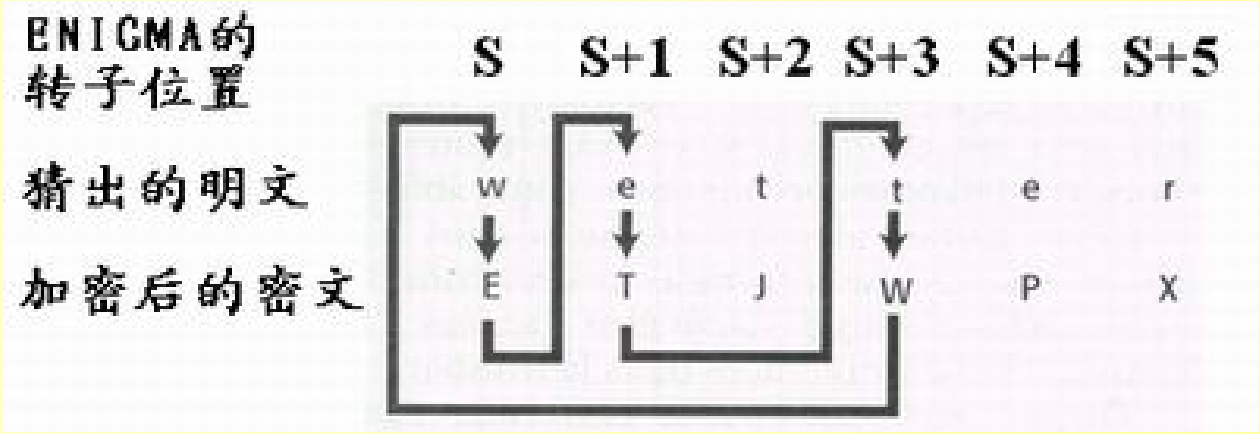
- 这就使得图灵想到了用“候选单词”这一 方法来破译ENIGMA电文。（已知明文攻击）



Enigma —图灵的破解

- 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。

Enigma —图灵的破解



- 图灵并不清楚在密文中出现这个候选单词时的转子状态
- 假设他猜对了这个候选单词
  - 把这个候选单词起始时转子的方向记为**S**，那么在此时**ENIGMA**把**w**加密成了**E**；
  - 然后转子转到下一个方向，就是**S+1**，**ENIGMA**把**e**加密成**T**；
  - 在方向**S+2**上一个不属于这个循环的字母被加密了，这个我们暂且不去管它；接下来在方向**S+3**，**ENIGMA**把**t**加密为**W**。
  - 注意：**S**是不知道的，破解的过程就是猜测**S**的过程。如何快速猜测？

Enigma —图灵的破解

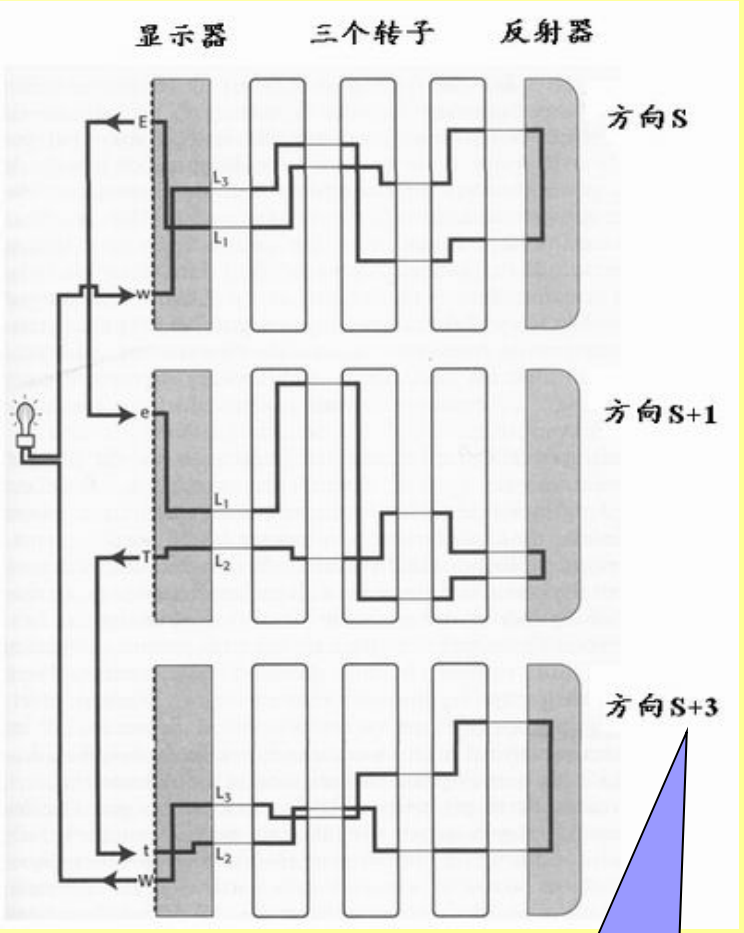
- 如果三个转子按密钥字的方式设置初始方向，则线路通路如图所示形成闭路：
  - 把第一台ENIGMA显示器上的E和第二台ENIGMA显示器上的e连起来，又把第二台上的T和第三台上的t连起来，最后把第三台上的W和第一台上的w连起来（注意ENIGMA上字母没有大小写之分，这里我们只是用大小写来区别密文和明文）。
  - 假设连接板上有关的交换字母的连线是下面这样的（三台ENIGMA机上的都一样）

    $E \longleftrightarrow L_1$

    $T \longleftrightarrow L_2$

    $W \longleftrightarrow L_3$

    当然这里的L_1、L_2和L_3都还是未知的。
- 注意：
  - 如果转轮方向对，按下按键w，灯泡会亮。
  - 如果转轮方向不对，按下按键w，灯泡不会亮。
  - 不断调整初始方向，直到灯泡发亮。
  - 闭路的形成与键盘的连线方式无关。



Enigma —图灵的破解

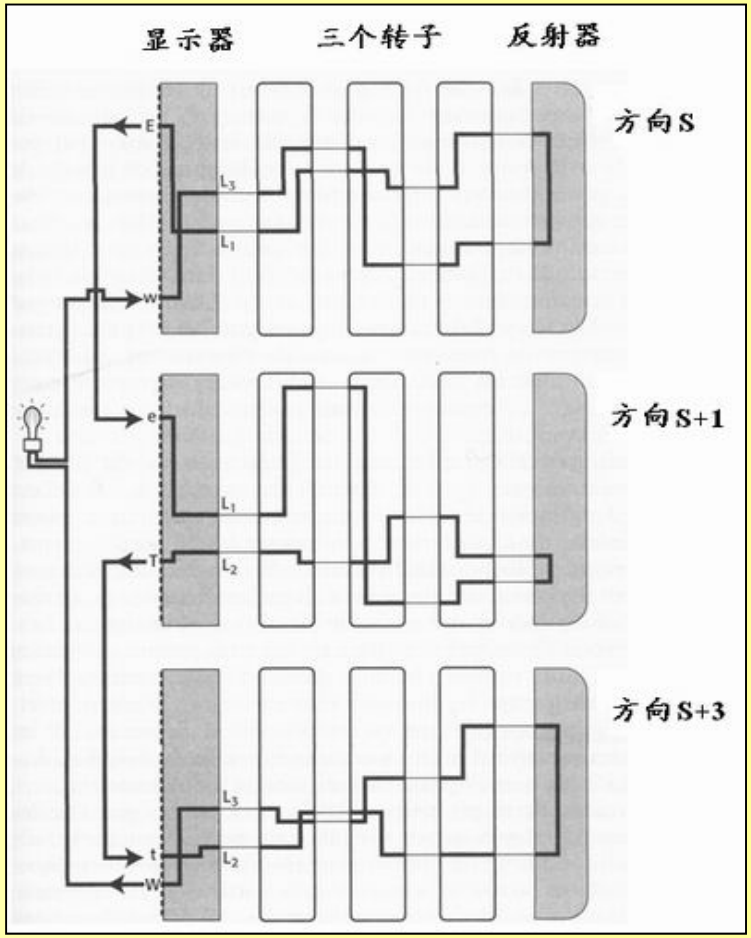
假定：

$E \longleftrightarrow L_1$

$T \longleftrightarrow L_2$

$W \longleftrightarrow L_3$

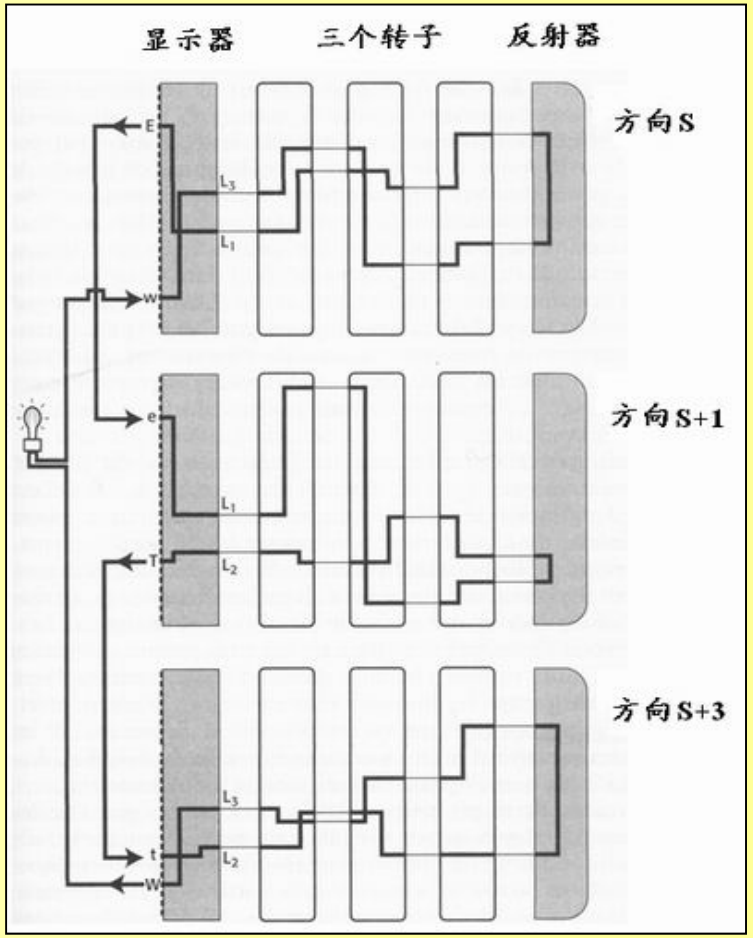
- 现在假设字母w被输入第一台ENIGMA，它先通过连接板变成了L_3，然后通过三个转子经过反射器，再通过三个转子返回连接板；因为我们根据候选单词知道w此时会被加密成E，所以没有经过接线板前它一定是和E对应的L_1；
- L_1经过接线板变成E后，直接成了第二台ENIGMA(转子方向是S+1)的输入。所以根据候选单词知道e此时会被加密成T。从第一台ENIGMA来的e通过连接板变成了L_1，再通过转子和反射器回来变成了连接板上和字母T对应的L_2；通过连接板后变成了T，然后这个T又变成第三台ENIGMA机上的输入t。
- 第三台ENIGMA机的转子方向是S+3，这个传送过来的t会被加密成E，具体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**。这个闭合与初始方向有关，与键盘连线无关。
- $L_1 \rightarrow L_2 \rightarrow L_3$：如果候选单词选对了话，键盘的连线板的后端就会有一个循环 $L_1 \rightarrow L_2 \rightarrow L_3$，当然不是 W→ E→ T （虽然我们看到W→ E→ T 循环）。



Enigma —图灵的破解

- 这个只与转子的初始方向有关。与连线板连接方法无关。
  - $L_1 \longleftrightarrow$E的本质是将T_1的L_1连接到了T_2的L_1：L_1连接任何一个键，因为这个键与T_2的同名键相连，所以也通用连接到了L_1
  - 同理可以证明与L_2是否连接到键T也是无关，与L_3是否连接到键W。
- 分析好了转子初始方向(密钥)，连线板就容易了。
- 赵燕枫，密码传奇——最高级的智力 最隐蔽的搏杀，科学出版社，2008年4月第1版。

$$E \longleftrightarrow L_1$$
$$T \longleftrightarrow L_2$$
$$W \longleftrightarrow L_3$$



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1

- 扩散(diffusion)：
  - 密钥的每一位都能影响密文的许多位，以防止对密钥进行逐段破译（减少复杂性）；
  - 明文的每一位也影响密文的许多位，以便隐蔽明文的统计结构。
  - 通常用线性变换达到扩散的目的。
- 扰乱(confusion)：
  - 使得密文和明文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。
  - 通常用非线性的替代变换达到扰乱的目的。

分组密码的设计准则—安全性2

- 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，
- π_s: $\{0,1\}^l \rightarrow \{0,1\}^l$为置换
- π_p: $\{0,1,\ldots, l\cdot m\} \rightarrow$
  $\{0,1,\ldots, l\cdot m\}$为置换
- P=C= $\{0,1\}^{l\cdot m}$;
- $k=(k^1,k^2,\ldots, k^{Nr+1})$

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

X：

X(1)	X(2)	……	X(m)

$w^0=x$

for r=1 to Nr-1 do

$u^r=w^{r-1} \oplus K^r$

for i=1 to m do

$v^r(i) = \pi_s(u^r(i))$ 替代

$w^r=(v^r_{\pi_p(1)}, v^r_{\pi_p(2)},\ldots, v^r_{\pi_p(lm)},))$ 置换

$u^{Nr}=w^{Nr-1} \oplus K^{Nr}$

for i=1 to m do

$v^{Nr}(i) = \pi_s(u^{Nr}(i))$

$y=v^{Nr} \oplus K^{Nr+1}$

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

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

则称S为幂等的， 即存在S×S到S的一一映射，记为S×S=S 。

- 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。**
- **现代密码的每一轮的变换必须不是幂等的。**

Feistel 密码

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

……

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

……

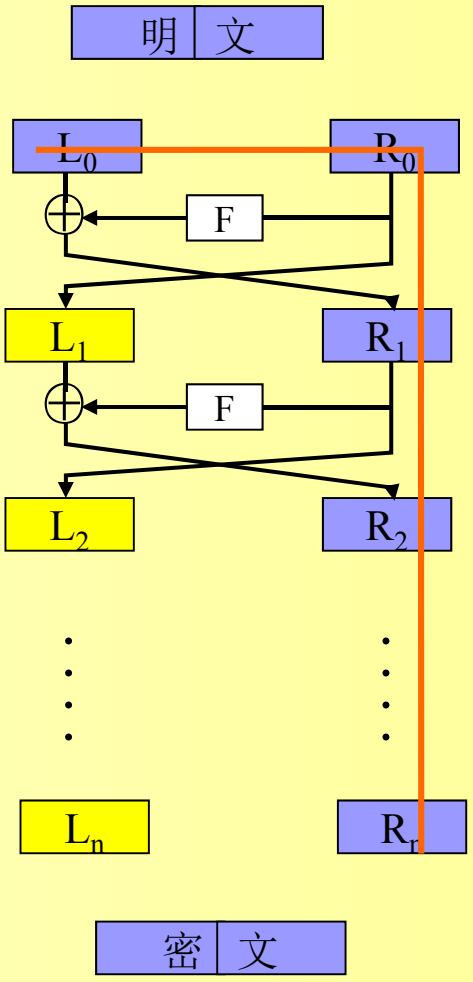
- $R_n=L_{n-1} \oplus F(k_n,R_{n-1})$, $L_n=R_{n-1}$;

轮函数F不必是可逆的：

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

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=$L_n \| R_n=m_{n-1} \| m_n$



- 参考文献：
- 冯登国，吴文玲，分组密码的设计与分析，清华大学出版社，2009年10月第二版。

4. 对称密钥算法DES

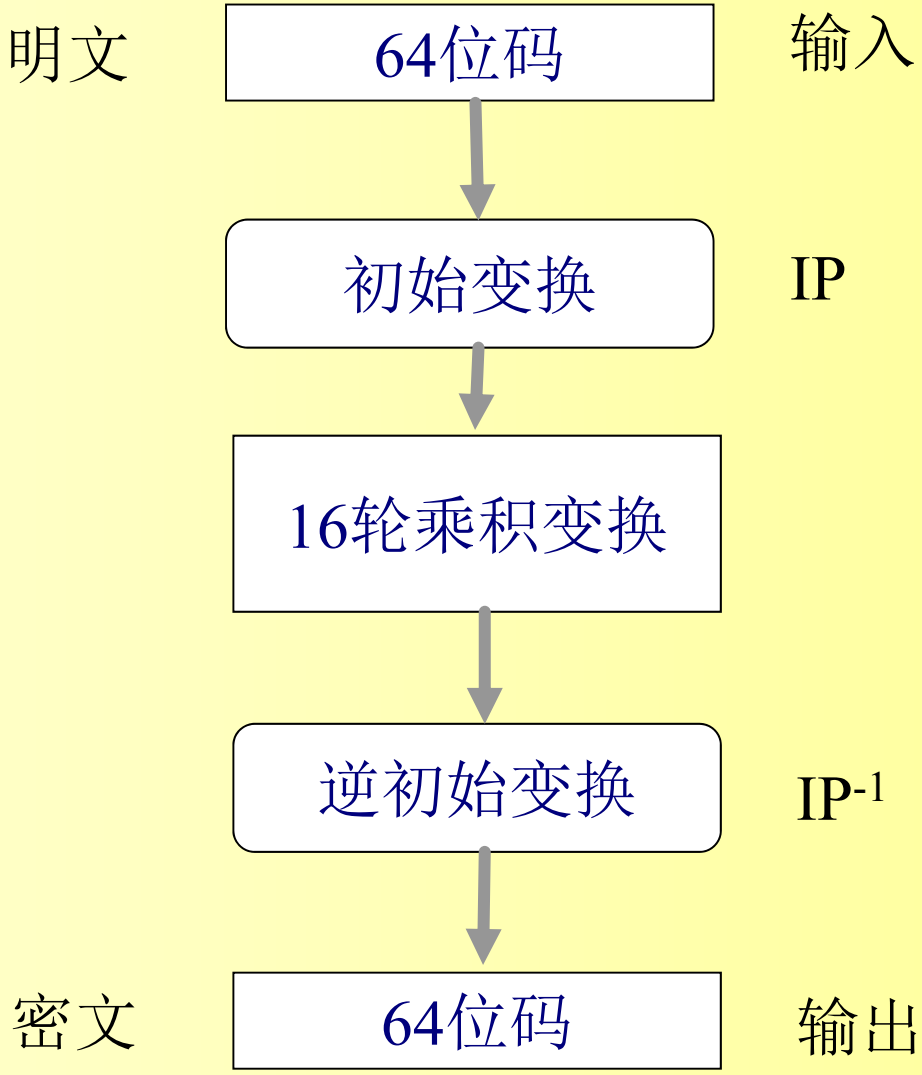
4.1 DES加密算法的背景

- 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

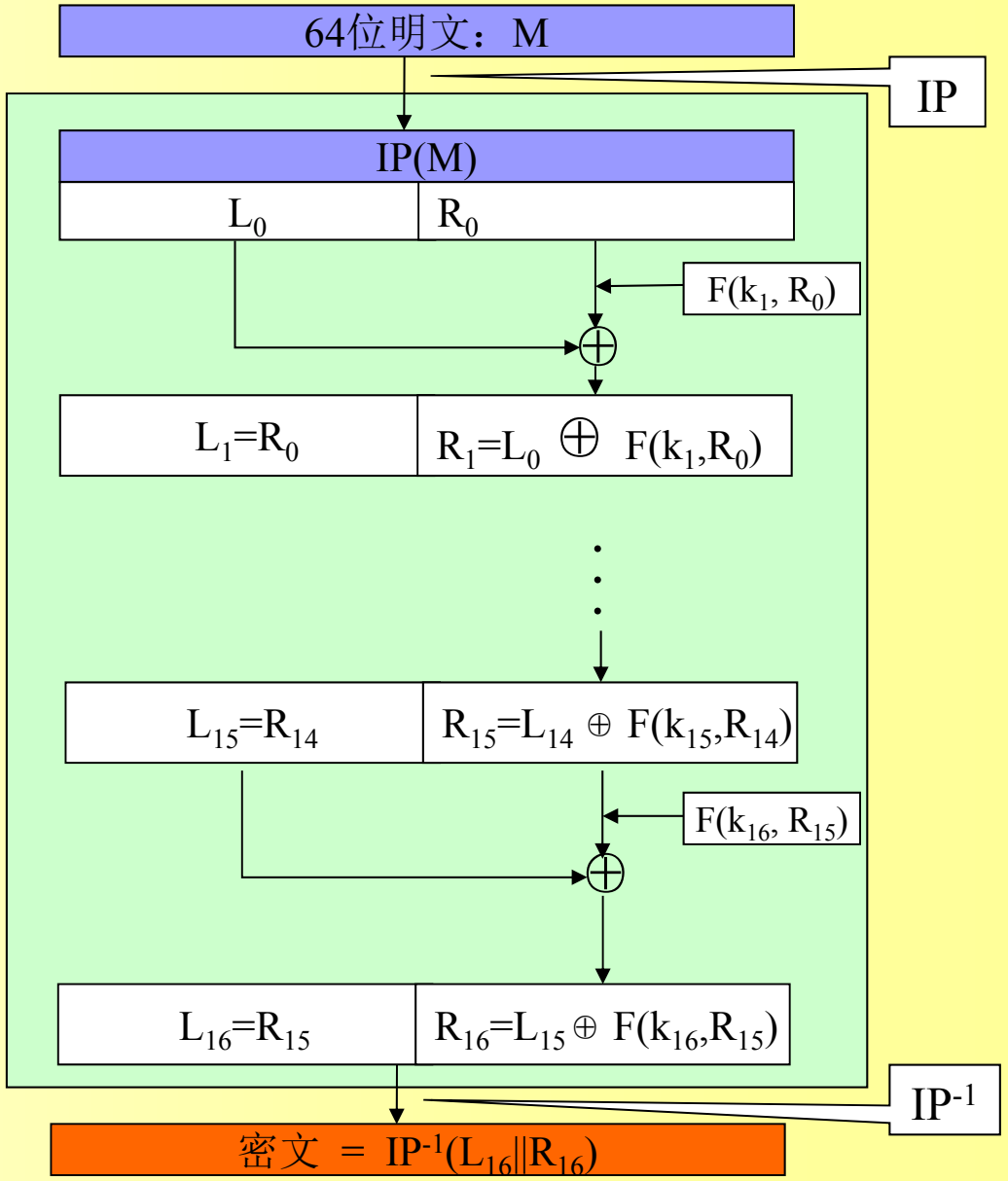
1. 输入64位明文M；
2. 初始变换IP：$IP(M)=L_0 | R_0$
3. 乘积变换：
   - $L_i=R_{i-1}$，$R_i=L_{i-1} \oplus f(R_{i-1}, K_i)$，　$i=1,\ldots16$
4. 初始变换逆IP^{-1}，　IP^{-1}（$L_{16}|R_{16}$）

4.2 DES算法



- R_{-1}, R_0, R_1, …R_{15}, R_{16}。
  - $R_{-1} = L_0$
  - $R_i = R_{i-2} \oplus F(k_i, R_{i-1})$

    $i = 2,\ldots\ldots,16$
  - 密文 = $R_{15} \| R_{16}$



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）

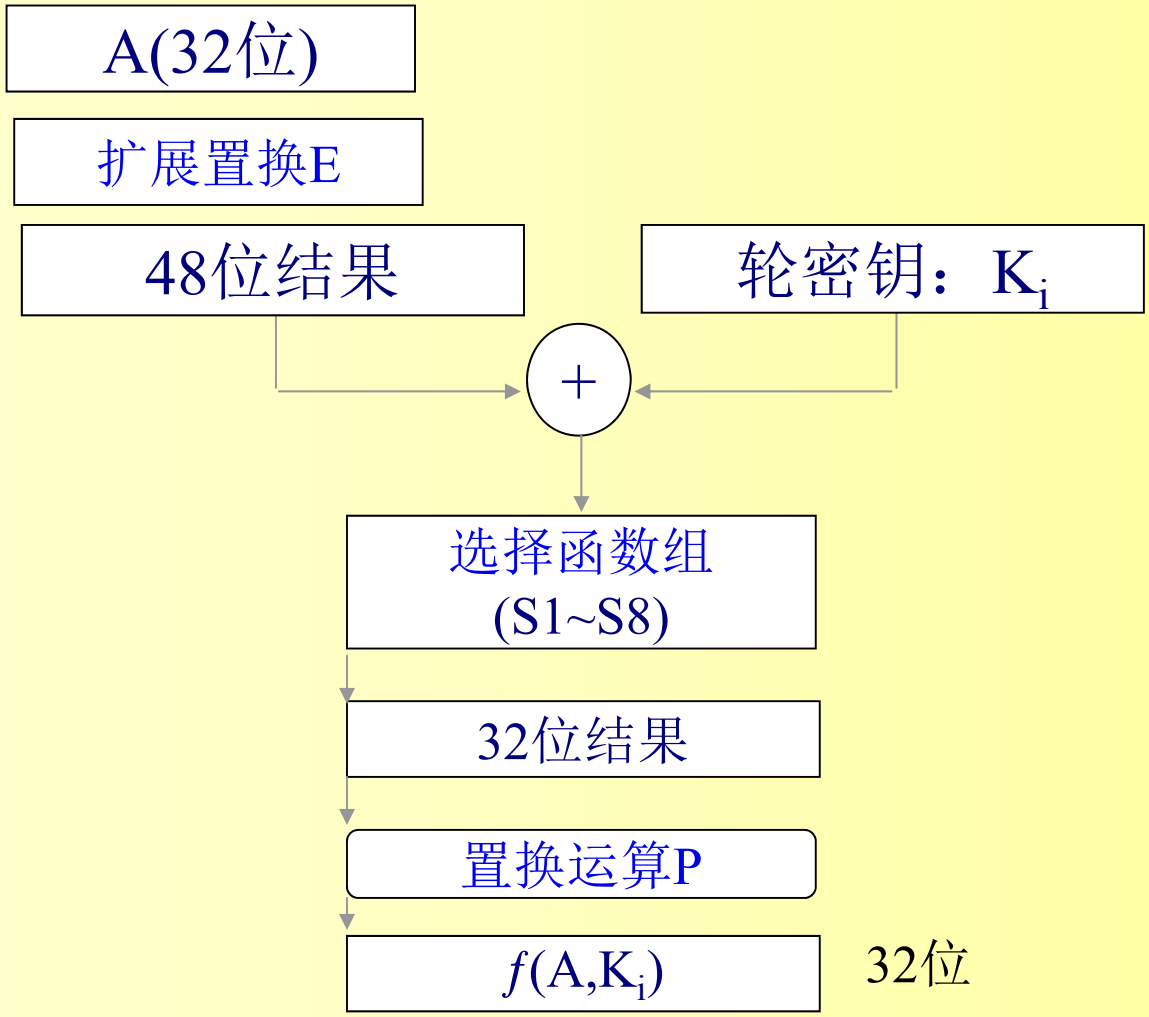
逆初始变换的逆IP^{-1}

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）

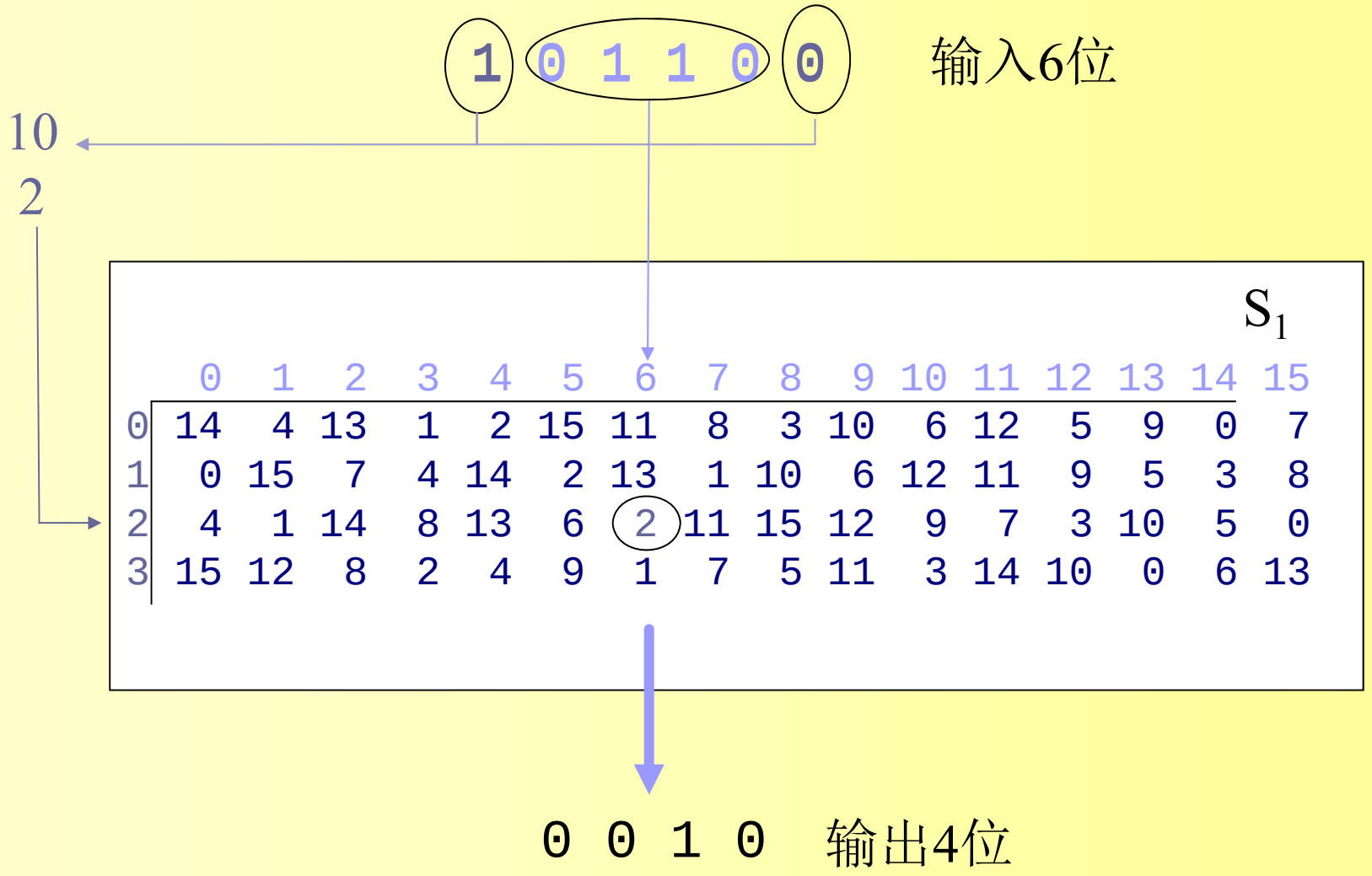
加密函数f(A,K_i)



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

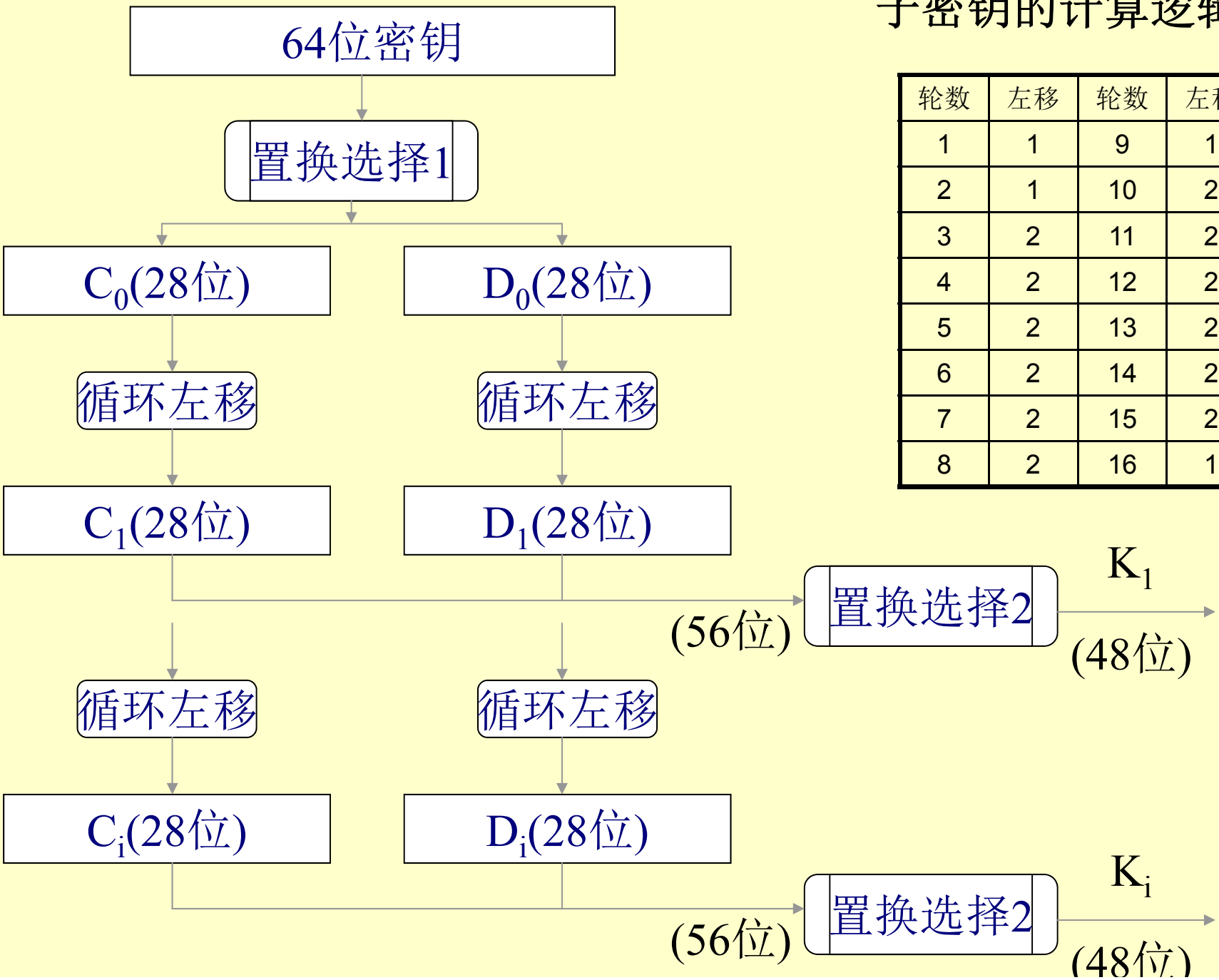
置换P

加密函数的结果 （32位）

第i轮的密码函数

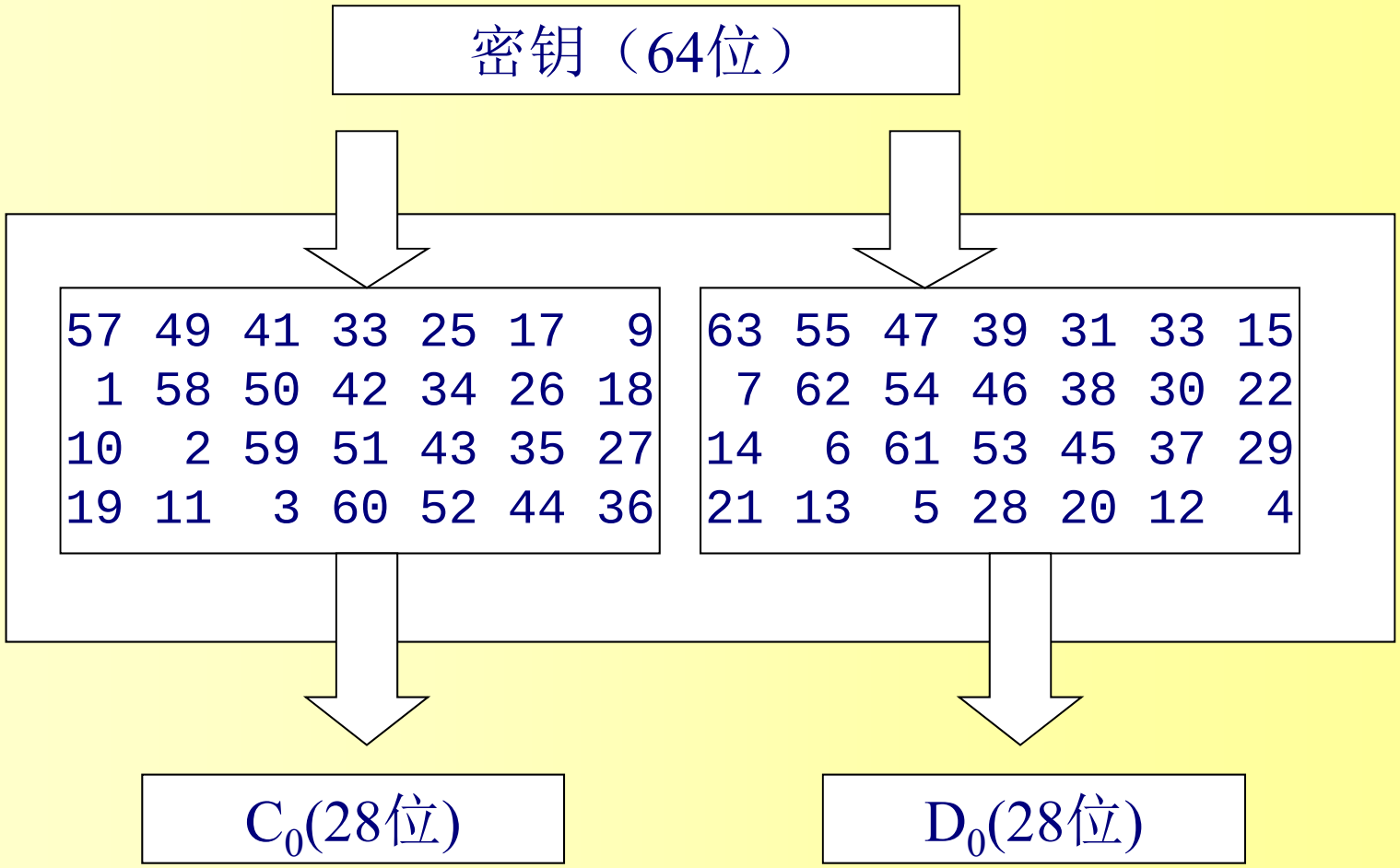
- R_{i-1}
- 扩展：　　$E(R_{i-1})= E_1E_2…E_8$
- 轮密钥：　$K_i=J_1J_2…J_8$
- S盒的输入：$E_1 \oplus J_1$，　$E_2 \oplus J_2$，… $E_8 \oplus J_8$，
  $B_1=E_1 \oplus J_1$，$B_2=E_2 \oplus J_2$，… $B_8=E_8 \oplus J_8$；共48位
- S盒的输出：$S(B_1), S(B_2),……S(B_8)$；　共32位
- 置换，与R_{i-2}作异或：R_i

子密钥的计算逻辑

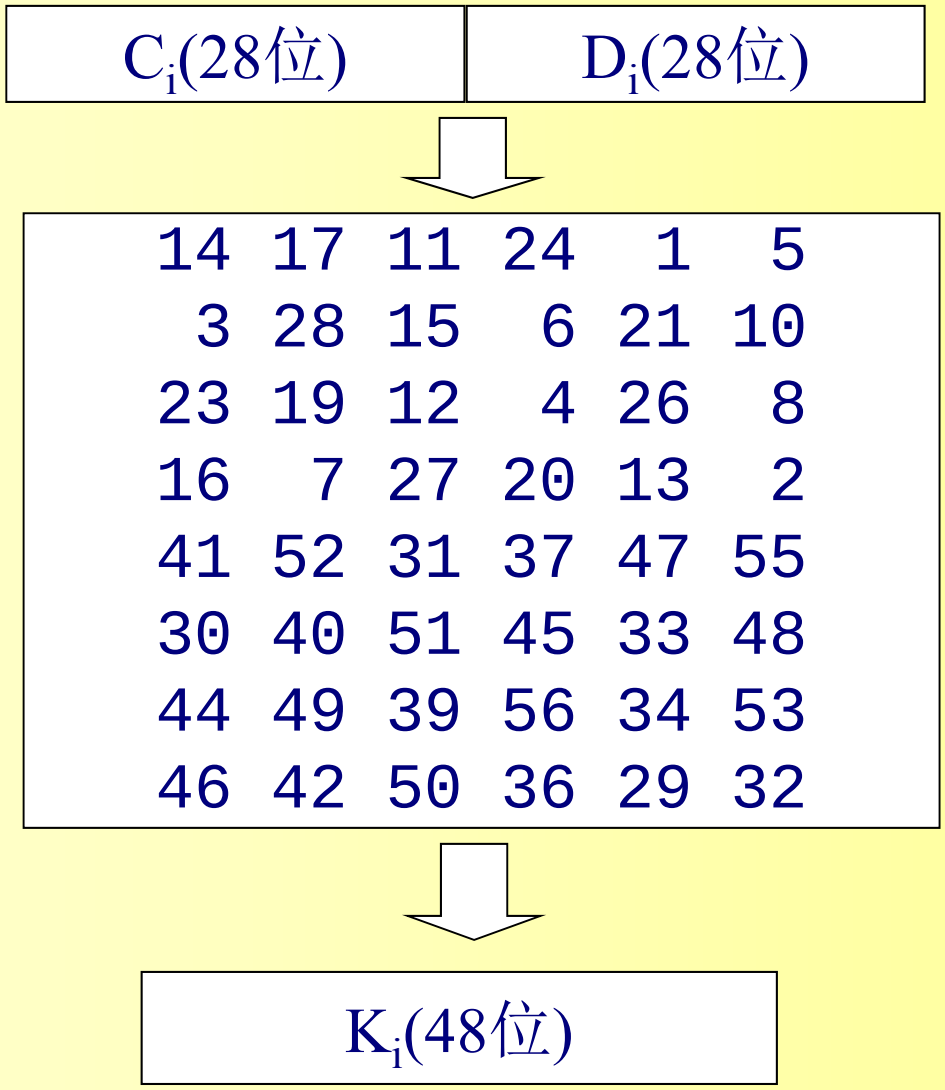


轮数	左移	轮数	左移
1	1	9	1
2	1	10	2
3	2	11	2
4	2	12	2
5	2	13	2
6	2	14	2
7	2	15	2
8	2	16	1

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$E_i=C_n^i F/2^n$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；
  - 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**$D=\{D_i=W_H(p_i \oplus c_i) \mid 0 \le i \le F-1\}$。
  - 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

  $P_i=\{ p_0, p_1, \ldots, p_i \oplus 1, \ldots, p_{n-1}\}$，$p_i \in \{0，1)$，$0 \le i \le n-1$

  得到其相应的密文

  $C_i=\{c^i_0, c^i_1, \ldots\ldots, c^i_{n-1}\}$，$c^i_k \in \{0，1)$，$0 \le k \le n-1$, $0 \le i \le n-1$
- 计算**C与C_i之间的汉明距离$d_i = W(C+C_i)$**，$0 \le i \le n-1$
- 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)。

DES的雪崩效应

- 两个明文只有一个比特不同
- 两个密钥只有一个比特不同
- 两个密文平均有一半的比特不同

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。

- 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
- $K_i=\{ k_0, k_1, \ldots, k_i \oplus 1, \ldots, k_{m-1}\}$，$0 \le i \le m-1$
- 得到其相应的密文
- $C_i=\{c^i_0, c^i_1, \ldots\ldots, c^i_{n-1}\}$，$c^i_k \in \{ 0, 1)$，$0 \le k \le n-1$, $0 \le i \le m-1$
- 计算**C与C_i之间的汉明距离$d_i = W(C+C_i)$**，$0 \le i \le m-1$
- 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-盒的设计准则与争论

- 这个问题涉及美国国家安全局(NSA)
  - 他们修改了IBM的S-盒。
  - 修改后的S－盒满足IBM原先关于S-盒的设计原则。
  - 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。

DES的安全性 — 并行计算

- 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

- MapReduce ？

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。
- 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 0l E0 0l E0 0l 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 vs. Triple-DES

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

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

DES的安全性 — 关于差分分析

- 对DES的分析发现，DES的S-盒具有差分特性，即：其输出的差分(异或)不能遍历0到15，不是等可能地在0 - 15之间取值。
- 根据这点，1990年，以色列密码学家Eli Biham和Adi Shamir对分组密码提出了差分密码分析方法。这是一种选择明文攻击最为有效的方法。
- 虽然对16轮DES没有攻破，但是，如果迭代的轮数降低，则DES可被成功地攻破。
- 例如，8轮DES在一台PC机上只需要2分钟即可被攻破。差分分析正是是利用了DES的S-盒输出的差分不能均匀遍历0-15这一特点。

4.4 差分密码攻击

- 迭代密码在下面的意义上是弱的：
  - 对于$Y_i=F(Y_{i-1}, K_i)$，　$Y^*_i=F(Y^*_{i-1}, K_i)$，　$\Delta Y_{i-1}=Y_{i-1}\oplus Y^*_{i-1}$
  - 若三元组 $(\Delta Y_{i-1}, Y_i, Y^*_i)$ 的一个或多个值是已知的，则确定子密钥K_i是容易的。
- 若密文对已知，并且最后一轮的输入差分能一某种方式得到，则一般来说，确定最后一轮的子密钥或者其中一部分是可行的。

$Y_{16}=F(Y_{15}, K_{16})$，　$Y^*_{16}=F(Y^*_{15}, K_{16})$，　$\Delta Y_{15}=Y_{15}\oplus Y_{15}^*$

最后一轮的输入差分
$\Delta Y_{15}=Y_{15}\oplus Y_{15}^*$

Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9	Y_{10}	Y_{11}	Y_{12}	Y_{13}	Y_{14}	Y_{15}	Y_{16}
Y_1^*	Y_2^*	Y_3^*	Y_4^*	Y_5^*	Y_6^*	Y_7^*	Y_8^*	Y_9^*	Y_{10}^*	Y_{11}^*	Y_{12}^*	Y_{13}^*	Y_{14}^*	Y_{15}^*	Y_{16}^*

DES差分密码分析

- 设S_j是特定的S-盒($1 \le j \le 8$), (B_j, B_j^*)是一对长度为6比特的串。称S_j的输入异或是$B_j \oplus B_j^*$， S_j的输出异或是$S_j(B_j) \oplus S_j(B_j^*)$

对任何$B_j' \in Z_2^6$，记$\Delta B(B_j') = \{(B_j, B_j^*) \mid B_j \oplus B_j^* = B_j'\}$。

易知$|\Delta B(B_j')| = 2^6 = 64$， 且$\Delta B(B_j') = \{(B_j, B_j \oplus B_j') \mid B_j \in Z_2^6\}$。

对中的每一对，我们能计算出的一个输出异或，共计64个异或，

它们分布在16个可能的值上，这些分布的不均匀性是差分密码攻击的基础。

- 对每一个S-盒都有64个可能的输入异或，DES有8个S－盒，共有512个这样的分布。

具有输入异或110100的所有可能的输入

输出异或	可能的输入
0000	
0001	000011,110111,001111, 1110110, 11110,011111,101010,101011
0010	000100,000101,001110,010001,010010,010100,011010,011011 100000,100101,010111,101110,101111,110000,110001,111010
0011	000001,000010,010101,100001,110101,110110
0100	010011,100111
0101	
0110	
0111	000000,001000,001101,010111,011000,011101,100011,101001 101100,110100,111001,111100
1000	001001,001100,011001,101101,111000,111101
1001	
1010	
1011	
1100	
1101	000110,010000,010110,011100,100010,100100,101000,110010
1110	
1111	000111,001010,001011,001011,110011,111110,111111

具有输入异或110100的所有可能的输入

输出异或	可能的输入异或为**110100**的串对（**32**对） 它们的输出异或值不均匀地分布在**8**个值上
0000	
C_j' 0001	{000011,101011, 011110, 101010, 011111, 110111, 001111 , 111011 }= IN(B_j', C_j')
0010	000100,000101,001110,010001,010010,010100,011010,011011 100000,100101,010111,101110,101111,110000,110001,111010
0011	000001,000010,010101,100001,110101,110110
0100	010011,100111
0101	
0110	
0111	000000,001000,001101,010111,011000,011101,100011,101001 101100,110100,111001,111100
1000	001001,001100,011001,101101,111000,111101
1001	
1010	
1011	
1100	
1101	000110,010000,010110,011100,100010,100100,101000,110010
1110	
1111	000111,001010,001011,001011,110011,111110,111111

第i轮的密码函数

- R_{i-1}
- 扩展：$E(R_{i-1})= E_1 \quad E_2 \quad \dots \quad E_8$
- 轮密钥：$K_i= J_1 \quad J_2 \quad \dots \quad J_8$
- S盒的输入：$E_1 \oplus J_1, \quad E_2 \oplus J_2, \quad \dots \quad E_8 \oplus J_8,$

  $B_1, \quad B_2, \quad \dots \quad B_8$ 共48位
- S盒的输出：$S(B_1), \quad S(B_2), \quad \dots\dots \quad S(B_8);$ 共32位
- 置换，与R_{i-2}作异或：R_i

DES差分密码分析的原理

- 在第i轮，S-盒的输入可以写作B=E⊕J，其中

  $E=E(R_{i-1})$，$J = K_i$，

- 输入异或 $B \oplus B^* = (E \oplus J) \oplus (E^* \oplus J) = E \oplus E^*$

  输入异或不依赖于密钥。然而输出异或必然依赖于密钥。

- $IN(B_j', C_j') = \{ B_j \in Z_2^6 \mid S(B_j) \oplus S(B_j \oplus B_j') = C_j' \}$

  能够找到一个B_j'使得输入异或为B_j'，且输出异或为C_j'的B_j的集合。

  $IN(B_j', C_j')$ 是由若干个$E_i \oplus J_i$组成的。

  如果已知输出异或为C_j'的输入对，就可以知道这个对中的E_i，用某一个E_i与 $IN(B_j', C_j')$ 的元素作异或得到的集合必然包含J_i。

- 定义：$test(E_j, E_j^*, C_j') = \{B_j \oplus E_j \mid B_j \in IN(E_j \oplus E_j^*, C_j')\}$
- 密钥比特出现在$test(E_j, E_j^*, C_j')$中。
- 给定一组(E_j, E_j^*, C_j')，C_j'是输出异或，可得$test_1$
- 再给定一组(E_j, E_j^*, C_j')，可得$test_2$
- 密钥比特包含在$test_1 \cap test_2 \cap \ldots\ldots$中。

- **已知一轮S盒的输出异或对应的输入异或；选择一对输入使其异或等于指定的异或值；得到一个包含密钥字的集合test。**
- **差分密码分析是选择密文攻击。**

对r轮迭代密码的差分密码攻击的算法

I. 找出一个**(r-1)**轮特征 $\Omega_{r-1}=\alpha_0, \alpha_1, \ldots, \alpha_{r-1}$，使得它的概率达到几乎最大。

II. 均匀随机地选择明文Y_0, 并计算Y^*_0, 使得的差分为 α_0 ；

找出Y_0 和Y^*_0在实际密钥加密下的密文Y_r和Y^*_r。

(选择明文攻击)

若最后一轮子密钥K_r有2^m个可能值K_r^j $(1\le j\le 2^m)$，设置相应2^m个计数器 Λ_j $(1\le j\le 2^m)$；

用K_r^j解密密文Y_r和Y^*_r, 得到Y_{r-1}和Y^*_{r-1} 。如果Y_{r-1}和Y^*_{r-1}的差分是 α_{r-1} ，则给相应的计数器 Λ_j加1。

I. 重复第II步，直到1个或几个计数器的值明显高于其它计数器的值，输出他们相应的子密钥。

差分密码攻击

- **r 轮特征 Ω**：是一个差分序列：α_0，α_1，…，α_r，
  其中α_0是明文对(Y_0, Y_0^*)的差分，α_i（$1 \le i \le r$）是第i轮输出(Y_i, Y_i^*)的差分。

- **r 轮特征 $\Omega = \alpha_0, \alpha_1, \ldots, \alpha_r$的概率**：在明文$Y_0$和子密钥$k_1, \ldots, k_r$独立均匀随机时，明文对$(Y_0, Y_0^*)$的差分为$\alpha_0$的条件下，第i轮（$1 \le i \le r$）输出$(Y_i, Y_i^*)$的差分为$\alpha_i$的概率。

- **r 轮特征 $\Omega = \alpha_0, \alpha_1, \ldots, \alpha_r$的正确对**：如果明文对$(Y_0, Y_0^*)$的差分为$\alpha_0$，第i轮$1 \le i \le r$）输出$(Y_i, Y_i^*)$的差分为$\alpha_i$，则称明文对$(Y_0, Y_0^*)$为特征r 轮特征 Ω 的正确对，否则成为的错误对。

DES差分密码分析

- 8轮DES需要2^{14}对选择明文；
- 10轮DES需要2^{24}对选择明文；
- 12轮DES需要2^{31}对选择明文；
- 14轮DES需要2^{39}对选择明文；
- 16轮DES需要2^{47}对选择明文；

5. AES

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。

AES

- AES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
  - $T_N = A_N R_C S_{u,v}$
- $AES = T_N \bullet T_{N-1} \bullet \bullet \bullet T_1 \bullet T_0$

有限域的基本概念

1. 整数按照模n加法构成一个交换群：$Z_n = Z / nZ$

   用[i] 记模n剩余类 i (mod n),

   则$Z_n = \{ [0]_n, [1]_n, \ldots\ldots, [n-1]_n \}$。

   $[3]_6 + [4]_6 = [1]_6$, $[3]_6 + [5]_6 = [2]_6$ ……

2. Z_n的非0元素全体按照模n乘法构成环（有些元素没有逆元），

   $[2]_6$ 没有逆元（$\forall x \in Z_6, [2]_6 \cdot [x]_6 \neq [1]_6$）：

   $[2]_6 \cdot [1]_6 = [2]_6$, $[2]_6 \cdot [2]_6 = [4]_6$, $[2]_6 \cdot [3]_6 = [0]_6$,

   $[2]_6 \cdot [4]_6 = [2]_6$, $[2]_6 \cdot [5]_6 = [4]_6$。

有限域的基本概念

1. 小于**n** 且与**n** 互素的正数全体按照模**n**乘法满足
   - 运算封闭：如果a 、b与n互素， 则a · b也与n互素。
   - 结合律： $([a]_n \cdot [b]_n) \cdot [c]_n = [a]_n \cdot ([b]_n \cdot [c]_n)$。
   - 存在单位元， $[1]_n$
   - 存在逆元：如果**a** 与**n**互素，则存在**b**使得**a · b = 1 mod n**。也就是：存在**$[b]_n$**,使得**$[a]_n \cdot [b]_n = [1]$** ，也就是**[a] n**存在逆元。
   - 运算可交换群： $[a]_n \cdot [b]_n = [b]_n \cdot [a]_n$
2. 小于6 且与6互素的正数为{1，5}。

   $\{[1]_6, [5]_6\}$构成群， $[5]_6$ 的逆元是 $[5]_6$ 。

有限域的基本概念

1. 如果p是素数，{1，2,…, p-1}中的数都与p互素。
2. {1，2,…, p-1} 按照模p 加法构成群记为Z_p 。
3. Z_p中非0数全体按照模p乘法构成群； Z_p 全体按照模n 加法和模n乘法构成域，记为 F_p或记为 GF(p)。
4. 特别地： F_2 或GF(2)={0,1}构成域。

有限域的基本概念

- 设p是素数，F_p上的多项式全体构成多项式环$F_p[x]$。

  $a(x) = a_n x^n+a_{n-1}x^{n-1}+\ldots+a_1x^1+a_0$ ， $a_i \in F_p$, i=0,1,…,n

  $b(x) = b_n x^n+b_{n-1}x^{n-1}+\ldots+b_1x^1+b_0$ ， $a_i \in F_p$, i=0,1,…,n

  加法：$a(x)+b(x)=(a_n+b_n)x^n+(a_{n-1}+b_{n-1})x^{n-1}+\ldots+(a_1+b_1)x^1+(a_0+b_0)$

  其中 a_i+b_i的加法运算都是F_p中的加法运算。

  乘法： $a(x) \times b(x) = \sum_{i=0}^{2n} \sum_{j=0}^{i} (a_{i-j} \cdot b_j) x^i$

  其中 $a_i \times b_j$的都是F_p中的乘法运算。

- 在$F_2[x]$中：

  $(x+1)^2=x^2+x+x+1=x^2+(1+1)x+1=x^2+1$, F_2中1+1=0

  x^2+1 有两个因式(x+1)，这与实数域中的多项式不同。

有限域的基本概念

- 如果$f(x)=d_m x^m+d_{m-1}x^{m-1}+\ldots+d_1x^1+d_0$ 没有最高次数大于1的多项式的因式，则称f(x)为**不可约多项式**。
- 如果$a(x) = a_n x^n+a_{n-1}x^{n-1}+\ldots+a_1x^1+a_0$并且n<m, 则

  存在b(x), $a(x)b(x) = 1 \mod f(x)$
- 所有次数小于m的多项式全体按照加法构成交换群，所有次数小于m的非零多项式模f(x)全体按照乘法构成交换群。
- 所有次数小于m的多项式全体构成域$F_p[x] / f(x)$。有p^m个元素$\mathbf{F_{p^m}}$ 。
- **任何一个有限域F，存在素数p和m， 使得F与F_{p^m}同构。**

系数为F_2的多项式域F_{2^8}上的算术

- 设p=2（素数）
  - F_2上的多项式全体$F_2[x]$：

    $a_7x^7+\cdots+a_1x+a_0$，其中$a_i \in F_2, i=0,\ldots,7$

  - 设f(x)是F_2上的8次不可约多项式，$F_2[x]/f(x)$是次数小于8的F_2上的多项式全体构成的域，也记为F_{2^8}或$GF(2^8)$。

  - **可以用8位二进制串$a_7a_6\cdots a_0$来表示$GF(2^8)$上的一个多项式。$GF(2^8)$中元素可以用一个字节表示。**

- 加法
  - $a_7x^7+\cdots+a_1x+a_0$与$b_7x^7+\cdots+b_1x+b_0$在GF(28)上的加法就是按位异或：**$(a_7a_6\cdots a_0)\ \hat{}\ (b_7b_6\cdots b_0)$**

系数为F_2的多项式域F_{2^8}上的算术— 乘法

- 设

  $a(x) = a_7x^7+a_6x^6+\ldots+a_1x^1+a_0$ ， $a_i \in F_2$, i=0,1,…,7

  $b(x) = b_7x^7+b_6x^6+\ldots+b_1x^1+b_0$ ， $b_i \in F_2$, i=0,1,…,7

  $f(x) = x^8 + f_7x^7 +\ldots+ f_1x^1 + f_0$ ， $f_i \in F_2$, i=0,1,…,8

- 计算 a(x)b(x) mod f(x)

- $a(x)\cdot b(x) \bmod f(x)$

  $= a_7x^7\cdot b(x)+a_6x^6\cdot b(x)+\ldots+a_1x^1\cdot b(x)+a_0\cdot b(x)$

- $x\cdot b(x)= b_7x^8+b_6x^7+\ldots+b_1x^2+b_0x$，
- 二进制表示：$b(x)= b_7b_6b_5b_4\, b_3b_2b_1b_0$　　$x\cdot b(x)= b_7b_6b_5b_4\, b_3b_2b_1b_00$
- $x\cdot b(x) \bmod f(x)$
  - **b_7=0**: $x\cdot b(x)= b_6b_5b_4\, b_3b_2b_1b_00$, 也就是左移一位，$b_7$自动丢失。
  - **b_7=1**: $x\cdot b(x) = (b_7b_6b_5b_4\, b_3b_2b_1b_00$) ^ (**1**$f_7f_6f_5f_4\, f_3f_2f_1f_0$)

- 计算$x^i\cdot b(x)$时可以利用$x^{i-1}\cdot b(x)$的计算结果。

系数为F_2的多项式域F_{2^8}上的算术— 乘法

INPUT: Binary polynomials *a(z)* and *b(z)* of degree at most *m-1*.
OUTPUT: *c(z)* = *a(z)* · *b(z)* mod *f (z)*.

I. f = 表示f(z)的二进制数
II. 如果 *a*0 = 1 则 c=b， 否则 c=0;
III. i=1
IV. b= b << 1
V. 如果 b_{m-1}=1 则 b=b+f
VI. 如果 a_i = 1 则 *c*=*c*+*b*.
VII. i=i+1; 如果i < m 转IV
VIII. return c

系数为F_2的多项式域F_{2^8}上的算术— 乘法

```
unsigned char mul_bf(unsigned char a, unsigned char b){
    int  j, b7;
    char c=0;
    char f=0x1b;   //  不可约多项式 f(x) = x^8+x^4+x^3+x^1+1
    if ( a&1 )  c=b;
    for (j=1; j<8; j++){
        b7= b&128;
        b = b<<1;                          (b7 b6 b5 b4 b3 b2 b1 b0 0 )
        if ( b7 )  b = b^f;                ^ (1 f7 f6 f5 f4 f3 f2 f1 f0)
        if ( a & (1<<j) ) c = c ^ b;
    }
    return c;
}
```

系数为F_2的多项式域F_{2^8}上的算术— 乘法

- 如何计算F_{2^8}上的逆元素？

求最大公因子的欧几里德算法

- 输入：整数a>b≥0
- 输出：gcd(a,b)
  - If b=0 return a;
  - return (gcd(b, a mod b))

$$a = b \cdot q_1 + r_1 \qquad r_1 \le b$$
$$b = r_1 \cdot q_2 + r_2 \qquad r_2 \le r_1$$
$$r_1 = r_2 \cdot q_3 + r_3 \qquad r_2 \le r_3$$
$$\cdots\cdots\cdots\cdots \qquad \cdots\cdots$$
$$r_{k-3} = r_{k-2} \cdot q_{k-1} + r_{k-1} \qquad r_{k-1} \le r_{k-2}$$
$$r_{k-2} = r_{k-1} \cdot q_{k-2} + r_k \qquad r_k \le r_{k-1}$$

- 对于a>b >0 成立：

  $a = b \cdot q + r \qquad 0 \le r < b$

  gcd(a, b) = gcd (b, r)

求最大公因子的欧几里德算法

$r_k=0$ $\qquad$ $r_{k-1} \mid r_{k-2}$

$r_{k-2}=r_{k-1} \cdot q_{k-2}+r_k \longrightarrow \gcd(r_{k-2}, r_{k-1})= r_{k-1}$

$r_{k-3}=r_{k-2} \cdot q_{k-1}+r_{k-1} \longrightarrow \gcd(r_{k-3}, r_{k-2})= \gcd(r_{k-2}, r_{k-1})= r_{k-1}$

$\cdots\cdots\cdots\cdots$

$r_1=r_2 \cdot q_3+r_3 \longrightarrow \gcd(r_1, r_2)= \gcd(r_2, r_3)= \cdots= r_{k-1}$

$b=r_1 \cdot q_2+r_2 \longrightarrow \gcd(b,r_1)=\gcd(r_1, r_2)= \cdots= r_{k-1}$

$a= b \cdot q_1 + r_1$ $\qquad$ $\gcd(a,b)=\gcd(b,r_1)=\cdots= r_{k-1}$

$$r_{k-1} \le r_{k-2} \le r_{k-3} \le \cdots\cdots \le r_2 \le r_1 \le b$$

求最大公因子的欧几里德算法

$a= b\cdot q_1 + r_1$	$r_1 = a - b\cdot q_1 = s_1a+t_1b$	$s_1=1,\ t_1=-q_1$
$b=r_1\cdot q_2+r_2$	$r_2=b-r_1\cdot q_2=b-s_1q_2a-t_1q_2b=s_2a+t_2b$	$s_2=-s_1q_2, t_2=1-t_1$
$r_1= r_2\cdot q_3 +r_3$	$r_3=r_1- r_2\cdot q_3 = s_3a+t_3b$	$s_3=s_1-q_3\cdot s_2\ ,\ t_3=t_1-q_3\cdot t_2$
•••••••••		
$r_{k-3}=r_{k-2}\bullet q_{k-1}+r_{k-1}$	$r_{k-1}=r_{k-3}- r_{k-2}\bullet q_{k-1} = s_{k-1}a+t_{k-1}b$	$s_{k-1}=s_{k-3}- q_{k-1}\cdot s_{k-2}\ ,\ t_{k-1}=t_{k-3}b-q_{k-1}\cdot t_{k-2}$
$r_{k-2}= r_{k-1}\bullet q_{k-2} +r_k$	$r_{k-1}= \gcd(a,b)= s\bullet a + t\bullet b$	$r_{k-1}= \gcd(a,b)= s_{k-1}\ a + t_{k-1}b$
$r_k=0$		

$\gcd(a, b)=\gcd(b, r_1)=\cdots=\gcd(r_{k-1}, r_{k-2})$

存在s, t, 满足： $\gcd(a, b)=s\bullet a+t\bullet b$

如果$\gcd(a, b)=1$， 则 $s\cdot a=1 \mod b$。 我们需要知道s和t的值。

求最大公因子的欧几里德算法

		$s_{-1}=1, t_{-1}=0$
		$s_0=0 , t_0=1$
$a= b\cdot q_1 + r_1$	$r_1 = a - b\cdot q_1 = s_1a+t_1b$	$s_1=1, t_1=-q_1$ $\quad s_1=s_{-1}-q_1 s_0, t_1=t_{-1}-q_1t_0$
$b=r_1\cdot q_2+r_2$	$r_2=b-r_1\cdot q_2=b-s_1q_2a-t_1q_2b=s_2a+t_2b$	$s_2=-s_1q_2, t_2=1-t_1$ $\quad s_2=s_0-q_2 s_1, t_2=t_0-q_2t_1$
$r_1= r_2\cdot q_3 +r_3$	$r_3=r_1- r_2\cdot q_3 = s_3a+t_3b$	$s_3=s_1-q_3\cdot s_2 , t_3=t_1-q_3\cdot t_2$
• • • • • • • • •		
$r_{k-3}=r_{k-2}\bullet q_{k-1}+r_{k-1}$	$r_{k-1}=r_{k-3}- r_{k-2}\bullet q_{k-1} = s_{k-1}a+t_{k-1}b$	$s_{k-1}=s_{k-3}- q_{k-1}\cdot s_{k-2} , t_{k-1}=t_{k-3}b-q_{k-1}\cdot t_{k-2}$
$r_{k-2}= r_{k-1}\bullet q_{k-2} +r_k$	$r_{k-1}= \gcd(a,b)= s\bullet a + t\bullet b$	$r_{k-1}= \gcd(a,b)= s_{k-1}\, a + t_{k-1}b$
$r_k=0$		

如果a，b互素，则存在s满足：

$$s \bullet a = 1 \mod b$$

s是a的模n的逆：　$s=a^{-1} \mod n$

二进制系数多项式的扩展欧几里得算法

- a,b, u,v,表示a(x),b(x), u(x),v(x)。deg(f(x))表示多项式的最高次数。
- 假定deg(b(x)) < deg(a(x))。
- 输入： a,b；输出： d = gcd(a,b)，以及 g, h 满足 ag+bh = d.

I. u=a; v=b;
II. g_1=1; g_2=0; h_1=0, h_2=1;
III. 如果 u=0 转VIII;
IV. i = deg(u) – deg(v);
V. 如果 i<0则， u交换v; g_1交换g_2; h_1交换h_2; i =-i;
VI. $u=u+x^iv$;
VII. $g_1= g_1+g_2x^i$; $h1= h_1+h_2x^i$
VIII. 输出：d=v; g=g_2; h=h_2;

二进制系数多项式的扩展欧几里得算法

- a,f 表示a(x),f(x)。 f(x)是8次不可约多项式。
- u,v,表示a(x),b(x), u(x),v(x)。deg(f(x))表示多项式的最高次数。
- 假定deg(b(x)) < deg(a(x))。

- 输入： a；输出：a^{-1}

I. u=a; v=f;
II. g_1=1; g_2=0;
III. 如果 u=0 转VIII;
IV. i = deg(u) – deg(v);
V. 如果 i<0则， u交换v; g_1交换g_2; i =-i;
VI. u=u+x^iv;
VII. g_1= g_1+g_2x^i;
VIII. 输出： g_1;

F_{2^8}中逆元的算法

```
int deg( unsigned short u ) {
    int degree = -1, i=15;
    while ( degree < 0 ){
        if (u >> i) degree = i;
        else i--;
    }
    return degree;
}
```

```
unsigned char inverse(unsigned char a) {
   unsigned short u,v,g1,g2,t;
   unsigned short f=0x11b;
   int  j;

   if ( a == 0 ) return 0;
   u =a; v=f;  g1=1;  g2=0;

   while ( u != 1 ) {
        j = deg(u) - deg(v);
        if ( j < 0 ) {
            t = u;   u = v;  v=t;
            t = g1;  g1=g2;  g2=t;  j = -j;
        }
        u = u ^ (v<<j);
        g1= g1^ (g2<<j);
   }

   a = g1;
   return a;
}
```

(1) 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

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

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

(2) AES — S-盒(ByteSub)

- 字节替代由逆运算和放射变换组成：$y=u\cdot x^{-1}+b$
- 逆元素变换：在有限域 $GF(2^8)=F_2[x]/(m(x))$ 进行

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

其中 $m(x)= x^8+x^4+x^3+x^1+1$

- 仿射变换：$a \rightarrow L_{u,v}(x)=u\cdot x+v$　在环 $F_2[x]/(x^8+1)$ 进行。
  其中$u= x^4+x^3+x^3+x+1$，　$v= x^7+x^6+x+1$
  我们将在后面说明仿射变换可以用F_2上的矩阵运算表示。
- 字节替代最终可以表示成：

$$\mathbf{S}_{u,v}(a(x))=(u(x)\bullet(a^{-1}(x)\bmod(m(x))))+v(x))\ \mathbf{mod}\ (\mathbf{x^8}+\mathbf{1})$$

- **在字节替代的运算过程中，从00到FF共256个元素中的某个元素a(x)，先看成是$F_2[x]/(m(x))$的元素，计算得到$a^{-1}(x)$；转而直接看成是$F_2[x]/(x^8+1)$中的元素，进行仿射变换。在不同的数学结构上进行，保证了AES的非线性。**

AES — S-盒(ByteSub)

- 仿射变换：$a \rightarrow L_{u,v}(a)=u \cdot a+v$　在环 $F_2[x]/(x^8+1)$ 进行。
- 其中$u(x)= x^4+x^3+x^2+x+1$, $v(x)= x^7+x^6+x+1$
- 如果$a(x)=a_7x^7+a_6x^6+\ldots+a_1x^1+a_0$ 则，

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

AES — S-盒(ByteSub)

- $a(x)u(x)= a_7x^7\cdot u(x)+\cdots+a_1x\cdot u(x)+a_0 \bmod (x^8+1)$
- $u(x)= x^4+x^3+x^2+x+1$

a_0	$1\cdot u(x) \bmod (1+x^8)$	$x^4+x^3+x^2+x+1$	00011111
a_1	$x\cdot u(x) \bmod (1+x^8)$	$x^5+x^4+x^3+x^2+x$	00111110
a_2	$x^2\cdot u(x) \bmod (1+x^8)$	$x^6+x^5+x^4+x^3+x^2$	01111100
a_3	$x^3\cdot u(x) \bmod (1+x^8)$	$x^7+x^6+x^5+x^4+x^3$	11111000
a_4	$x^4\cdot u(x) \bmod (1+x^8)$	$x^7+x^6+x^5+x^4+1$	11110001
a_5	$x^5\cdot u(x) \bmod (1+x^8)$	$x^7+x^6+x^5+x+1$	11100011
a_6	$x^6\cdot u(x) \bmod (1+x^8)$	$x^7+x^6+x^2+x+1$	11000111
a_7	$x^7\cdot u(x) \bmod (1+x^8)$	$x^7+x^3+x^2+x+1$	10001111

第1列 第2列

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix}$$

AES — S-盒(ByteSub)

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} = a_0 \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + a_1 \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \ldots + a_7 \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

$a_7\, a_6 \cdots a_0$ 都是为0 或1.

AES — S-盒(ByteSub)

```
int bit[8]={1,2,4,8,16,32,64,128};
unsigned char affine(unsigned char a){
    //unsigned char u[8]={0xf1, 0xe3, 0xc7, 0x8f, 0x1f, 0x3e, 0x7c, 0xf8};
    unsigned char u[8]={0x1f, 0x3e, 0x7c, 0xf8, 0xf1, 0xe3, 0xc7, 0x8f}; // 矩阵
    unsigned char v=0x63;
    unsigned char ret=0;
    int i;

    for (i=7; i>=0; i--){
        if ( a  & bit[i] ){
        ret=ret ^ u[i];
        }
    }
    ret = ret ^ v;
    return ret;
}
```

$S_{u,v}(xy)$： GF(2^8)→GF(2^8) 映射表

- 从工程上看$S_{u,v}$将GF(2^8)中的每个元素映射到GF(2^8)中的唯一的一个元素，因此可以用一个映射表来描述这个映射，如下图所示。
- 编码为16进制xy的一个字节，映射到下表的第x行第y列。

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7C	77	7B	F2	6B	6F	C5	30	1	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	4	C7	23	C3	18	96	5	9A	7	12	80	E2	EB	27	B2	75
4	9	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	0	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	2	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	6	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	8
c	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
d	70	3E	B5	66	48	3	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

- 因为(x^8+1) 在GF(2) 中不是不可约多项式， $x^8+1= (x+1)^8$。所以$GF(2)[x]/(x^8 + 1)$ 是一个环，不是域。
- 但是环$GF(2)[x]/(x^8 + 1)$中有些元素也是有逆元的。
- AES选择的$u(x)= x^4+x^3+x^2+x+1$就有逆元： $u(x)^{-1}= x^6+x^3+x$
- 函数$L_{u,v}(a)=u\cdot a+v$有反函数：$L^{-1}_{u,v}(a)=u^{-1}\cdot a+u^{-1}v$
- $L^{-1}_{u,v}$的映射矩阵如下图所示。

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$L_{u,v}^{-1}$的映射表

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00	52	9	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
10	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
20	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
30	8	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
40	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
50	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
60	90	d8	ab	0	8c	bc	d3	0a	f7	e4	58	5	b8	b3	45	6
70	d0	2c	1e	8f	ca	3f	0f	2	c1	af	bd	3	1	13	8a	6b
80	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
90	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a0	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b0	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c0	1f	dd	a8	33	88	7	c7	31	b1	12	10	59	27	80	ec	5f
d0	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e0	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f0	17	2b	4	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

(3) 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:

  $t(x) = t_3x^3+t_2x^2+t_1x+t_0$

- 列混合变换表示为

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

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

这个多项式环称为AES的字运算。

通过多项式乘法和取模运算d(x)的系数可以表示成下列矩阵方程：

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

,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

$$\mathbf{H_M(T) = MT}$$

(3) AES —列混合变换(MixColumn)

- GF(2^8)上的多项式全体模多项式(x^4+1)构成一个环，不构成域。因为x^4+1不是不可约多项式，$(x^4+1)=(x+1)^4$

  注意：$(x^4+1) \neq x^4+4x^3+6x^2+4x+1$

$$c_3x^3+c_2x^2+c_1x^1+a_0=(a_3x^3+a_2x^2+a_1x^1+a_0)\cdot(t_3x^3+t_2x^2+t_1x^1+t_0) \pmod{x^4+1}$$

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{pmatrix} \begin{pmatrix} t_0 \\ t_1 \\ t_2 \\ t_3 \end{pmatrix} = MT = \begin{pmatrix} a_0t_0 & a_3t_1 & a_2t_2 & a_1t_3 \\ a_1t_0 & a_0t_1 & a_3t_2 & a_2t_3 \\ a_2t_0 & a_1t_1 & a_0t_2 & a_3t_3 \\ a_3t_0 & a_2t_1 & a_1t_2 & a_0t_3 \end{pmatrix}$$

在AES中，$a(x) = 03x^3+01x^2+01x+02 \pmod{x^4+1}$

$a(x)^{-1}= 0Bx^3+0Dx^2+09x+0E \pmod{x^4+1}$

$$M = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \qquad M^{-1} = \begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{pmatrix}$$

虽然GF(2^8)/(x^4+1)是一个环，不是一个域，但是，AES选择的a(x)是有逆元的。

一个环中不是每个元素都有逆元，有些元素是可能有逆元的。

GF(2^8)/(x^4+1)中的元素只要与(x^4+1)互素就有逆元。

(3) AES —列混合变换(MixColumn)

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{pmatrix} \begin{pmatrix} t_0 \\ t_1 \\ t_2 \\ t_3 \end{pmatrix} = MT = \begin{bmatrix} a_0t_0 \\ a_1t_0 \\ a_2t_0 \\ a_3t_0 \end{bmatrix} + \begin{bmatrix} a_3t_1 \\ a_0t_1 \\ a_1t_1 \\ a_2t_1 \end{bmatrix} + \begin{bmatrix} a_2t_2 \\ a_3t_2 \\ a_0t_2 \\ a_1t_2 \end{bmatrix} + \begin{bmatrix} a_1t_3 \\ a_2t_3 \\ a_3t_3 \\ a_0t_3 \end{bmatrix}$$

- 构建a_0, a_1,a_2,$a_3$4张表，分别查t_0, t_1,t_2,$t_3$4等4个值。
- 构建4个向量，只要做异或运算，得到一个列。

(3) AES —列混合变换(MixColumn)

- 参考文献

Darrell Hankerson等，椭圆曲线密码学引论，电子工业出版社，2005年。

该书讨论并给出了很多有限域算术的算法，对于编写AES算法和ECC算法都很有意义。

陈鲁生，沈世镒，现代密码学，科学出版社，2002年7月。

王衍波，薛通，应用密码学，机械工业出版社，2003年8月第1版。

(4) 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；

- AES的计算部件
  - 加密钥A_K
  - 字节代替$S_{u,v}$：参数

    （1） **$x^8+x^4+x^3+x^1+1$；**

    （**2**） **x^4+1，　$u(x)=x^4+x^3+x^3+x+1$，　$v(x)= x^7+x^6+x+1$**
  - 行移位R_C
  - 混合H_M：　$a(x) = 03x^3 + 01x^2 + 01x + 02$，　$x^4 +1$

6. 其他分组密码算法

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。

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。

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

7. 分组密码的操作方式

分组密码的操作方式

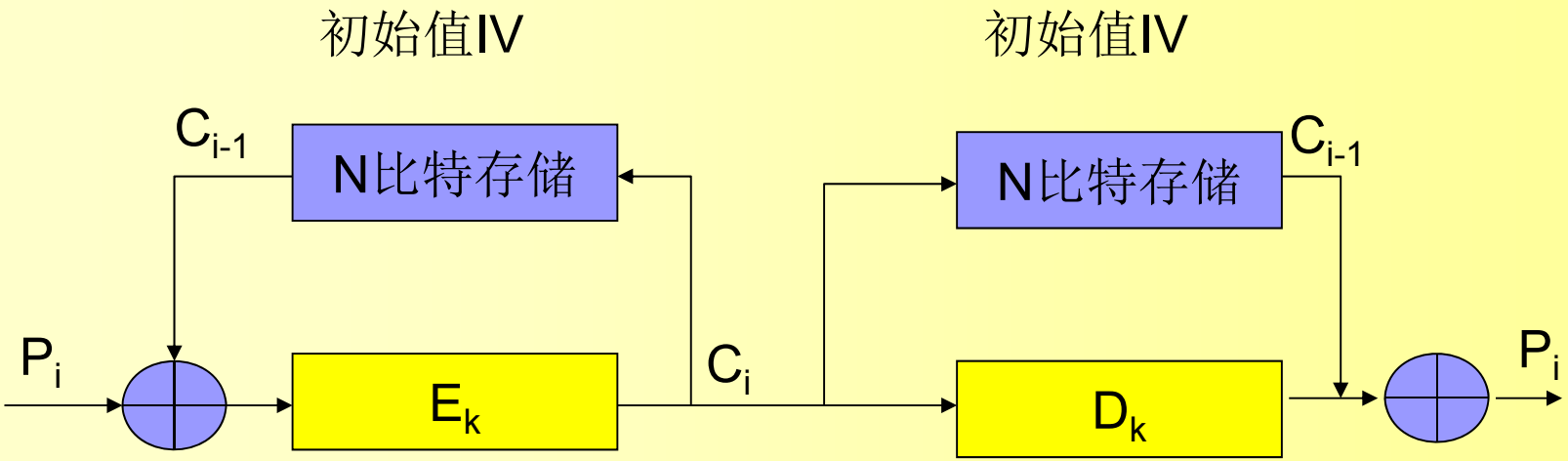
- 电子密码本ECB(Electronic Code Book)
- 密码分组链接方式CBC (Cipher Block Chaining)
- 密码反馈方式CFB (Cipher FeedBack mode)
- 输出反馈方式 OFB (Output FeedBack mode)

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

- 如果银行之间发送的报文的分组关系如上图所示。一个攻击者有可能
  - 记录下原先某次发送的5-12分组中是他的姓名和账号的报文。
  - 随机截取存款的报文，替换其中的5-12分组。

CBC（密码分组链接）



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

$C_1=E_k(P_1 \oplus IV)$

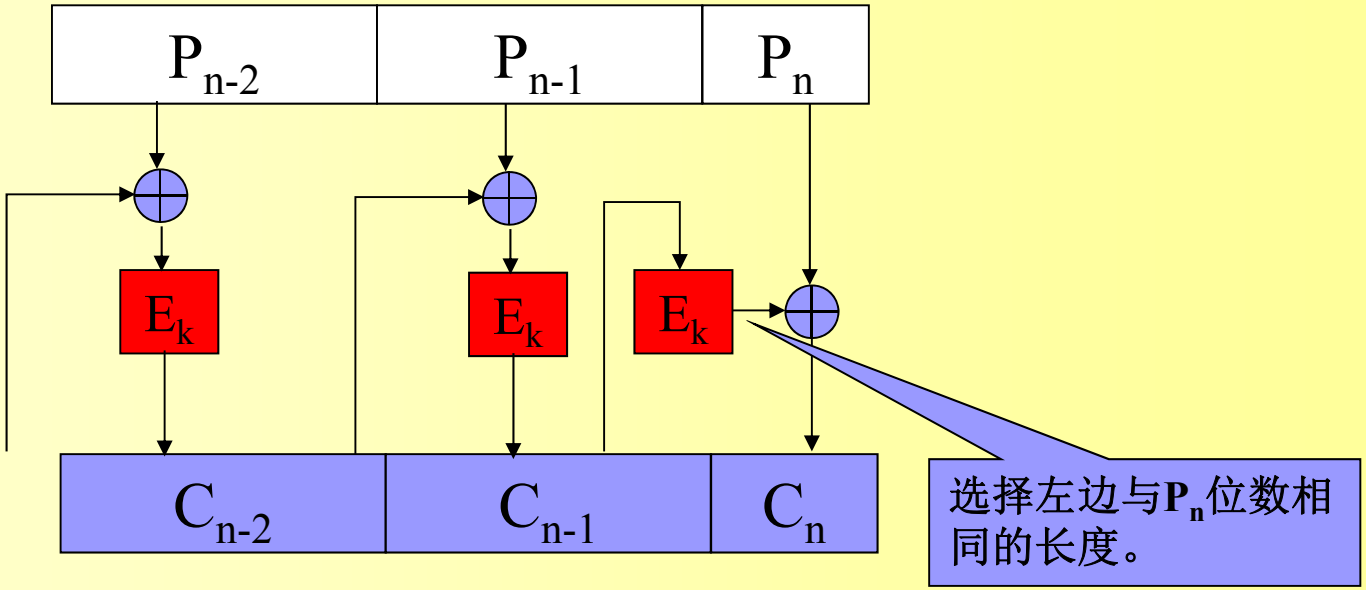
$P_i= C_{i-1} \oplus D_k(C_i)$

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

CBC 填充

- 问题：最后一个分片小于分组的长度。需要密文与明文长度相同。

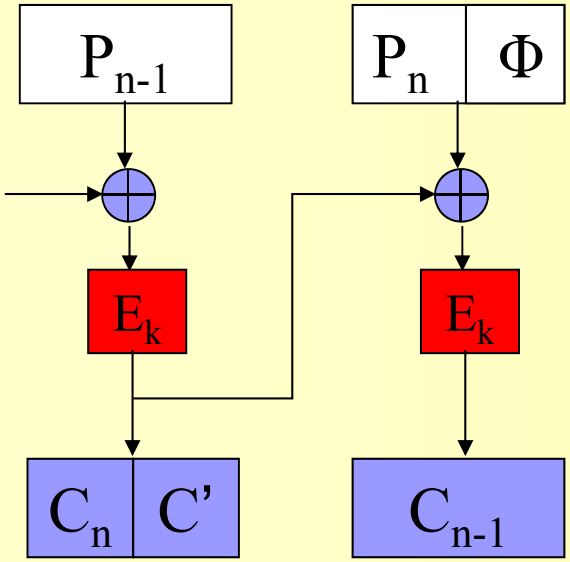
1. 最后一个不完全分组与“密钥”异或。

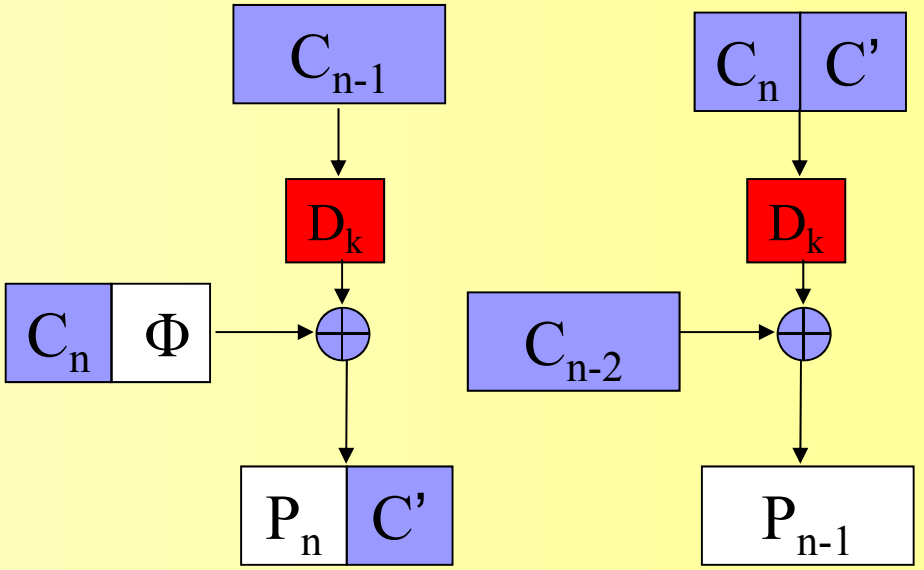


- 攻击者可以修改最后一个密文分组的个别位来修改明文。

CBC 填充

2. 密文挪用。





CBC 错误扩散

- 明文的一位错误将影响该密文分组及其以后的所有密文分组。但是，解密后会反转错误，只有那一位发生错误。
- 密文在传输过程中发生一位的错误，则该密文分组无法解密出正常的明文，也会影响到下一个分组的明文的相同的位。扩散了一个分组。
- CBC无法恢复同步错误，一个密文分组的一位丢失，而后继的所有位都相应地移动，则解密会全部错误。
- **注意：任何使用CBC的加密系统必须保证分组结构完整，有明确的帧或组块等方式。**

序列密码学

- 基本概念
  - 将明文分割成字符串或比特传：

    $M=m_0,m_1,\ldots,m_j,\ldots$

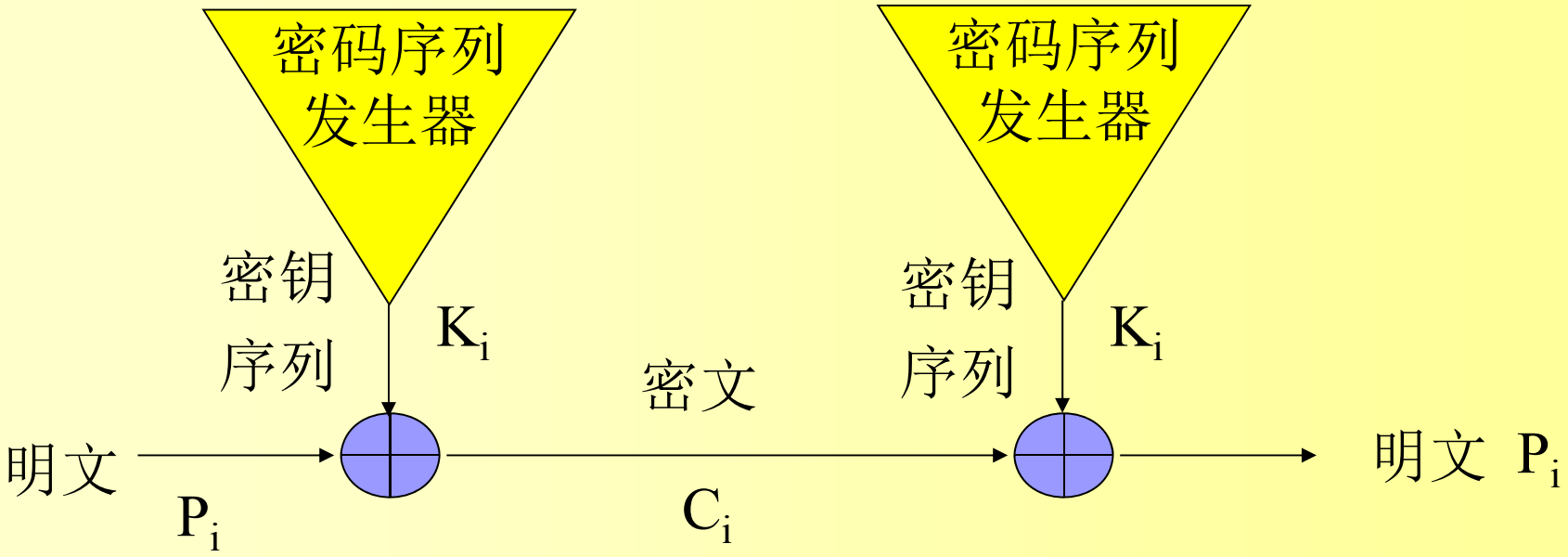
    并逐字符或逐位进行加密：

    $$E_k(M)=E_{k_0}(m_0),E_{k_1}(m_1),\ldots,E_{k_j}(m_j),\ldots$$

  - 其中密钥流是：

    $K=k_0,k_1,\ldots,k_j,\ldots$
  - 周期性序列密码：存在固定的正整数r，使得密钥流每隔r隔字符（r个比特）以后重复。
  - 非周期性序列密码：密钥流不重复。

序列密码算法



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

- 有时候序列密码与分组密码的界限并不十分清楚。
- Vigenère密码既可以看成分组密码

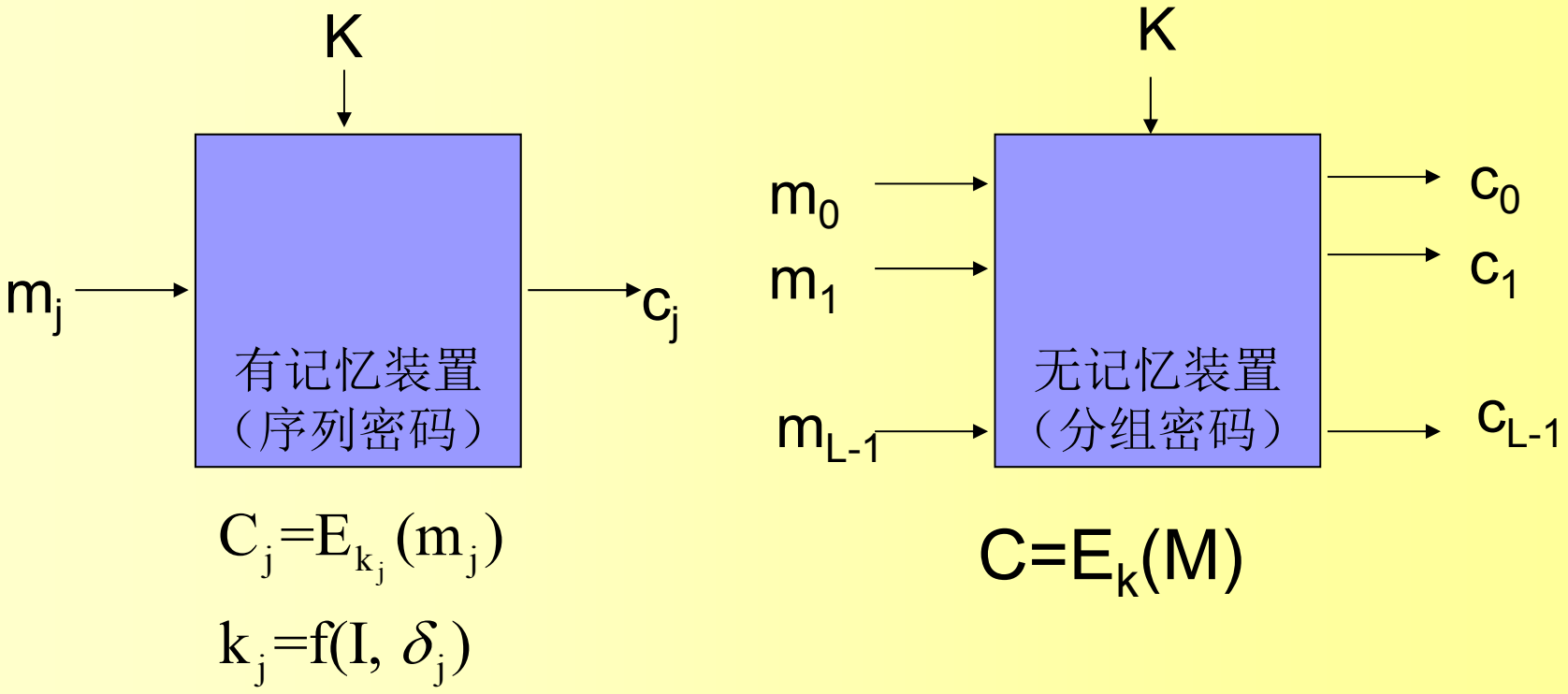
  密钥字是$K=(k_1,k_2,\ldots k_d)$

  也可以看成序列密码：

  周期性的密钥序列是

  $k_1,k_2,\ldots k_d$， …., $k_1,k_2,\ldots k_d$， ….

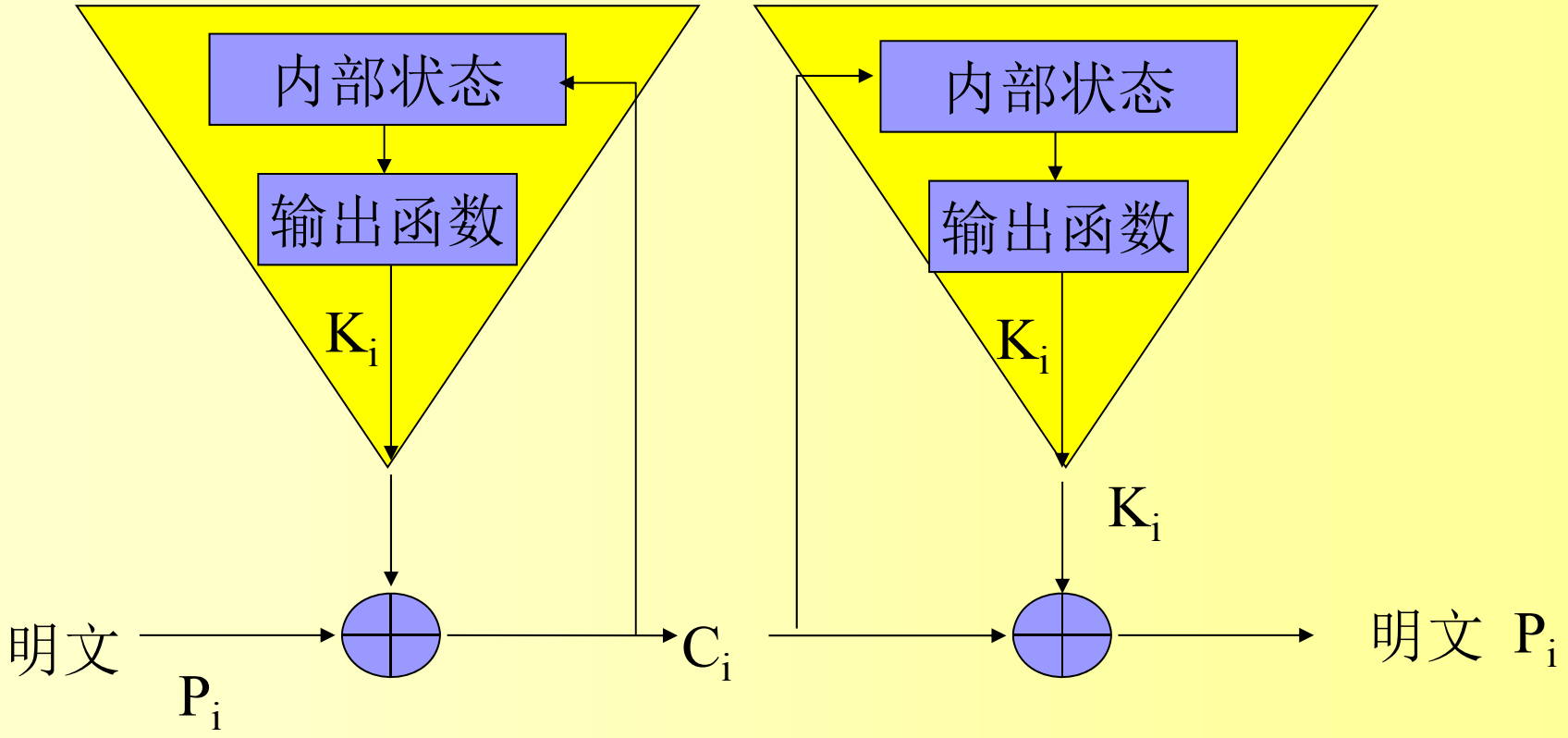
- 序列密码与分组密码的区别
  - 序列密码确定一个内部的记忆装置，用一个依赖与种子密钥I和序列密码在时刻j的内部状态 δ_j 的函数 $f(I, \delta_j)=k_j$



同步与自同步

- 同步序列密码
  - 密钥流的产生完全独立于消息流。密文传输的错误必须重新同步密钥生成器。
- 自同步序列密码
  - 每个密钥字符由前面n个字符的密文推倒出来。传输过程中的一个密文字符的错误会向前传播n个字符。

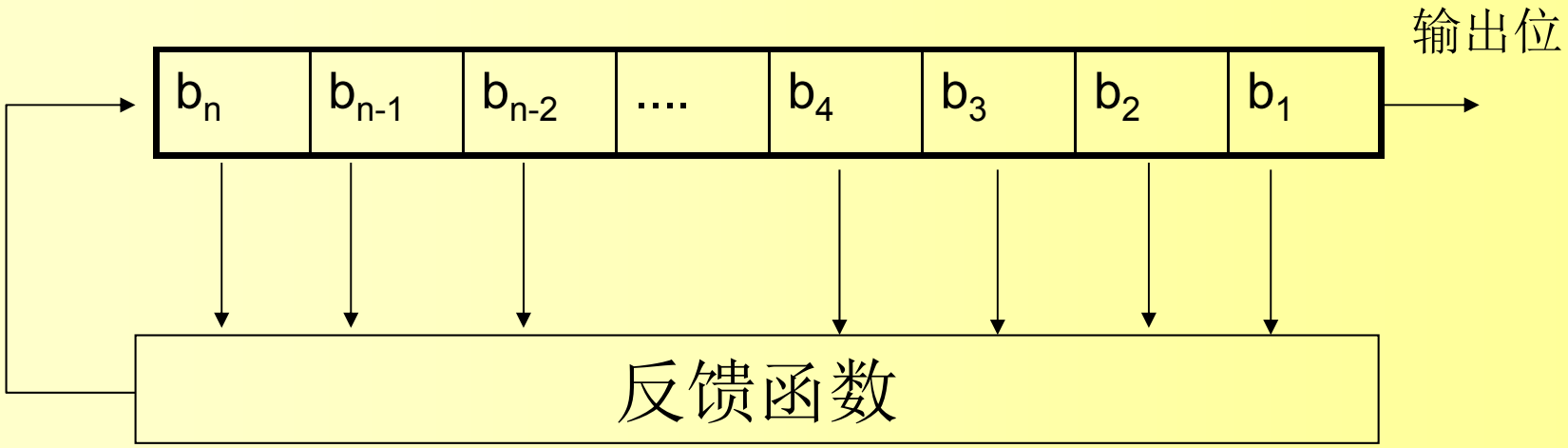
自同步序列密码



密文自动密钥， 会受到回放攻击

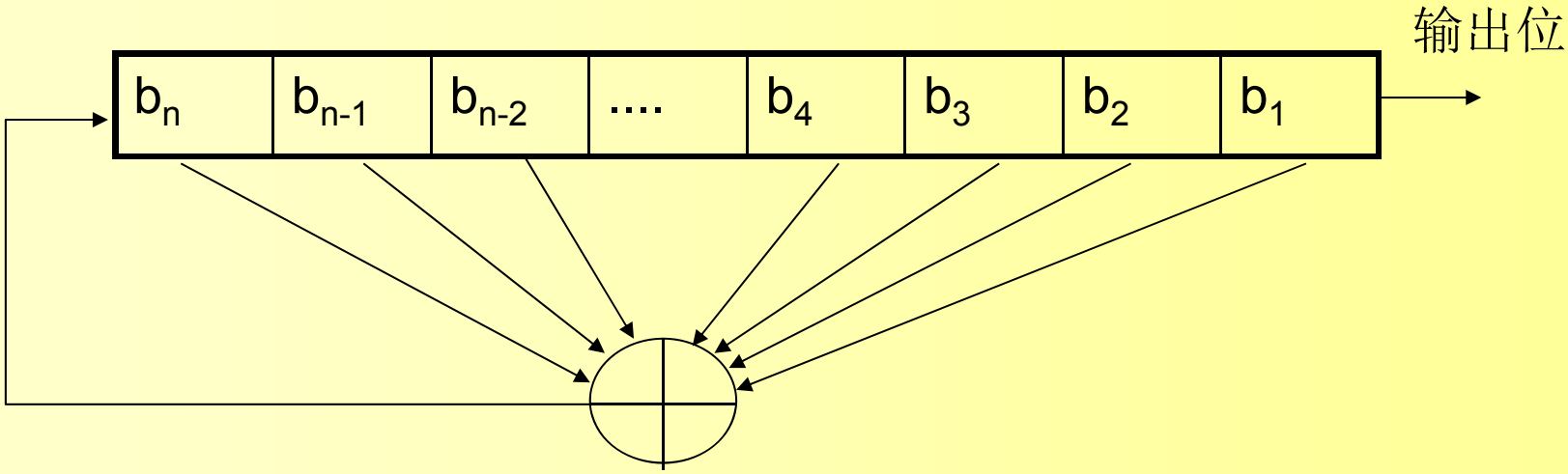
反馈移位寄存器

feedback shift register



线性反馈移位寄存器LFSR

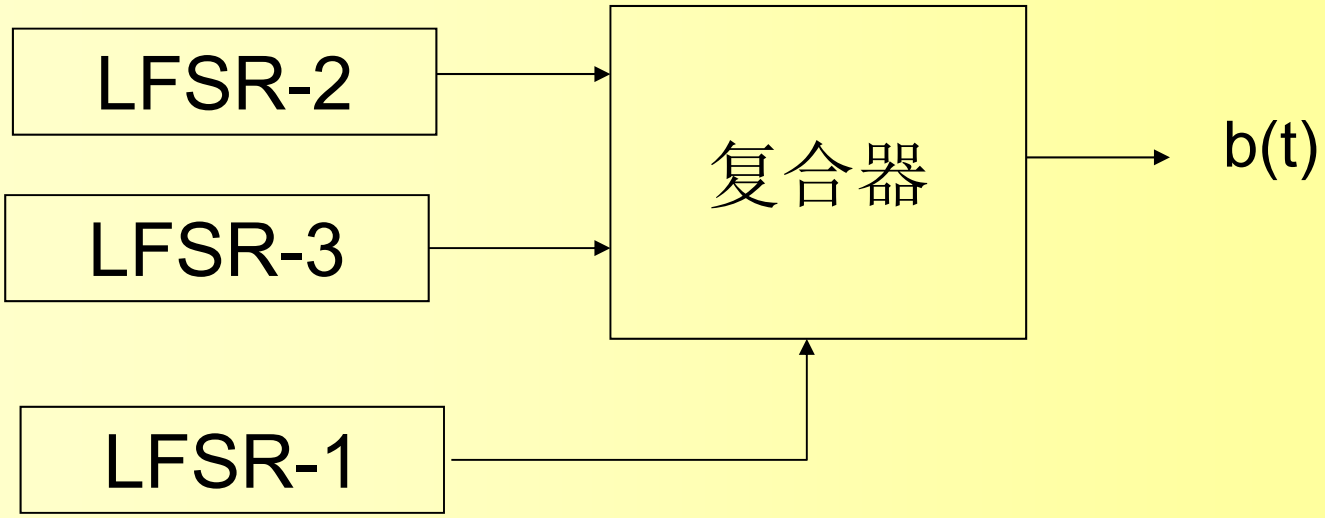
Linear feedback shift register



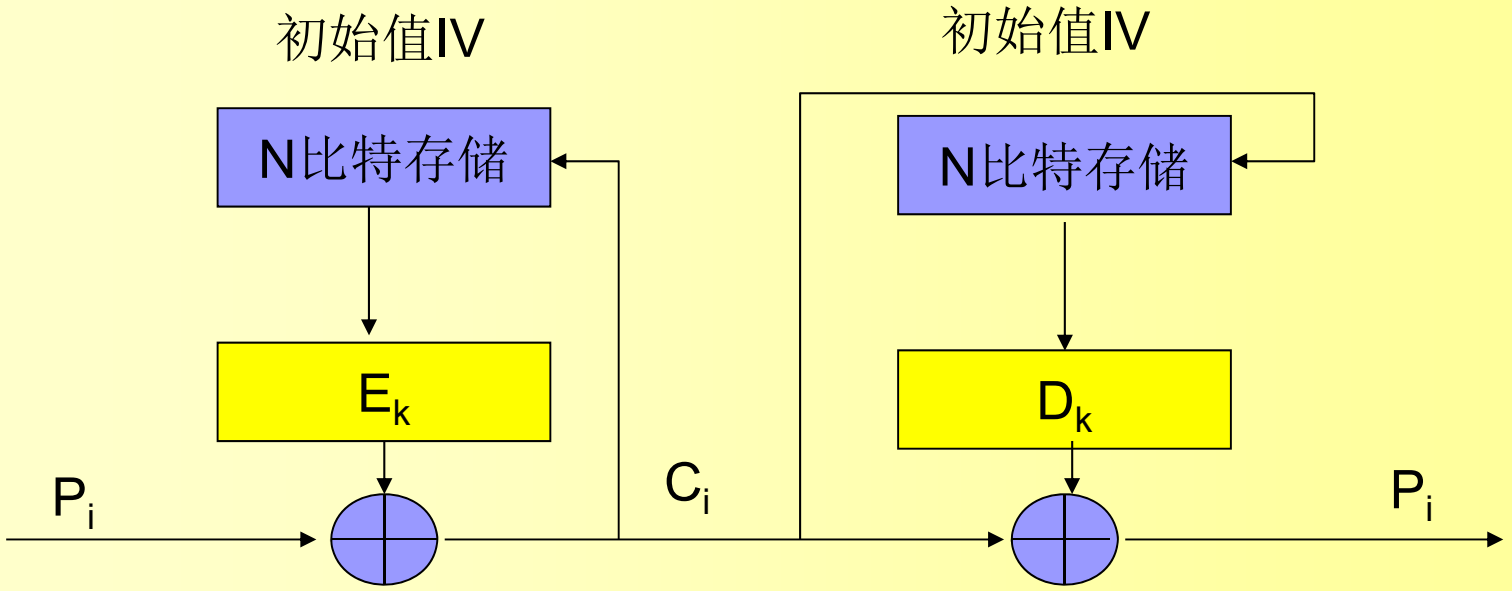
- LFSR的理论包括多项式理论，使得生成的密钥序列周期尽可能地长。

Geffe发生器

$$b = (a_1 \wedge a_2) \oplus ((\neg a_1) \wedge a_3)$$



CFB (CipherFeedBack 密码反馈)



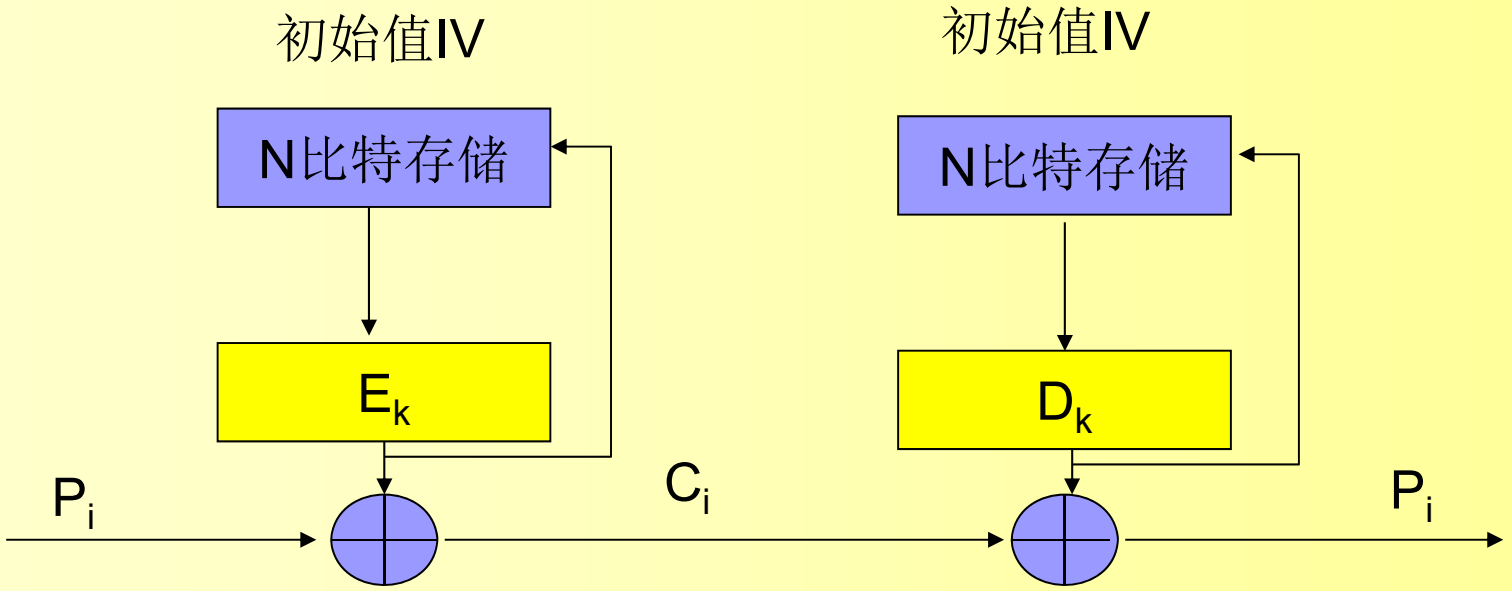
$C_i=P_i \oplus E_k(C_{i-1})$

$C_1=P_1 \oplus E_k(IV)$

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

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，但影响后面的一个分组。

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。由于不能自同步，所以不知道通信起止的攻击者篡改密文就容易被发现。

本章要求

- 掌握　　理解　　了解

- 掌握古典密码的思想和弱点
- 掌握现代密码算法的原理，SPN网络原理，Feistel密码原理。
- 掌握密码操作的原理和方法。
- 掌握AES密码的设计思想。
- 理解DES密码的设计思想。
- 了解密码分析的方法。

实践题目参考

- 编写一个单表替代密码算法的破解程序，给出测试用例与测试结果。
- 编写一个模拟的Enigma加密程序。模拟Enigma的破解。
- 研究DES算法，用C语言快速实现快速，详细阐述实现的程序，阐述快速的机制，给出测试用例与测试结果。解密标准DES加密算法加密的密文。
- 研究与实现重合指数法与重合互指数法分析多表替代算法的破解方法。

- 研究AES的算法思想，编写快速的AES算法，研究在多核处理器上的并行实现，详细阐述实现程序方法。设计测试方案、测试用例，给出测试结果。解密标准AES算法加密的密文。
- 研究AES的原理，详细阐述修改模函数、系数函数等算法的参数，实现新的加解密算法。设计测试用例，给出测试结果。解密标准AES加密算法加密的密文。
- 利用GPU编写AES的并行快速实现程序。详细阐述GPU的并行原理，AES的并行设计。设计测试方案、测试用例，给出测试结果。
- 在Windows内核中实现一个文件系统过滤驱动，实现文件透明加解密，提供用户管理接口。设计测试用例，给出测试结果。
- 在Linux内核中实现一个文件系统过滤驱动，实现文件透明加解密，提供用户管理接口。

学习报告的格式

- 问题的描述
- 一种或多种解决问题的原理
- 程序设计的数据结构说明，函数结构说明，函数实现算法说明。
- 测试方案，测试用例，测试结果。
- 参考文献