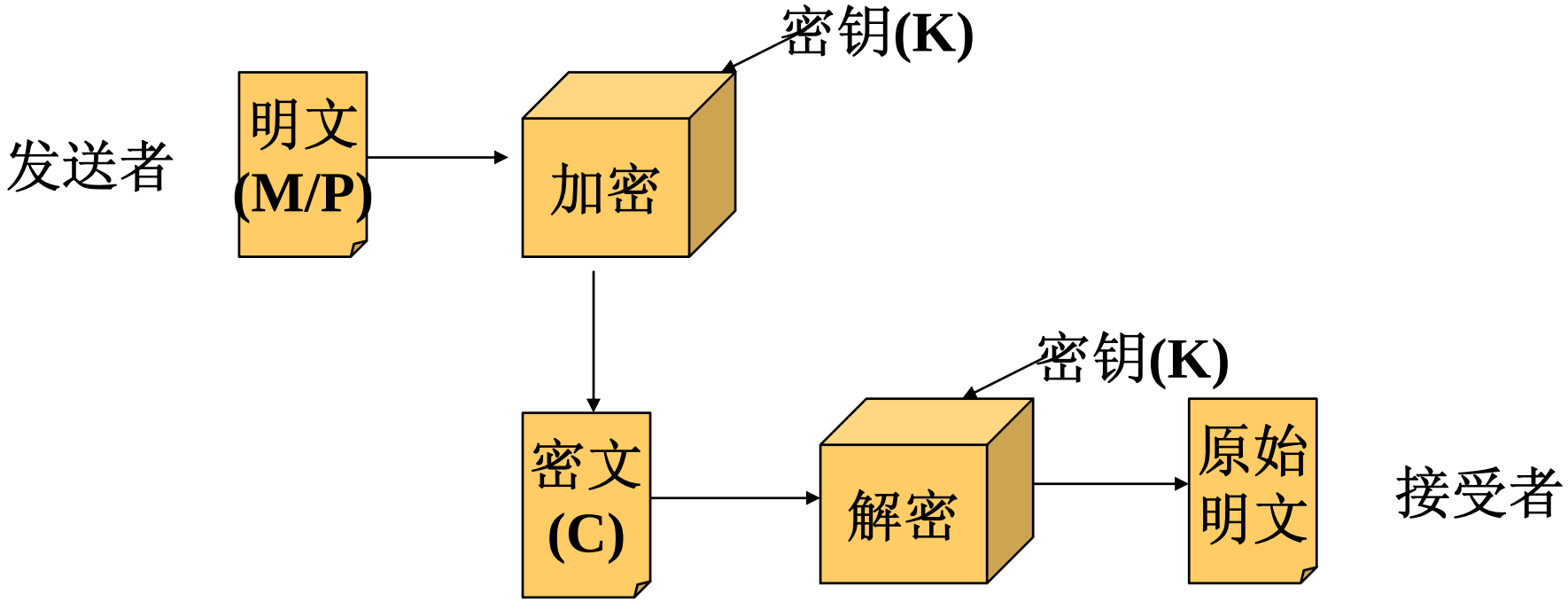
cshao@gdut.edu.cn

第二章 密码学概论

本章内容

- 一 概述
- 二 古典密码体制
- 三 对称密码体制
- 四公钥密码体制

一 概述--加密解密基本过程



使消息保密的技术和科学叫做密码编码学

破译密文的科学和技术为密码分析学

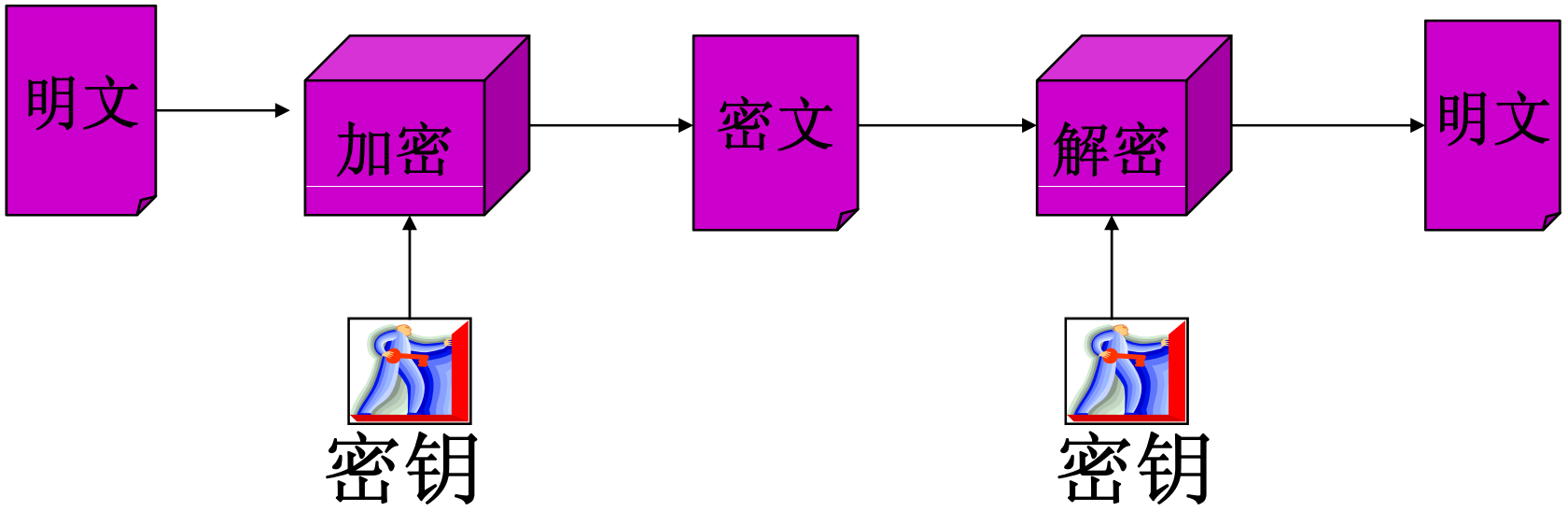
密码学作为数学的一个分支，包括密码编码学和密码分析学两部分。

一 概述--加密解密基本过程

- 密码算法也叫密码，是用于加密解密的函数。
- 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。
- 现代密码学用密钥解决这个问题，密钥用**K**表示，**K**的可能取值范围叫做密钥空间。
  - 如，密钥**56**位，2^{56}种
- 基于密钥的算法通常有两类
  - 对称密钥算法
  - 公开密钥算法

一 概述—密码体制分类

- *对称密码体制*
- 又叫传统密码算法:加密密钥能够从解密密钥中推算出来，反过来也成立。在大多数对称算法中，加解密密钥是相同的。
- 这些算法也叫秘密密钥算法或单密钥算法，它要求发送者和接收者在安全通信之前，商定一个密钥。



•对称密码体制

•对称算法的安全性依赖于密钥，泄漏密钥就意味着任何人都能对消息进行加/解密。

•加密和解密用函数表示为：

$$\bullet E_K(M)=C$$

$$\bullet D_K(C)=M$$

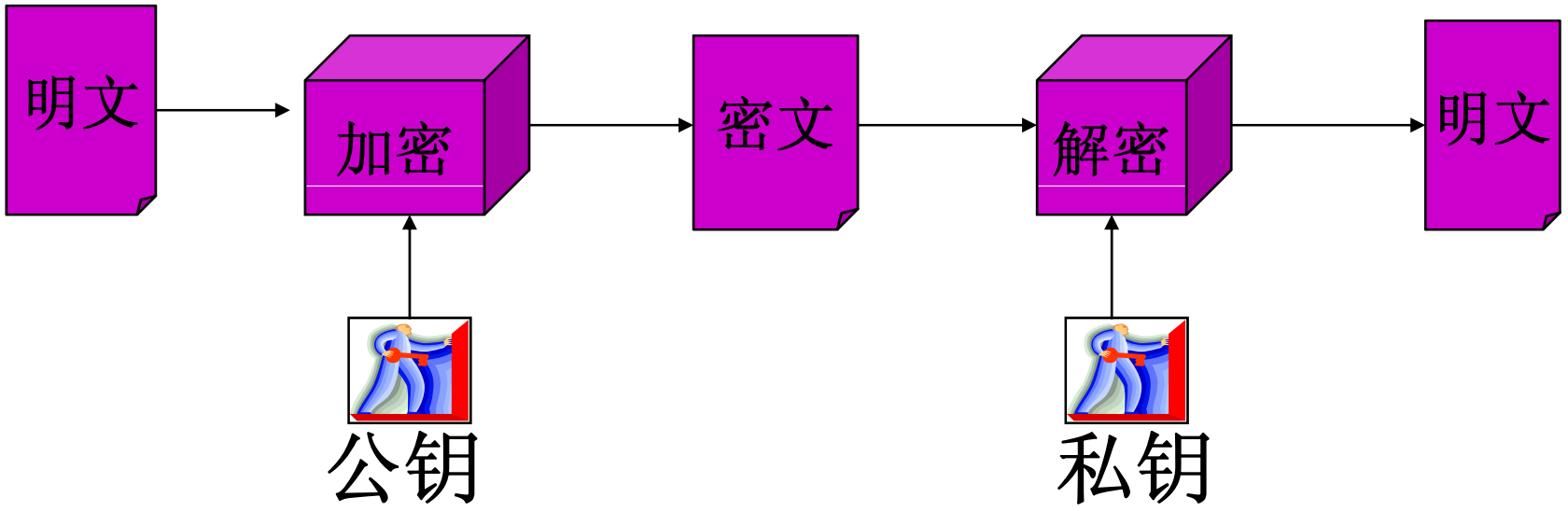
•对称算法可分为两类

- 一次只对明文中的单个位(有时对字节)运算的算法称为序列算法(stream algorithm)或序列密码(stream cipher)。
- 另一类算法是对明文的一组位进行运算，这些位组称为分组，相应的算法称为分组算法(block algorithm)或分组密码(block cipher)。

一 概述—密码体制分类

- 公钥密码体制

- 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(至少在合理假定的长时间内)。
- 加密密钥K1与相应的解密密钥K2不同，函数表示为
- E_{K1}（M）=C
- D_{K2}（C）=M



一 概述—常用的密码分析攻击

- 唯密文攻击(ciphertext-only attack)
- 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，以便可采用相同的密钥解出其他被加密的消息。
- 密码分析者已知的素材是：①. 加密算法、②. 待破译的密文。

一 概述—常用的密码分析攻击

- 已知明文攻击(known-plaintext attack)

•密码分析者已知的素材是：①. 加密算法、②. 待破译的密文、③. 由密钥形成的一个或多个明文-密文对。

•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。

一 概述—常用的密码分析攻击

- 选择明文攻击(chosen-plaintext attack)
- 密码分析者已知的素材是：①. 加密算法、②. 待破译的密文、③. 由密码分析者选择的明文，连同它对应的由其密钥生成的密文。
- 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。

一 概述—常用的密码分析攻击

- ***选择密文攻击(chosen- ciphertext attack)***
- 密码分析者素材的东西是：①.加密算法、②.待破译的密文、③.由密码分析者选择的猜测性密文，连同它对应的由密钥生成的已破译的明文。

一 概述—常用的密码分析攻击

- *自 适 应 选 择 明 文 攻 击 (adaptive-chosen-plaintext attack)。*

- 密码分析者不仅能选择被加密的明文，而且也能基于以前加密的结果修正这个选择。在自适应选择密文攻击中，可选取较小的明文块，然后再基于第一块的结果选择另一明文块，以此类推。

- 密码分析者已知的素材是：①. 加密算法、②. 待破译的密文、③. 由密码分析者选择的明文，连同它对应的由密钥生成的密文、④. 由密码分析者选择的猜测性密文，连同它对应的由密钥生成的已破译的明文。

一 概述—常用的密码分析攻击

- 上述攻击中，唯密文攻击是困难的攻击，因为密码分析者可利用的信息最少。
- 若密码分析者除了密文外，还能获取一段或多段明文，或者可能知道某种明文模式将出现在某个消息中，此时可以进行已知明文攻击。
- 如果密码分析者能在加密源系统中插入由分析员选择的消息，则可进行选择明文攻击。
- 其他两种类型的密码分析：选择密文攻击和自适应选择明文攻击较少使用，但无论如何是可能的攻击途径。

一 概述—常用的密码分析攻击

- 对密码设计者
  - 被设计的加密算法一般要能经受得住已知明文攻击。
- 对密码分析者
  - 则有两条必须遵循的基本准则
    - ①是破译该密码的成本不能超过被加密信息的价值；
    - ②是破译该密码的时间不能超过被加密信息有用的生命周期。

本章内容

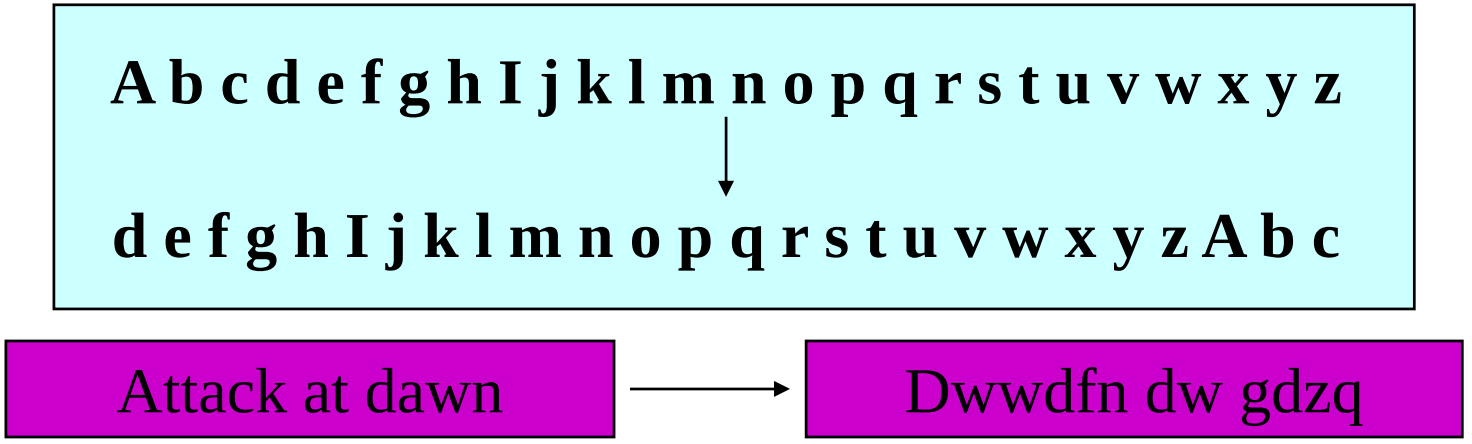
- 一 概述
- *二 古典密码体制*
- 三 对称密码体制
- 四 公钥密码体制

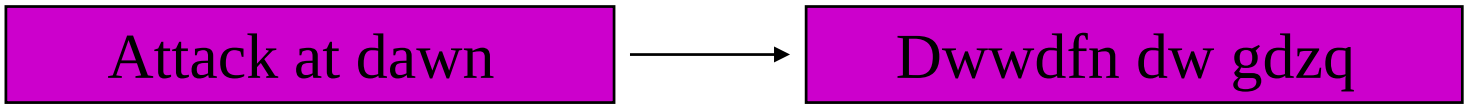
二 古典密码体制

- 古典密码是基于字符替换的密码
- 加密技术的两个基本构造模块是替代和置换，下面介绍的Caesar（恺撒）密码、Vigenere（维吉尼尔）密码、Hill密码、转轮机都属于使用替代技术的加密技术。

二 古典密码体制—恺撒密码

- 基本原理
  - 在开拓罗马帝国的时候，恺撒担心信使会阅读他送给士兵的命令，因此发明了对命令进行加密的算法——恺撒密码器。
  - 恺撒密码器非常简单：把字母表中的每个字母向前循环移动3位。





- 加密信息将字母向前移动三位
- 解密信息将字母向后移动三位
- 移动的位数是关键，称之为密钥
- 加密和解密的密钥是相同的，我们称之为对称密码器

二 古典密码体制—恺撒密码

- 数学表达

- 为每个字母分配一个数值（如a=0、b=1等），对每个明文字母p，用密文字母C代替，则恺撒密码的加密算法可表示为：

  - $C=E(p)=(p+3) \mod 26$

- 而解密算法则可表示为

  - $p=D(C)=(C-3) \mod 26$

二 古典密码体制—恺撒密码

- 改进的恺撒密码系统

- 明文的发送方和接收方事先协商好一个密钥。
- 用k(1≤k≤25)表示密钥，则通用的恺撒加密算法表示为：

$$C=E(p)=(p+k) \bmod 26$$

- 相应的，解密算法可表示为：

$$p=D(C)=(C-k) \bmod 26$$

二 古典密码体制—恺撒密码

- 攻击方式
- 唯密文攻击
  - 攻击者只有密文消息，他的策略就是穷尽搜索25个可能密钥。
  - 加快攻击速度：
  - 只须利用猜想的密钥对前5个字符进行解密
  - 每个字母在短文中出现的概率不相同的

- 已知明文攻击：如果攻击者知道一个字符以及它的密文，那么他很快就可以通过加密字符与解密后的明文之间的间距推出密钥是多少。例如，如果知道t（=19）加密后变成D（=3），那么密钥
  - k=（3－19）mod 26=－16 mod 26=10。
- 选择明文攻击：选择一个字符a作为明文，则由密文可以推导出密钥。例如，如果密文是H，则密钥k就是7。
- 选择密文攻击：选择字符A作为密文，则由明文也可推导出密钥。例如，如果明文是h，则密钥
  - k=0-7 mod 26=19。
- 恺散密码虽然脆弱，仅对明文进行了不透明的封装，但它可防止消息明文被人意外的获取。

二 古典密码体制—Playfair密码

- 基本原理
  - 多字母加密密码：将明文中的双字母组合作为一个加密单元对待，并将这些单元转换为密文双字母组合。
  - **Playfair**算法基于一个**5×5**的字母矩阵，该矩阵使用一个关键词构造，方法是按从左到右、从上到下顺序，填入关键词的字母（去除重复字母）后，将字母表其余字母填入。

- *基本原理*

  - 例如关键词取为monarchy时，字母矩阵为

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

加密方法是先将明文按两个字母一组进行分组，然后在矩阵中找对应的密文，取密文的规则如下：
(1) 若明文分组出现相同字母在一组，则在重复的明文字母中插入一个填充字母（譬如k）进行分隔后重新分组（如balloon被重新分组为ba lk lo on）；

(2) 若分组到最后一组时只有一个字母，则补充字母**k**；
(3) 若明文字母在矩阵中同行，则循环取其右边字母为密文（如ar被加密为RM）；
(4) 若明文字母在矩阵中同列，则循环取其下边字母为密文（如um被加密为MC）；
(5) 若明文字母在矩阵中不同行不同列，则取其同行且与下一字母同列的字母为密文（如hs被加密为BP，ea被加密为IM或JM）。

- *例*
- 明文we are discovered save yourself
- 分组成为：
- we ar ed is co ve re ds av ey ou rs el fk
- 用上述矩阵加密后的密文为：UG RM KC SX HM UF KM TB XO GC VM TA LU GE。

$$\begin{pmatrix} M & O & N & A & R \\ C & H & Y & B & D \\ E & F & G & I/J & K \\ L & P & Q & S & T \\ U & V & W & X & Z \end{pmatrix}$$

二 古典密码体制—Playfair密码

- **Playfair**密码比恺撒密码前进了一大步，一方面是它改变了单字母替代密码的频率分布，另一方面是双字母组合有**676**种，识别各种双字母组合要比识别**26**个单字母困难得多。
- 因此**Playfair**密码过去很长一段时期被认为是不可破译的，第一次世界大战中被英国陆军作为最好的密码系统使用，在第二次世界大战中也曾被美国陆军和盟军大量使用。

二 古典密码体制—Vigenere密码

- 密钥
  - 一个字符序列$k=(k_1, k_2, \ldots, k_m)$，其中m为任意值。
- 明文$X=(x_1, x_2, \ldots, x_n)$将被分为长度为m的段，如果消息的长度恰好不是m的倍数，则在末尾填充随机字符。
-

- *加密函数* $E_k(x_1, x_2, \ldots, x_n)=$

- $((x_1+k_1) \bmod 26,\ (x_2+k_2) \bmod 26,\ \ldots,\ (x_m+k_m) \bmod 26,$
- $(x_{m+1}+k_1) \bmod 26,\ (x_{m+2}+k_2) \bmod 26,\ \ldots,\ (x_{2m}+k_m) \bmod 26,$
- $(x_{2m+1}+k_1) \bmod 26,\ (x_{2m+2}+k_2) \bmod 26,\ \ldots,\ (x_{3m}+k_m) \bmod 26,$
- ……
- $(x_{N-m+1}+k_1) \bmod 26,\ (x_{N-m+2}+k_2) \bmod 26,\ \ldots,\ (x_N+k_m) \bmod 26$)

- 密钥的第一个字符被加到明文的第1个、第(m+1)个、第(2m+1)个、第(3m+1)个字符上（进行mod 26运算）
- 密钥的第二个字符被加到明文的第2个、第(m+2)个、第(2m+2)个、第(3m+2)个字符上，依此类推。

- *解密函数D_k*假设密文Y=(y_1, y_2, …, y_n)，则解密函数为：$D_k(y_1, y_2, \ldots, y_n)$=
- $((y_1-k_1) \bmod 26, (y_2-k_2) \bmod 26, \ldots, (y_m-k_m) \bmod 26,$
- $(y_{m+1}-k_1) \bmod 26, (y_{m+2}-k_2) \bmod 26, \ldots, (y_{2m}-k_m) \bmod 26,$
- $(y_{2m+1}-k_1) \bmod 26, (y_{2m+2}-k_2) \bmod 26, \ldots, (y_{3m}-k_m) \bmod 26,$
- ……
- $(y_{N-m+1}-k_1) \bmod 26, (y_{N-m+2}-k_2) \bmod 26, \ldots, (y_N-k_m) \bmod 26$)

- *例子*
  - 为了容易记住密钥，常常使用英文单词来充当Vigenere密码的密钥。
  - 使用密钥为vector，用数值表示则k=(21, 4, 2, 19, 14,17),m=6,来加密明文：*here is how it works*。

密钥的第一个字符被加到明文的第1个、第(m+1)个、第(2个、第(3m+1)个字符上（进mod 26运算）

密钥的第二个字符被加到明文的第2个、第(m+2)个、第(2m+2)个、第(3m+2)个字符上，

依此类推

则第一个明文字符用其后面的第21个字符来代替(即向后移21位)，相应的，第二个明文字符则向后移4位，第三个字符向后移2位，以此类推。当用完密钥k的最后一位时，又从密钥的第一位开始，如此循环下去。因此，第7个明文字符被向后移21位，第8个明文字符向后移4位

- *例子*
- 具体加密过程如下：
- 明文：**h e r e i s h o w i t w o r k s**
- 密钥：**21 4 2 19 14 17 21 4 2 19 14 17 21 4 2 19**
- 密文：**C I T X W J C S Y B H N J V M L**
- **Vigenere**密码的强度在于对每个明文字母有多个密文字母对应，因此该字母的频率信息是模糊的(如，e, s)。
- 维吉尼亚（Vigenere）密码是一种多表加密算法，在密文的不同位置出现的字符通常不是以同样的方式加密的，但它是一种周期密码，如果两个同样的字符出现的间隔固定，并且为密钥长度的倍数，则它们将以同样的方法进行加密。

二 古典密码体制—Hill密码

- **Hill**密码也是一种多字母替代密码，它由数学家**Lester Hill**于**1929**年研制成功。
- 该密码算法取**m**个连续的明文字母，并用**m**个密文字母代替，用向量或矩阵表示为：

$$\begin{pmatrix} C_1 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \end{pmatrix}$$

C和P是长度为3的列向量，分别表示明文和密文，K是3×3矩阵，表示加密密钥，加密操作要执行模26运算。

- 例如，如果使用的加密密钥是

$$k=\begin{pmatrix}17 & 17 & 5\\ 21 & 18 & 21\\ 2 & 2 & 19\end{pmatrix}$$

- 对明文：pay more money进行加密，则先按三个字母一组分组（不足三个时补字母x），然后对每组字母求密文。
- 该明文的前三个字母表示为pay=$(15\ \ 0\ \ 24)^T$，计算密文的过程如下：
- $K(15\ \ 0\ \ 24)^T = (375\ \ 819\ \ 486)^T \bmod 26 = (11\ \ 13\ \ 18)^T$ = LNS
- 如此类推，可得密文为：LNS HDL EWM TRW。
-

- 解密时使用逆矩阵，

$$K^{-1} = \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix}$$

- 对密文 $(11\ \ 13\ \ 18)^T$，做运算
  - $K^{-1}(11\ \ 13\ \ 18)^T \bmod 26 = (431\ \ 494\ \ 570)^T = (15\ \ 0\ \ 24)^T = pay$
- Hill密码的强度在于完全隐藏了单字母的频率。

本章内容

- 一 概述
- 二 古典密码体制
- *三 对称密码体制*
- 四 公钥密码体制

三 对称密码体制—DES

- 数据加密标准（**Data Encryption Standard，DES**）
- **DES**成为一个使用规模庞大的信息安全处理标准。美国的许多专用系统纷纷宣布采用**DES**作为该系统的信息处理安全标准，如美国的海军系统、金融系统等。计算机厂商也大量生产以**DES**为基本算法的专用机、芯片、软件，形成了一个以**DES**算法为核心的数据安全加密市场。
- **DES**算法每隔五年就被重新审查一次，分别在**1983**年、**1987**年、**1992**年通过了对其安全性的评估，作为标准一直使用到**1998**年。
- 随着计算机处理能力的提高，对**DES**的攻击不断显示出对其不利的因素，主要是其密钥过短，仅有**56**位。
- 最终，**NIST**于**1997**年发布公告，征集新的数据加密标准作为联邦信息处理标准以代替**DES**。新的数据加密标准称为**AES**
- 尽管如此，**DES**仍然具有重要的参考价值，它对于我们掌握分组密码的基本理论与设计思想具有重要的意义。

三 对称密码体制—DES

•DES的描述

•分组加密算法，它以64位为分组对数据进行加密

•DES是一个对称算法：加密和解密用的是同一算法

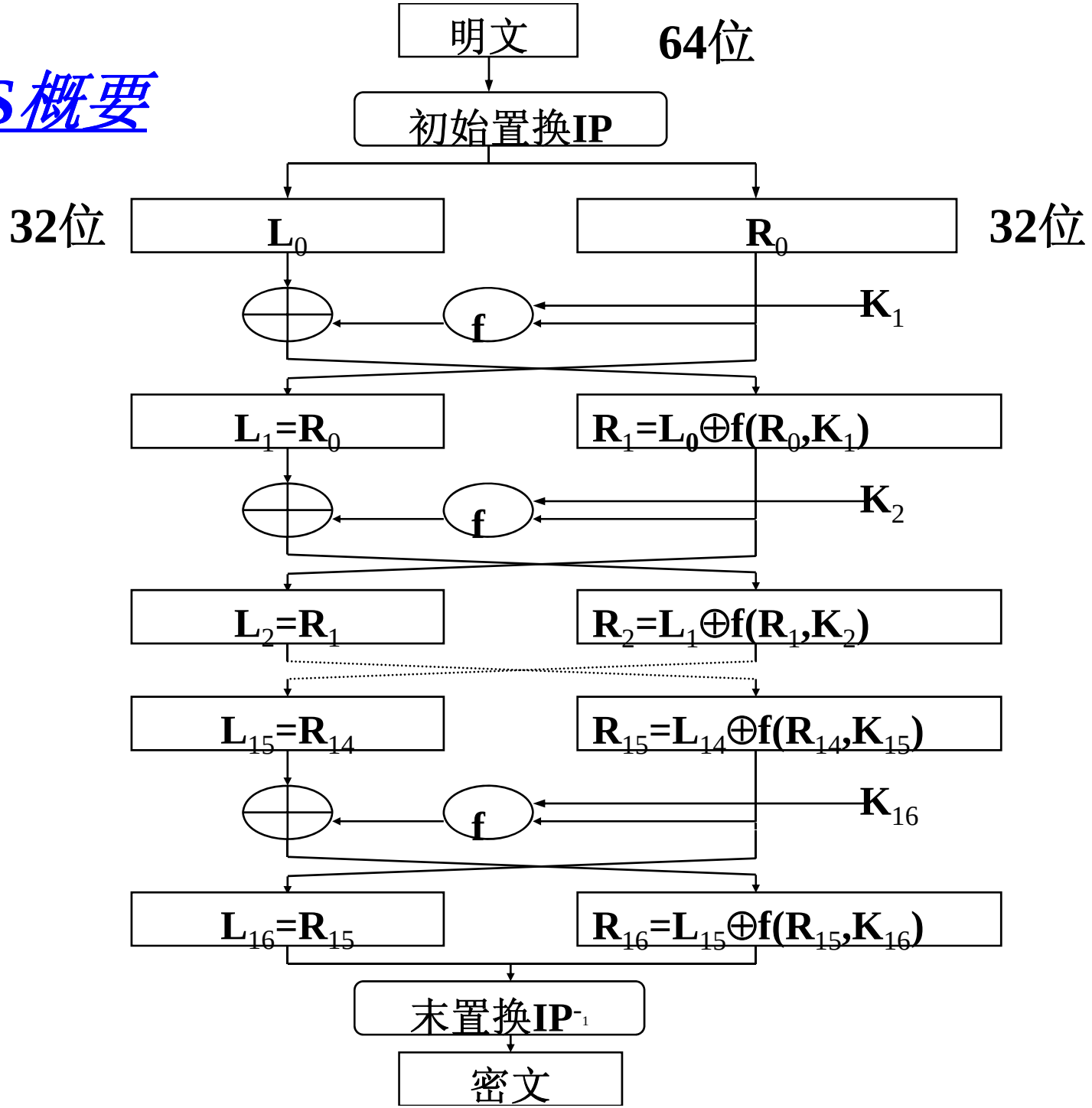
•密钥的长度为56位。所有的保密性依赖于密钥。

•算法是混乱和扩散的组合。

•DES基本组建分组是这些技术的一个组合(先代替后置换)，它基于密钥作用于明文，“轮(round)”。DES有16轮，这意味着要在明文分组上16次实施相同的组合技术

•此算法使用了标准的算术和逻辑运算，而其作用的数也最多只有64位，因此用70年代末期的硬件技术很容易实现。算法的重复特性使得它可非常理想地用在一个专用芯片中。

DES概要

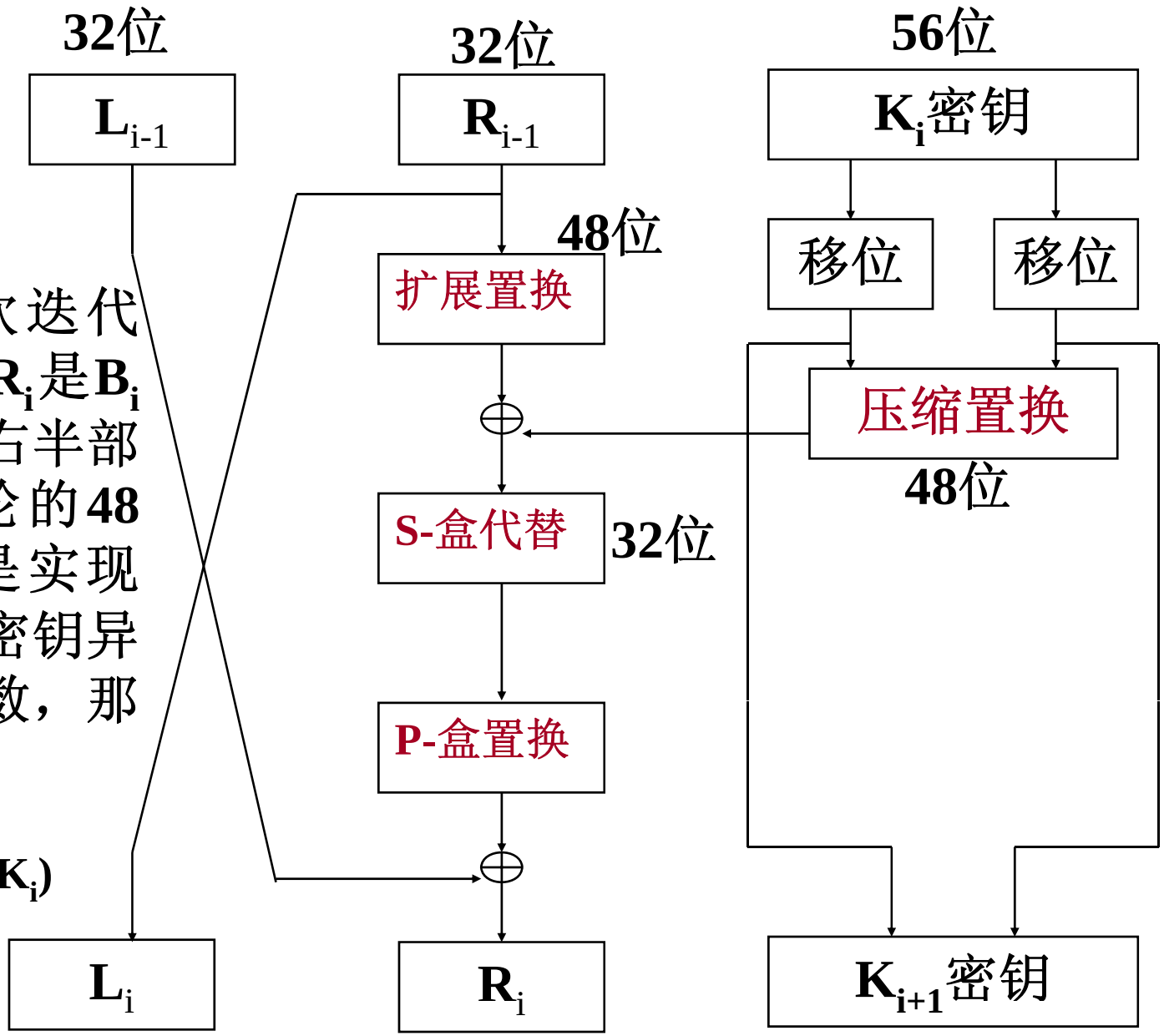

明文
64位
初始置换IP
32位
L_0
R_0
32位
f
K_1
$L_1=R_0$
$R_1=L_0⊕f(R_0,K_1)$
f
K_2
$L_2=R_1$
$R_2=L_1⊕f(R_1,K_2)$
$L_{15}=R_{14}$
$R_{15}=L_{14}⊕f(R_{14},K_{15})$
f
K_{16}
$L_{16}=R_{15}$
$R_{16}=L_{15}⊕f(R_{15},K_{16})$
末置换IP^{-1}
密文

一轮DES

假设B_i是第i次迭代的结果，L_i和R_i是B_i的左半部分和右半部分，K_i是第i轮的48位密钥，且f是实现代替、置换及密钥异或等运算的函数，那么每一轮就是：

$L_i = R_{i-1}$

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



三 对称密码体制—DES

•DES解密

•加密和解密可使用相同的算法。

•二者的唯一不同之处是密钥的次序相反。这就是说，如果各轮的加密密钥分别是K_1，K_2，K_3…，及K_{16}那么解密密钥就是K_{16}，及K_{15}，K_{14}……，及K_1。

三 对称密码体制—DES

- **DES的安全性**
- 1、弱密钥
  - 由于算法各轮的子密钥是通过改变初始密钥这种方式得到的，因此有些初始密钥成了弱密钥**(weak key)**。
  - 密钥初始值分成了两部分，每部分各自独立地移动。如果每一部分的所有位都是**0**或**l**，那么算法的任意一周期的密钥都是相同的。当密钥是全**1**、全**0**或者一半是全**1**、一半是全**0**时，会发生这种情况。（**4**个）

- ***DES的安全性***
- 1、弱密钥
  - 还有一些密钥对里的一个密钥能解密另一个密钥加密的信息。
  - 这些密钥只产生2个不同的子密钥，而不是16个不同的子密钥。算法中每个这样的子密钥都使用了8次。这些子密钥叫做半弱密钥**(semiweak key)**（12个）
  - 也有只产生4个子密钥的密钥，每个这样的子密钥在算法中使用了4次。（48个）
  - **DES的64个弱密钥和半弱密钥相对于总数为72，057，594，037，927，936个可能密钥的密钥集**

•*DES的安全性*

•2 补密钥

•将密钥的每一位取反

•假设用原来的密钥加密一个明文分组得到一个密文分组，那只用该密钥的补密钥加密将该明文分组的补使得到该密文分组的补。

•如果x′是x的补，则有如下的等式：

• $$E_K(P)=C$$

• $$E_{K'}(P')=C'$$

•这表明，对DES的选择明文攻击仅需要测试其可能的2^{56}个密钥的一半，2^{55}个即可。

- ***DES的安全性***
- **3 代数结构**
  - 将所有可能的**64**位明文分组映射到所有可能的**64**位的密文分组共有2^{64}！种不同的方法。
  - **56**位密钥的**DES**算法为我们提供了2^{56} **(**大约10^{17}**)**个这种映射关系
  - 采用多重加密看起来似乎可以获得这些可能的映射关系中的更多部分。然而，这仅在**DES**运算不具有某种代数结构的条件下才成立。

– 如果**DES**是闭合的**(closed)**，那么对任意的K_1和及K_2，必将存在K_3使得

$$E_{K_2}(E_{K_1}(P))=E_{K_3}(P)$$

– 换言之，**DES**对一组明文用K_1加密后再用K_2加密，这等同于用K_3对该明文进行加密。

– 如果**DES**是纯洁的**(pure)**，那么对任意的及K_1，K_2和K_3，必将存在K_4使得

$$E_{K_3}(E_{K_2}(E_{K_1}(P)))=E_{K_4}(P)$$

– 三重的加密将是无用的。

- ***DES的安全性***
- **4 密钥的长度**
  - 穷举攻击
- **5 迭代的次数**
  - **DES为什么是16轮而不是32轮?经过5轮迭代后，密文每一位基本上是所有明文和密钥位的函数，经过8轮迭代后，密文基本上是所有明文和密钥位的随机函数。那为什么算法在8轮后还不停止呢?**
  - **近年来，多种降低轮数的DES已被成功地攻击。1982年3轮或4轮DES就被轻易地破译了。几年后，6轮DES也被破译了。Biham和Shamir的差分密钥分析同样也了阐明了这一点：对低于16轮的任意DES的已知明文攻击要比穷举攻击有效。有趣的是当算法恰好有16轮时，只有穷举攻击最有效的。**

- ***DES的安全性***
- **6 S-盒的设计**
  - 设计者表示，算法设计原理是“敏感的”，不宜公之于众。许多密码学家担心设计S-盒时隐藏了“陷门”，使得只有他们才可以破译算法。
  - **70**年代中叶，**Bell**实验室研究**S-**盒的运算，尽管他们都发现了不能解释的特征，但研究中并没有找到弱点。
  - 在差分分析公开后IBM公布了S-盒和P-盒的设计准则。

三 对称密码体制—DES

- 多重DES
  - 对它最有效的攻击仍然是穷举攻击
  - 多重加密(multiple encryption)是一种组合技巧：用同一个算法在多重密钥的作用下多次加密同一个明文分组。一定要保证多重密钥是不同且相互独立的。
  -

- ***多重DES***
  - **1** 双重加密
  - 双重加密是用不同的密钥对一个分组进行两次加密。解密是一个相反的过程 。

$$C = E_{K_2}(E_{K_1}(P))$$

$$P = D_{K_1}(D_{K_2}(C))$$

  - **DES**没有某种代数特征，则对以上合成的双重加密的密文分组，利用穷举搜索的方法破解它非常困难。
  - 它不只需要2^n次尝试(***n***是密钥的位长度)，而是需要2^{2n}次尝试。如果算法是一个**64**位密钥算法，那么，要找到双重加密的密钥需要2^{128}次尝试

- ***多重DES***
- **2** 三重加密
  - 用两个密钥对一个分组进行三次加密
    - 发送者首先用第一个密钥加密，然后用第二个密钥解密，最后再用第一个密钥加密。
    - 接收者首先用第一个密钥解密，然后用第二个密钥加密，最后再用第一个密钥解密。
    - 安全性更高的是用三个独立密钥进行三重加密
    - 采用三重DES加密，因为DES不是一个群，所以得到的密文更难用穷举搜索破译

本章内容

- 一 概述
- 二 古典密码体制
- 三 对称密码体制
- *四 公钥密码体制*

四 公钥密码体制(背包算法)

- 为第一个推广的公开密钥加密算法。
- 其安全性基于背包问题，这是个**NP**完全问题
- 虽然后来发现这个算法不安全，但仍值得研究，因为它表示了如何将**NP**完全问题用于公开密钥算法。
- **0-1**背包问题：给定一些物体，每个物体有不同的重量，是否有可能将这些物体放入一个背包，使背包的重量等于一个给定的值。

四 公钥密码体制(背包算法)

- 例如，这些物体的重量分别为**1,5,6,11,14,20**则可将重**5,6,11**的物体放入，装成一个重**22**的背包。但是无法装成一个重**24**的背包。
  - 背包问题：等于一个给定的值
  - 解为选择物品装入的情况，装入用**1**，未装入用**0**。例子中对给定值**22**的解为**{0,1,1,1,0,0}**
- 这个问题需要的时间随物体的数量的增加成指数时间

- ***基本思想***
- 一块明文的长度等于物体的个数，表示从中选取物体装入背包，密文为和，密钥为背包问题中物品重量序列。
- 明文:**1 1 1 0 0 1　　0 1 0 1 1 0　　0 1 1 0 0 0**
- 背包问题中物品重量序列(密钥)：
- **1 5 6 11 14 20　　1 5 6 11 14 20　1 5 6 11 14 20**
- 密文：**1+5+6+20=32　　5+11+14=30　　5+6=11**
- 安全性体现为：
- 若攻击者获得密文(背包问题中的指定值)，密钥(重量序列)也无法在线性时间内求明文(物品的装入情况)。

- ***基本思想***
- 算法的关键是有两个不同的背包重量序列，这两个重量序列对于给定的相同的值，解相同(物品的装入情况相同)。
  - 如： **{1,3,6,13,27,52}**和**{1 2 4 16 102 37}**
  - 给定值**22**
  - 对于第一个背包的装入情况是**{0,1,1,1,0,0}**
  - 对于第二个背包的装入情况是**{0,1,1,1,0,0}**
- 一个可以在线性时间求解的(私钥)用来解密，一个是难解的背包(公钥)用来加密。

- ***1 构造可在线性时间求的背包***
- 易解的的背包问题为：若物品的重量列表为一个超递增序列，则该背包问题很容易解。
- 超递增序列：其中每个元素都大于前面所有元素的和。如：**1 3 6 13 27 52**
- 而**1 3 4 9 15 25**就不是
- 超递增背包问题很好解。
  - 将背包重量与序列中最大元素相比较，如果总重量小于这个数则这个数不在背包中。
  - 如果总重量大于或等于这个数，则这个数在背包中，用总重量减去这个数。
  - 再与序列中下一个最大数比较。重复比较，直到比较完为止。如果背包总重量减到**0**则得到背包的解，如果不为**0**则无解。

- 如：考虑一个背包总重量为**70**，重量序列为**{2,3,6,13,27,52}**
- 最大重量为**52**小于**70**因此**52**在此背包中
- 下一个重量**27**大于**70－52＝18**，不在包中
- 继续比较下去总重量减少到**0**表示解已找到
- 可把**70**看作密文，其所对应的明文为**110101**

- 非超递增背包是一个难问题
- 背包算法先找到一个超递增背包的重量序列作为私钥，再由此构造一个有相同解的一般背包问题的序列作为公钥。
- ***2从私钥构造公钥***
- 选择一个超递增序列作为私钥**{2,3,6,13,27,52}**
- 然后每一个值都乘以一个数**n**，对**m**求余。
- 去作模的数**m**应该大于序列中所有数的和
- 如**m=105,n**应该与**m**互素，如取**n=31**
- 则一般背包问题序列为：
- **2×31 mod 105=62 3×31 mod 105=93**
- **6×31 mod 105=81 13×31 mod 105=88.…**

- 序列为：**{62 93 81 88 102 37}**作为公钥
- ***3 加密***
- 要加密一个二进制消息，先将它分成长度和背包序列相同的块。**1**表示在包中，**0**不在包中
- 消息：**011000 110101 101110**
- 公钥：**{62 93 81 88 102 37}**
- **011000**对应 **93＋81＝174**
- **110101**对应**62＋93＋88＋37＝280**
- **101110**对应**62＋81＋88＋102＝333**
- 则密文为**174，280，333**

- ***4 解密***
- 消息的合法接受者知道私钥，**m**和**n**的值
- 先计算n^{-1} (乘法逆元)满足$n \times n^{-1} \bmod m = 1$
- 将密文的每一个值与n^{-1}相乘模**m**，然后用私钥来求解得到明文的解
- 上例中私钥**{2,3,6,13,27,52}**，**m=105,n=31**，$n^{-1}=61$
- 密文为**174**，**280**，**333**
- 则密文必须和**61**相乘模**105**
- **174×61 mod 105=9=3+6** 对应于**011000**
- **280×61 mod 105=70=2+3+13+52** 对应于**110101**
- **333×61 mod 105=43=2+6+13+27** 对应于**101110**

求乘法逆元－辗转相除法

- **5×d mod 24＝1　　d=5**

余数	d	商		
24	**0**	**y**		
5	**1**	**4**		
4	**–4**	**1**	**4=24-5*4**	**d1=0-1*4=-4**
1	**5**	**4**	**1=5-4*1**	**d2=1-(-4*1)=5**
0				

31×d mod 105 =1　　d=-44+105=61

余数	d	商	
105	0	y	
31	1	3	
12	–3	2	12=105-31*3 d1=0-1*3=-3
7	7	1	7=31-12*2　d2=1-(-3*2)=7
5	-10	1	5=12-7*1　d3=-3-(7*1)=-10
2	17	2	2=7-5*1　d4=7-(-10*1)=17
1	-44	2	1=5-2*2　d5=-10-(17*2)=-44
0			

四 公开密钥算法(RSA算法)

- 第一个完善的公开密钥算法**RSA**
- 经受住了多年的密码分析。密码分析者既不能证明但也不能否定**RSA**的安全性。
- 其安全性基于大数分解的难度
- 求一对大素数的乘积很容易但是要做因式分解就难。因此可以把一对大素数的乘积公开作为公钥，而素数作为私钥。
- 从而从一个公开密钥和密文中恢复出明文的难度等价于分解两个大素数之积。

公开密钥n：两素数p和q的乘积(p,q必须保密)

e：与(p-1)(q-1)互素

私钥d：$e \times d \bmod [(p-1)(q-1)]=1$(辗转相除法)

等价表示为$d=e^{-1} \bmod [(p-1)(q-1)]$

加密：$c=m^e \bmod n$

解密：$m=c^d \bmod n$

- 例子：$p=47$ $q=71$ 则$n=pq=3337$ $(p-1)(q-1)=3220$
- 随机选取$e=79$ 则$79 \times d \bmod 3220=1$ $d=1019$
- 算法公开e和n，保密d，丢弃p和q
- 这样对于待加密的消息$m=688$
- $c=m^e \bmod n= 688^{79} \bmod 3337=1570$
- 解密： $m=c^d \bmod n=1570^{1019} \bmod 3337=688$

- **79×d mod 3220=1　　d=1019**
- 辗转相除法
- 余数　　d　　商　　**60=3220-79*40**
- **3220　　0　　y　　D1=0-1*40=-40**
- **79　　1　　40　　19=79-60*1**
- **60　　D1 =-40　　1　　D2=1-(-40*1)=41**
  19　　D2=41　　3　　3=60-19*3
  3　　D3=-163　　6　　D3=-40-(41*3)=-163
  1　　D4=1019　　3　　1=19-3*6
  0　　D4=41-(-163*6)=1019

$79 \times d \bmod 3220 = 1 \quad d = 1019$

$$3220 = 79 \times 40 + 60$$
$$79 = 60 \times 1 + 19$$
$$60 = 19 \times 3 + 3$$
$$19 = 3 \times 6 + 1$$
$$3 = 1 \times 3 + 0$$

$$60 = 3220 - 79 \times 40$$
$$19 = 79 - 60 \times 1$$
$$3 = 60 - 19 \times 3$$
$$1 = 19 - 3 \times 6$$

$$t = x$$
$$x = y$$
$$y = t - (m/n).y$$
$$= t - Q.y$$

$$1 = 19 - 3 \times 6 \text{ ----------} \mid x = 1, y = -6$$
$$\xrightarrow{3} 19 - (60 - 19 \times 3) \times 6 = 19 - 60 \times 6 + 18 \times 19$$
$$= -60 \times 6 + 19 \times 19 \text{ ----------} \mid x = -6, y = 1_x - 3_{(m/n)} \times (-6)_y = 19$$
$$\xrightarrow{19} 19 \times (79 - 60 \times 1) - 60 \times 6 = 19 \times 79 - 19 \times 60 - 6 \times 60$$
$$= 19 \times 79 - 25 \times 60 \text{ ----------} \mid x = 19, y = -6 - 1 \times 19 = -25$$
$$\xrightarrow{60} 19 \times 79 - 25 \times (3220 - 79 \times 40) = 19 \times 79 - 25 \times 3220 + 25 \times 40 \times 79$$
$$= -25 \times 3220 + 1019 \times 79 \text{ ----------} \mid x = -25, y = 19 - 40 \times (-25) = 1019$$

定理 5 设 a, b 是任意两个正整数，则

$$s_n a + t_n b = (a, b), \qquad (2)$$

对于 $n = 0,\ 1,\ 2,\ \ldots$，这里 s_n, t_n 归纳地定义为

$$\begin{cases} s_0 = 1,\ s_1 = 0, & s_j = s_{j-2} - q_{j-1}s_{j-1}, \\ t_0 = 0,\ t_1 = 1, & t_j = t_{j-2} - q_{j-1}t_{j-1}, \end{cases} \quad j = 2, 3, \ldots, n \qquad (3)$$

其中 q_j 是 (1) 式中的不完全商.

上述过程可以列成如下表格：

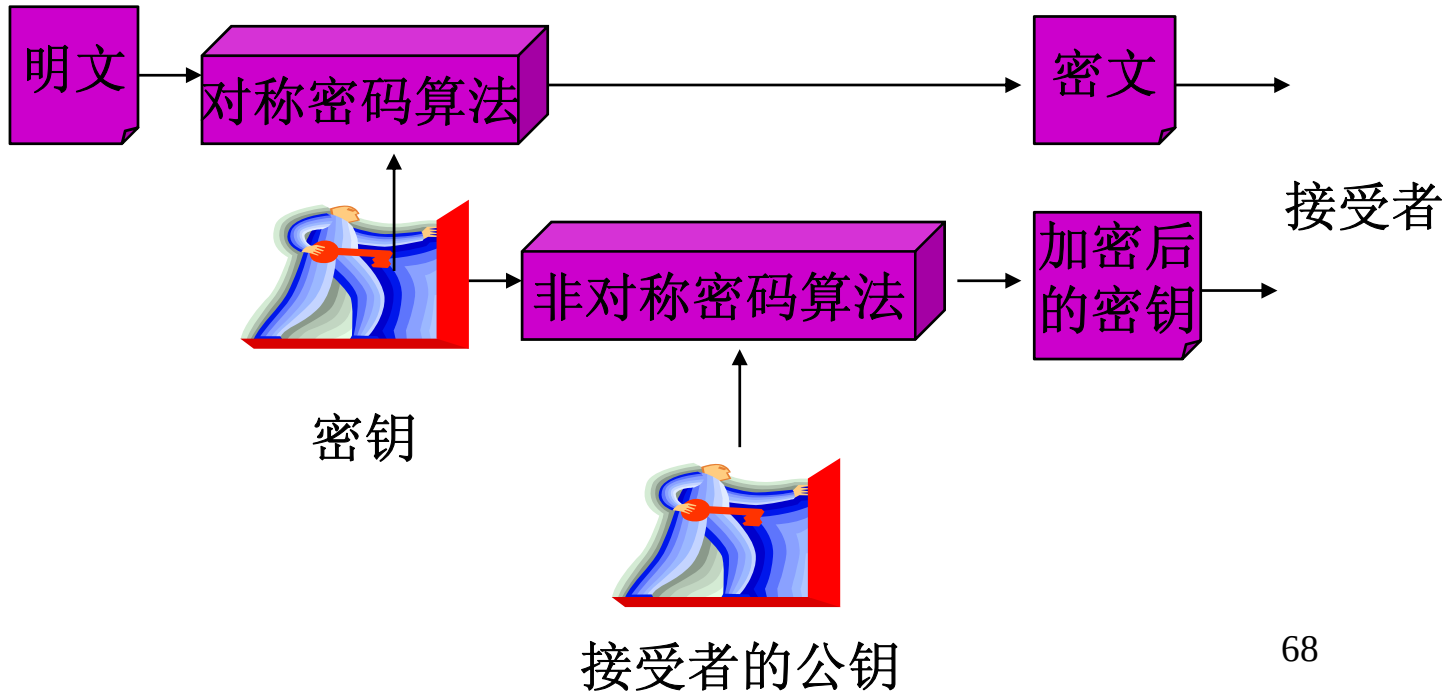
j	q_j	r_j	r_{j+1}	s_{j-1}	s_j	t_{j-1}	t_j
		a	b		1		0
1	q_1	r_1	r_2	1	0	0	1
2	q_2	r_2	r_3	s_1	s_2	t_1	t_2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
n	q_n	r_n	r_{n+1}	s_{n-1}	s_n	t_{n-1}	t_n

(5)

其中，对于 $j = 1,\ 2,\ \ldots,\ n$,

$$\begin{cases} q_j &= [\frac{r_{j-1}}{r_j}], \\ r_{j+1} &= r_{j-1} - q_j r_j, \\ s_j &= s_{j-2} - q_{j-1}s_{j-1}, \\ t_j &= t_{j-2} - q_{j-1}t_{j-1}t_j. \end{cases} \qquad (6)$$

- ***RSA的速度***
- 硬件实现时，**RSA比DES慢大约1000倍。**
- 软件实现时，**RSA比DES慢大约100倍。**
- 混合加密：采用对称密钥对消息进行加密，而用公钥对密钥本身进行加密
  - 有对称加密的速度和非对称的灵活性
  - 适用于**CPU**能力较弱的小型设备，如智能卡，掌上电脑等



- ***RSA的安全性***
- 三种攻击RSA算法的可能的方法是：
  - 强行攻击：这包含对所有的私有密钥都进行尝试。
  - 数学攻击：有几种方法，实际上都等效于对两个素数乘积的因子分解。
  - 定时攻击：这依赖于解密算法的运行时间。

- ***RSA的安全性—因子分解问题***
- 目前，**129**位十进制的模数是能分解的临界数，所以**n**应该大于这个数

公开密钥**n**：两素数**p**和**q**的乘积

e：与**(p-1)(q-1)**互素

私钥**d**： $e \times d \bmod [(p-1)(q-1)] = 1$

加密：$c = m^e \bmod n$

解密：$m = c^d \bmod n$

- 一个危险的情况是**p**和**q**比较接近时，分解特别容易
- **(p-q)/2**是一个很小的数；
- 而**[(p+q)/2]²**是比**n=pq**大一点的数,
- 则可以从$[\sqrt{n}$开始，对**k=1,2,3…,**寻找正整数**k**使
- **(**$[\sqrt{n}]$**+k)²-n**为一平方数记为**t²**
- 则大数**n**被分解为 $([\sqrt{n}]+k+t)([\sqrt{n}]+k-t)$
- 如：**n=200819**
- $[\sqrt{n}]$ **=448,** 搜索平方数**k=2**
- 即**(448+2)²-200819=1681=(41)²**
- 于是**200819=(448+2+41)(448+2-41)=491*409**

- ***RSA的安全性—定时攻击***
- 密码顾问**Paul Kocher**证明窥探者可以通过监视一台计算机解密报文所花费的时间来确定私有密钥。
- 定时攻击有点像一个窃贼通过观察一个人转动拨号盘转出一个个数字所用的时间来猜测一个保险箱的密码数字组合。
- 定时攻击不仅可以用于**RSA**，还可以用于其他公开密钥密码系统。

- ***RSA的安全性—定时攻击***
- 因此，如果当一个特定算法在比特为1的情况下很慢时，观察到的执行解密算法的时间总是很慢，那么这个比特就被认为是1。
- 如果对于整个算法许多观察到的执行时间都很快，那么这个比特就被认为是0。

- ***RSA的安全性—定时攻击***
- 简单的防范措施
  - *常数时间*：保证所有操作在返回一个结果之前花费同样多的时间。这是一个不大的更改，但是确实使算法性能下降。
  - *随机延时*：可以通过对算法增加一个随机延时来迷惑定时攻击者，这样可以得到更好的性能
  - *盲化*：在进行取幂运算之前先用一个随机数与密文相乘。这个处理防止了攻击者了解计算机中正在处理的密文，因此就防止了对于定时攻击来说关键的逐位分析。

•RSA的安全性—选择密文攻击

- Even在alice的通信过程中进行窃听，成功获取了一个用alice的公钥加密的密文c。Even想读出消息，从数学上讲为获得m。
- 首先选取一个随机数r，r小于n。使用alice的公钥计算：
- $x=r^e \bmod n$
- $y=xc \bmod n$
- $t= r^{-1} \bmod n$ 可推出$r=x^d \bmod n$
- 现在Even骗取alice对y的数字签名(用私钥加密的摘要)
- Alice发给Eve $u=y^d \bmod n$ (d是alice的私钥)
- 现在Eve计算：

$tu \bmod n = r^{-1} y^d \bmod n = r^{-1} x^d c^d \bmod n = c^d \bmod n$

公开密钥n：两素数p和q的乘积
e：与(p-1)(q-1)互素
私钥d： $e \times d \bmod [(p-1)(q-1)]=1$
加密：$c=m^e \bmod n$
解密：$m=c^d \bmod n$

- ***RSA的安全性—公共模数攻击***
- 对于一个可能的**RSA**的实现，若每个人有相同的**n**，但有不同的指数**e**和**d**。这种做法是不行的。
- **m**是明文消息，两个加密密钥为e_1和e_2，公共模数是**n**。两个密文为
- $c_1=m^{e1} \bmod n$；$c_2=m^{e2} \bmod n$
- 这样密码分析者知道**n**，e_1，e_2，c_1，c_2
- 若e_1和e_2互素，由扩展的欧几里德算法能找出**r**，**s**满足$r\times e_1+s\times e_2=1$
- 假定**r**是负数(**r**和**s**中必有一个是负数)，那么再用辗转相除法计算出c_1^{-1},有$(c_1^{-1})^{-r} \times c_2^{s}=m \bmod n$

- ***对RSA的攻击—低解密指数攻击***
- 如果**d**达到**n**的**1/4**大小，且**e**比**n**小，那么该方法可以恢复**d**。
- 假如**e**和**d**随机选择，或**e**的值非常小则该攻击很少发生。

四 公开密钥算法(ECC算法)

- 椭圆曲线密码体制是目前已知的公钥体制中，对每比特所提供加密强度最高的一种体制。
- 当RSA的密钥使用2048位时，ECC的密钥使用234位所获得的安全强度还高出许多。它们之间的密钥长度却相差达9倍，当ECC的密钥更大时它们之间差距将更大。

四 公开密钥算法(ECC算法)

- 有限域上椭圆曲线
  - 椭圆曲线是连续的 $y^2+axy+by=x^3+cx^2+dx+e$
  - 不适合用于加密，必须把椭圆曲线定义在有限域上。
  - 不是所有的椭圆曲线都适合加密。**$y^2=x^3+ax+b$**是一类可以用来加密的椭圆曲线，也是最为简单的一类。

四 公开密钥算法(ECC算法)

- 我们就把$y^2=x^3+ax+b$ 这条曲线定义在有限域上：
- 选择两个满足下列条件的小于p(p为素数)的非负整数a、b：
- $4a^3+27b^2 (\mod p) \neq 0$
- 则满足下列方程的所有点(x,y)，再加上无穷远点0，构成一条椭圆曲线。
- $y^2=x^3+ax+b (\mod p)$
- 其中x，y属于0到p-1间的整数,这条椭圆曲线记为Ep(a,b)。

四 公开密钥算法(ECC算法)

- 例 $y^2=x^3+x+1 (\mathrm{mod}\ 23)$, 记为$E_{23}(1,1)$。
- 对于椭圆群来说，我们只关心那些满足模p方程的处于从(0，0)到(p，p)的正方形中的整数。下表列出了组成$E_{23}(1,1)$的点(不包含O点)。

(0, 1)	(0, 22)	(1, 7)	(1, 16)	(3, 10)	(3, 13)	(4, 0)	(5, 4)	(5, 19)
(6, 4)	(6, 19)	(7, 11)	(7, 12)	(9, 7)	(9, 16)	(11, 3)	(11, 20)	(12, 4)
(12, 19)	(13, 7)	(13, 16)	(17, 3)	(17, 20)	(18, 3)	(18, 20)	(19, 5)	(19, 18)

四 公开密钥算法(ECC算法)

- 表是以下列方式创建的：
  - **(1)**对于每个满足0≤x<p的x，计算(x^3+ax+b) mod p。
  - 在本例中计算(x^3+x+1) mod 23
  - 设x=1,则计算上式的结果为3
  - **(2)**对于上一步骤得到的每个结果确定它是否有一个模p的平方根。 y^2 **mod 23=3**
  - 若没有，则曲线上没有与这一x相对应的点。
  - 若有，就有两个满足平方根运算的y值(y=0时只有一个平方根)。这些(x，y)值就是E_p(a，b)中的点。
  - y=7时 7^2 **mod 23=3**
  - y=16时 16^2 **mod 23=3**

四 公开密钥算法(ECC算法)

- 因此对于书中例子p67
- 椭圆曲线$E_{751}(-1，188)$为$y^2=x^3-x+188$，
- 取G=(0，376)
- X=0 $x^3-x+188$ mod 751=188
- y^2 **mod 751=188 y=376**

四 公开密钥算法(ECC算法)

- 椭圆曲线的特性是
  - 已知曲线上某两点相加的结果，欲反推出是哪两点相加是非常困难的。
  - 纯量相乘：某一点乘上一个数字。
  - **p1=2*p0**，无法由**p1**反推出**p0**

四 公开密钥算法(ECC算法)

- 对于所有的点P，Q∈E_p(a，b)有如下的加法规则：
  - ① P+0=P
  - ② 如果P=(x，y)，则P+(x，-y)=0，(x，-y)点是P的负点，记为-P。而且(x，-y)也在E_p(a，b)中。
    - 如p=(1,7) 则-p=(1,-7) 而 -7 mod 23=16,所以p的负点也可以是(1,16)
  - ③ 如果P=(x_1, y_1)，Q=(x_2, y_2)，则P+Q=(x_3, y_3)为
    - $x_3 = \lambda^2 - x_1 - x_2 \mod p$
    - $y_3 = \lambda (x_1 - x_3) - y_1 \mod p$
  - 其中，如果P≠Q，则 $\lambda = (y_2 - y_1)/(x_2 - x_1)$
  - 如果P=Q，则 $\lambda = (3x_1^2 + a)/(2y_1)$
  - 椭圆曲线上一个点Q被一个正整数k相乘的乘法被定义为k个Q相加，因而有3Q=Q+Q+Q等等

四 公开密钥算法(ECC算法)

- 椭圆曲线用于加密/解密
- 加密过程：
  - A选取一条椭圆曲线，并得Ep(a,b)，再选择Ep(a,b)的元素G，使得G的阶n是一个大素数(G的阶是指满足nG=O的最小n值)。
  - A秘密选择整数r_a(r_a为私钥)，计算$P_a=r_aG$(P_a为公钥)。然后，公开(p，a，b，G，P_a)，保密r_a。
  - B加密消息M。先把消息M变换成为Ep(a,b)中的一个点P_m，然后，选择随机数k，计算密文C_m ={kG，P_m+kP_a}，并将C_m发送给 A 。（如果k使得kG或者kP_a为O，则要重新选择k）

- *椭圆曲线用于加密/解密*
- 解密过程：
- A 收到 B 的密文$C_m = \{kG,\ P_m + kP_a\}$后，A 用这一对中的第一个点乘以 A 的私钥，再从第二个点中减去这个值：
  - $(P_m + kP_a) - r_a(kG) = P_m + krG - r_akG = P_m$

椭圆曲线用于加密/解密

解密过程安全性分析：

C_m ={kG， P_m +kP_a}

公开(**p，a，b，G， P_a**)

P_a=r_aG (r_a为私钥)

B通过对**P_m**加上**kP_a**而掩盖了**P_m**。除了 B 没有人知道**k**的值，因此即便**P_a**是公开密钥，也没有其他人能够去掉遮掩**kP_a**。

一个攻击者要恢复报文，他就必须在只知道 G 和**kG**的条件下计算出**k**，这被认为是十分困难的。

然而 B 也包括了一个“线索”，如果一个人知道私有密钥**r_a**，就能够去掉遮掩。

椭圆曲线用于加密/解密

- 例： A取p=751， Ep(-1， 188)， 即椭圆曲线为$y^2=x^3-x+188$，取G=(0，376)。A的公钥为=(201，5)。假定B已将欲发往A的消息嵌入到椭圆曲线上的点P_m=(561，201)，B选取随机数k=386，则：
  - kG=386(0，376)=(676，558)
  - P_m+kP_a=(562，201)+386(201，5)=(385，328)
- 因而B发送密文{(676，558)，(385，328)}。

四 公开密钥算法(ELGamal算法)

- 该算法比**RSA**算法稍微复杂一些，但仍然是一个十分基本的算法。
- **ELGamal**体制的安全性基于有限域上求解离散对数的困难性。
  - 素数**P**和另外一个整数**g**，**1<g<p-1.**
  - 给定整数**x**，求得**$y = g^x \bmod p$**,很容易。
  - 但是给定**y**，**p**，**g**求得**x**是困难的。

四 公开密钥算法(ELGamal算法)

- 密钥的产生
  - 要产生一对密钥，首先选择一大素数**p**(例如**$p>10^{150}$**)，
  - 再选取两个小于**p**的随机数**g**和**x**，并且**g**是**p**的本原根(意味着任何数**y**都可以表示为：**$y=g^L \bmod p$**)。
  - 然后计算**$y=g^x \bmod p$**
  - 于是，得到公钥**KU={y, g, p}**
  - 私钥**KR={x,p}**。

四 公开密钥算法(ELGamal算法)

- 加密
  - $y=g^x \bmod p$

- 把要加密的明文m编码为一个整数，在范围0<m<p之间。
- 再选择一个随机整数r，这是一个只有加密者知道的临时秘密数，对明文m的加密函数如下：
  - $C=E_{KU}(m)=(m\times y^r) \bmod p$
- 最后加密者把(C，g^r)发送出去。

四 公开密钥算法(ELGamal算法)

- 解密

- 接收者收到密文(C，g^r)之后，先取出g^r，再利用私钥x计算：

$$(g^r)^x=(g^x)^r=y^r \bmod p$$

- 接着，再用辗转相除法求出y^r 的乘法逆元$(y^r)^{-1}$；
- 最后，利用乘法逆元$(y^r)^{-1}$求出明文，解密函数如下：

$$\text{明文}=D_{KR}(C)=(y^r)^{-1}\times C \bmod p=(y^r)^{-1}\times m\times y^r \bmod p=m \bmod p$$

- 这样，接收者便利用私钥解出密文，得到原始信息m。

四 公开密钥算法(ELGamal算法)

- 例子：Bob想发送消息m=559给Alice。
- （1）Alice产生密钥
- Alice选择p=1009作为她的随机素数，再选择两个小于p的随机素数g=101和x=237，并计算：
  - $y=g^x \bmod p=101^{237} \bmod 1009=482$
- 于是，她发布她的公开密钥KU={y, g, p}={482, 101, 1009}，并保持私钥KR={r, p}={237, 1009}。

四 公开密钥算法(ELGamal算法)

- （2）Bob利用Alice的公钥加密消息m
- Bob选择随机的r=291，并计算：
  - $y^r=482^{291} \bmod 1009=378$
- 然后，Bob给Alice发送：
  - $C=E_{KU}(m)=(m\times y^r) \bmod p=559\times 378 \bmod 1009=421$
- 以及头部
  - $g^r=101^{291} \bmod 1009=661$

四 公开密钥算法(ELGamal算法)

- （3）Alice利用私钥解密消息，得到明文m
- Alice接到消息(421,661)后，由于她知道离散对数x，根据g^r=661她可能计算行到y^r。
  - $y^r=(g^x)^r=(g^r)^x=661^{237} \bmod 1009=378$
- Alice于是利用辗转相除法计算乘法逆元：
  - $(y^r)^{-1}=(378)^{-1} \bmod 1009=670$
- 最后，她计算明文：
- 明文$=(y^r)^{-1}\times C \bmod p=670\times 421 \bmod 1009=559$
- 这样，她就解出了原始消息“**559**”。

四 公开密钥算法(ELGamal算法)

- 这个密码算法的安全性取决于计算整数y基于g的离散对数x的难度，且模素数p，而且这个整数满足：
  - $g^x = y \mod p$