第3章 同余式

第2章引入了同余的概念，本章介绍在模 m 情况下同余式的求解。首先介绍一次同余式的求解，再介绍一次同余式组的求解。

本章的重点是中国剩余定理及其在RSA中的应用，难点是模重复平方法。

3.1 一次同余式

3.1.1 一次同余式的求解

定义 3.1 设 m 是一个正整数，$f(x)$为多项式且

$$f(x)=a_n x^n+\cdots+a_1 x+a_0$$

其中 a_i 是整数，则

$$f(x)\equiv 0 \pmod m$$

称作模 m 同余式。若 $a_n\not\equiv 0 \pmod m$，则 n 叫做 $f(x)$的次数，记为 degf。此时叫做模 m 的 n 次同余式。

如果整数 a 使得

$$f(a)\equiv 0 \pmod m$$

成立，则 a 叫做同余式的解。

事实上，满足 $x\equiv a \pmod m$的所有整数都使得该同余式成立。因此，在讨论同余方程的解时，以一个剩余类作为一个解。在模 m 的完全剩余系中，使得同余式成立的剩余的个数叫做同余式的解数。

下面首先考虑一次同余式的求解。

定理 3.1 一次同余方程 $ax\equiv b \pmod m$有解的充要条件是$(a,m)\mid b$，且当其有解时，其解数为(a,m)。

证明：先证明必要性。设同余方程 $ax\equiv b \pmod m$有解，解为 x_0，则存在整数 k，使得

$$ax_0\equiv km+b$$

即

$$b = ax_0-km$$

由于$(a,m)\mid a$，$(a,m)\mid m$，因此

$$(a,m)\mid ax_0-km=b$$

再证明充分性。

设 $a'=\frac{a}{(a,m)}, m'=\frac{m}{(a,m)}, b'=\frac{b}{(a,m)}$，易知，$a', m', b'$ 均为整数。

首先考虑同余方程

$$a'x \equiv 1 \pmod{m'}$$

因为 $\gcd(a', m')=1$，可知 a' 存在模 m' 的乘法逆元 x_0（见定义 2.3）。满足 $a'x_0 \equiv 1 \pmod{m'}$，且在模 m' 下，逆元是唯一的，即同余方程 $a'x \equiv 1 \pmod{m'}$ 存在唯一解

$$x \equiv x_0 \pmod{m'}$$

因此，易知同余方程

$$a'x \equiv b' \pmod{m'}$$

也存在唯一解

$$x \equiv x_1 \equiv x_0 b' \pmod{m'}$$

下面证明这个解也是 $ax \equiv b \pmod{m}$ 的特解：

不妨设 $x = k_1 m' + x_0 b'$，$k_1 \in z$

$$ax = ak_1 m' + ax_0 b' = ak_1 \frac{m}{(a,m)} + ax_0 \frac{b}{(a,m)} = a'k_1 m + a'x_0 b$$

$$\equiv a'x_0 b \pmod{m}$$

由于 $a'x_0 \equiv 1 \pmod{m'}$，不妨设 $a'x_0 = k_2 m' + 1$，$k_2 \in z$

$$a'x_0 b = (k_2 m' + 1)b = k_2 m' b + b = k_2 \frac{m}{(a,m)} b + b = k_2 m b' + b$$

$$\equiv b \pmod{m}$$

最后，$ax \equiv b \pmod{m}$ 的全部解为：

$$x \equiv x_1 + t\frac{m}{(a,m)} \pmod{m}, \quad t = 0, 1, \cdots, (a,m)-1$$

其原因是：

如果同时有同余式

$$ax \equiv b \pmod{m} \text{ 和 } ax_1 \equiv b \pmod{m}$$

成立，两式相减得到

$$a(x - x_1) \equiv 0 \pmod{m}$$

这等价于

$$x \equiv x_1 \left(\bmod \frac{m}{(a,m)}\right)$$

因此，x 只要与 x_1 关于 $\frac{m}{(a,m)}$ 同余，即为 $ax \equiv b \pmod{m}$ 的解。 ■

定理 3.1 的证明过程其实给出了一次同余式求解的过程。

定理 3.2 设 m 是一个正整数，a 是满足 $(a,m) \mid b$ 的整数，则一次同余式

$$ax \equiv b \pmod{m}$$

的全部解为

$$x \equiv \left(\left(\frac{a}{(a,m)}\right)^{-1}\left(\bmod \frac{m}{(a,m)}\right)\right)\frac{b}{(a,m)} + t\,\frac{m}{(a,m)} \pmod m$$

$$t = 0,1,\cdots,(a,m)-1$$

定理 3.2 其实给出了一次同余式“根与系数的关系”。定理 3.2 是定理 3.1 的充分性证明过程的结果，图 3.1 给出了解的 3 个部分与定理 3.1 的对应关系。为便于记忆，这里将这 3 个部分简称为：“模系收缩求逆元”、“逆元扩大求特解”、“t 变模扩求全解”。“模系收缩”的目的是使得“系数 a”和“模 m”之间互素，从而可以求“系数 a 的逆元”。有了“逆元”之后就可以扩大“$b' = \dfrac{b}{(a,m)}$”求出特解。最后再由特解求全解。

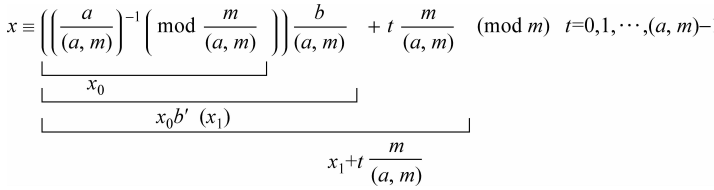

图 3.1　一次同余式“根与系数的关系”示意图

由定理 3.2 可以给出一个求解一次同余式的算法。

例 3.1　求解同余方程 $32x \equiv 12 \pmod 8$。

解答：$(32,8) \nmid 12$，由定理 3.1 知，同余方程无解。

例 3.2　求解同余方程 $6x \equiv 2 \pmod 8$。

解答：先判断 $(6,8) \mid 2$，所以同余方程有解。

先解同余方程 $3x \equiv 1 \pmod 4$，此方程解唯一，易知其解为

$$x \equiv 3 \pmod 4$$

取 $x_0 = 3$，则

$$x = x_0 + \frac{8}{2}t = 3 + 4t, \quad (t = 0,1)$$

为原方程的解。原方程的所有解为

$$x \equiv 3 \pmod 8$$

$$x \equiv 3+4 \equiv -1 \pmod 8$$

3.1.2　一次同余式在仿射加密中的应用

仿射密码是一种古典密码，其算法设计时用到的数学基础是模运算和同余方程。上一章曾经介绍过移位密码，由于移位密码的密钥量太小，且移位密码在加密代换后字母的先后次序其实没有改变。仿射密码可以改进上述两个弱点。

定义明文空间 $P = Z_{26}$、密文空间 $C = Z_{26}$、密钥空间为：

$$K = \{(a,b) \in Z_{26} \cdot Z_{26} : \gcd(a,26) = 1\}$$

对于

$$x \in P, y \in C, k = (a,b) \in K$$

定义加密函数：

$$e_k(x)=ax+b \bmod 26$$

解密函数：

$$d_k(y)=a^{-1}(y-b) \bmod 26$$

例 3.3 利用仿射加密，$a=3$，$b=5$，模为 26。

解答：易知，加密函数为 $e(x)=3x+5 \pmod{26}$

加密明文“hello”，其在 Z_{26} 表示为 7，4，11，11，14。

加密密文用 Z_{26} 表示为 0，17，12，12，21。即密文为“armmv”。

思考 3.1 仿射密码中 a 是否有要求。

根据定理 3.1，这里要求 $(a,26)=1$，否则，加密函数就不是一个单射函数。

例如：当 $k=(6,1)$ 时，$(a,26)=(6,26)=2$ 时，对 $x\in Z_{26}$，有

$$6(x+13)+1=6x+1 \bmod 26$$

于是 x，$x+13$ 都是 $6x+1$ 的明文。

思考 3.2 证明 $(a,26)=1$ 时，仿射密码的解唯一。

证明：设存在 $x_1,x_2\in Z_{26}$，使得 $e_k(x)=ax_1+b=ax_2+b \bmod 26$，于是 $ax_1=ax_2 \bmod 26$，有 $26\mid a(x_1-x_2)$，又因为 $(a,26)=1$，所以 $26\mid(x_1-x_2)$，由于 $x_1,x_2\in Z_{26}$，得到 $x_1=x_2$。 ■

思考 3.3 仿射密码的密钥空间大小，即密钥的数量有多少。

a 的可能性为 12，因为 $a\in Z_{26}$，$\gcd(a,26)=1$，即

$$a=\phi(26)=\phi(2\cdot 13)=\phi(2)\cdot\phi(13)=1\cdot 12=12$$

$b\in Z_{26}$，b 的可能性为 26。

故整个密钥空间大小为 $12\cdot 26=312$。

如果密钥空间太小，容易导致穷举所有可能的密钥，然后看能否解密密文的攻击。

另外，当 $a=1$ 时，仿射密码就退化为移位密码。

3.2 中国剩余定理

在研究了一次同余式之后，下面考虑一次同余式组。

中国剩余定理（Chinese Remainder Theorem，CRT），又称孙子定理，最早见于公元 5～6 世纪，我国南北朝的一部经典数学著作《孙子算经》中的“物不知数”问题：

“今有物不知其数，三三数之剩二，五五数之剩三，七七数之剩二，问物几何？”

这其实是求解一个一次同余方程式组，即

$$\begin{cases} x\equiv 2 \bmod 3 \\ x\equiv 3 \bmod 5 \\ x\equiv 2 \bmod 7 \end{cases}$$

《孙子算经》中给出了解法，但这只是一个孤立的例子。南宋数学家秦九韶创立的“大衍求一术”一般性地解决了一次同余方程组的求解问题，中国古代这一成果统称为“孙子定理”，在外国文献中常被称为“中国剩余定理”。它可能是最著名的由中国人给出的

算法。

在给出算法之前，请先体会一下如下同余式组：

$$\begin{cases} x \equiv 2 \bmod 3 \\ x \equiv 2 \bmod 5 \\ x \equiv 2 \bmod 7 \end{cases}$$

容易知道，这个问题容易解决，即 $x \equiv 2 \pmod{105}$，于是 $x = 3 \cdot 5 \cdot 7K + 2, K \in Z$。

问题是，如果余数不再相等，则问题就有点麻烦了，再来看同余式组：

$$\begin{cases} x \equiv 2 \bmod 3 \\ x \equiv 3 \bmod 5 \\ x \equiv 4 \bmod 7 \end{cases}$$

设法从 $3K_1+2, 5K_2+3, 7K_3+4 (K_1, K_2, K_3 \in Z)$ 中找到同一个数看上去是不容易的。也就是说，找到一个数同时满足这 3 个条件是不容易的。于是，能否换个思路，即能否把 x“想象”成由 3 个部分组成。为便于对 CRT 求解过程的理解，下面专门给出一个特别的表述和解释。

写成 3 个部分的好处是：每个部分的条件将“更容易”满足。

令 $x = A + B + C$，若 A, B, C 满足如下条件，则为解。这里 $[a]_m$ 表示模 m 与 a 同余。

$$A = [2]_3, B = [0]_3, C = [0]_3$$

$$A = [0]_5, B = [3]_5, C = [0]_5$$

$$A = [0]_7, B = [0]_7, C = [2]_7$$

看最左边一列，A 为 5 · 7 的倍数，且模 3 余 2，这样 A 就比较容易求解了。A 可用如下方法计算：

$A = (5 \cdot 7) \cdot (5 \cdot 7)^{-1} \cdot 2$。因为，这个表达式中有 $(5 \cdot 7)$，因此是 5 和 7 的倍数。同时这个表达式模 3 余 2。原因是：$(5 \cdot 7) = 35$。$(5 \cdot 7)^{-1}$ 表示 $(5 \cdot 7)$ 模 3 的逆元。35 模 3 的逆元是 2。两者相乘必然模 3 余 1。因此再乘以 2 后必然模 3 余 2。即 $A = 35 \cdot 2 \cdot 2 = 140$ 模 3 余 2，且为 5 和 7 的倍数。

同理，B 为 3 · 7 的倍数，且模 5 余 3，于是，可用如下方法计算：$B = (3 \cdot 7) \cdot (3 \cdot 7)^{-1} \cdot 3$，这里 $(3 \cdot 7)^{-1}$ 表示 $(3 \cdot 7)$ 模 5 的逆元。$3 \cdot 7 = 21$，21 模 5 的逆元是 1。因此，$B = 21 \cdot 1 \cdot 3 = 63$。

同理，C 为 3 · 5 的倍数，且模 7 余 2，于是，可用如下方法计算：$C = (3 \cdot 5) \cdot (3 \cdot 5)^{-1} \cdot 2$，这里 $(3 \cdot 5)^{-1}$ 表示 $(3 \cdot 5)$ 模 7 的逆元。$3 \cdot 5 = 15$，15 模 7 的逆元是 1。因此，$C = 15 \cdot 1 \cdot 2 = 30$。

于是 $x = A + B + C = 140 + 63 + 30 = 233$。又 $233 \bmod 105 = 23$。因此，所有解为 $105K + 23$。

在程大位著的《算法统要》(1593 年)，用四句诗给出了上述解答过程的中几个关键数字：

三人同行古来稀，五数梅花廿一枝；

七子团圆正月半，除百零五便得之。

这几个关键的数字是：与模 3 对应的是 70(“古来稀”)，即 35 · 2；与模 5 对应的是 21

("廿一枝"),即 $21\cdot 1$;与模 7 对应的是 $15\cdot 1=15$("正月半")。有了这三个关键数字,只要给出 x 模 3,5,7 的余数 a_1,a_2,a_3,即可以计算出结果 $x=a_1\cdot 70+a_2\cdot 21+a_3\cdot 15 \pmod{105}$。结果除去 $105=3\cdot 5\cdot 7$ 的倍数("除百零五")即可以得到答案。

一般地,如果 m_1,m_2,m_3 是两两互素的正整数,对于同余方程组

$$\begin{cases} x\equiv a_1 \bmod m_1 \\ x\equiv a_2 \bmod m_2 \\ x\equiv a_3 \bmod m_3 \end{cases}$$

令 $m=\prod_{i=1}^{3} m_i$, $M_i=m/m_i$,则解为

$$x\equiv \prod_{i=1}^{3} M_i\cdot M_i^{-1}\cdot a_i \pmod m$$

当然,上述解法可以推广到一般情况。

例 3.4 韩信点兵问题:有兵一队,若列成五行纵队,则末行一人,若列成六行纵队,则末行五人;若列成七行纵队,则末行四人;若列成十一行纵队,则末行十人。求兵数。

解答:转化为同余式组

$$\begin{cases} x\equiv 1 \bmod 5 \\ x\equiv 5 \bmod 6 \\ x\equiv 4 \bmod 7 \\ x\equiv 10 \bmod 11 \end{cases}$$

应用 CRT,$m=5\cdot 6\cdot 7\cdot 11=2310$

$$M_1=2310/5=462, M_1^{-1}=3 \pmod 5$$
$$M_2=2310/5=385, M_2^{-1}=1 \pmod 6$$
$$M_3=2310/5=330, M_3^{-1}=1 \pmod 7$$
$$M_4=2310/11=210, M_4^{-1}=1 \pmod{11}$$
$$\begin{aligned} x &\equiv 462\cdot 3\cdot 1+385\cdot 1\cdot 5+330\cdot 1\cdot 4+210\cdot 1\cdot 10 \\ &\equiv 6731\equiv 2111 \pmod{2310} \end{aligned}$$

下面给出 CRT 的严格证明。

定理 3.3(中国剩余定理) 设 $m_1,m_2,\cdots,m_k$ 是 k 个两两互素的正整数,令 $m=\prod_{i=1}^{k} m_i$, $M_i=m/m_i$,$(i=1,2,\cdots,k)$,则对任意的整数 $a_1,a_2,\cdots,a_k$,同余式组

$$\begin{cases} x\equiv a_1 \bmod m_1 \\ x\equiv a_2 \bmod m_2 \\ \cdots \\ x\equiv a_k \bmod m_k \end{cases} \tag{3.1}$$

有唯一解

$$x\equiv \prod_{i=1}^{k} M_i\cdot M_i^{-1}\cdot a_i \pmod m \tag{3.2}$$

其中 $M_iM_i^{-1}\equiv 1 \pmod{m_i}$

证明：由于$(m_i,m_j)=1,i\neq j$，即得$(M_i,m_i)=1$，由定义 2.3 和定理 2.8 知，对每一个 M_i，有一个 M_i^{-1} 存在，使得 $M_iM_i^{-1}\equiv 1 \pmod{m_i}$。另外，由于 $m=m_iM_i$，因此，$m_j \mid M_i, i\neq j$，故

$$\prod_{i=1}^{k} M_i \cdot M_i^{-1} \cdot a_i \equiv M_i \cdot M_i^{-1} \cdot a_i \equiv a_i \pmod{m_i} \quad i=1,2,\cdots,k$$

即(3.2)为(3.1)的解。

若 x_1,x_2 是满足(3.2)的任意两个整数，则 $x_1\equiv x_2 \pmod{m_i}(i=1,2,\cdots,k)$，因为$(m_i,m_j)=1,i\neq j$，于是 $x_1\equiv x_2 \pmod{m}$，故式(3.1)仅有解式(3.2)。 ■

根据 CRT，可以给出一个求解一次同余式组的算法。

3.3　同余式的应用

3.3.1　RSA 公钥密码系统

一个公钥密码系统的示意如图 3.2 所示，具体的描述如下。

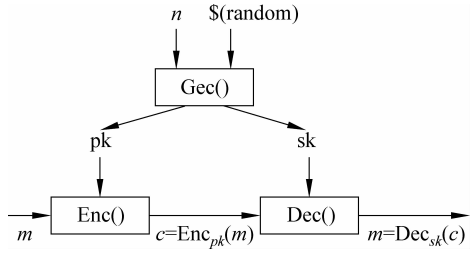

图 3.2　公钥加密体制的示意图

定义 3.2　一个公钥加密体制是这样的一个 6 元组$(M,C,K,\mathrm{Gen}(),\mathrm{Enc}_{\mathrm{pk}}(),\mathrm{Dec}_{\mathrm{sk}}())$，满足如下条件：

(1) M 是可能消息的集合。

(2) C 是可能密文的集合。

(3) 密钥空间 K 是一个可能密钥的有限集。

(4) 密钥生成算法 Gen()：输入安全参数，输出公钥 pk 和私钥 sk。

(5) 加密算法 $\mathrm{Enc}_{\mathrm{pk}}()$：根据输入的公钥 pk 和明文 m，输出密文 $c=\mathrm{Enc}_{\mathrm{pk}}(m)$。

(6) 解密算法 $\mathrm{Dec}_{\mathrm{sk}}()$：根据输入的私钥 sk 和密文 c，输出明文 $m=\mathrm{Dec}_{\mathrm{sk}}(c)$。

注解：

(1) 与对称加密体制的一个最大的不同是加密密钥和解密密钥是不同的，且加密密钥可以公开，解密密钥需要保密。

(2) 密钥生成算法在对称加密体制中是没有的(或者说是平凡的)，而在公钥密码体制中却是不平凡的。密钥生成算法可能是随机生成的密钥。

(3) 随机性。加密算法可能是随机的，即在算法中可以使用一个随机数，输出的密文

与该随机数有关，这种算法叫做概率加密。（例如 5.4 节的 ElGamal 加密。）

（4）确定性。解密算法一定是确定的。

（5）安全性（单向性）：在已知密文 c 和公钥 pk 的情况下，推出明文 m 在计算上是不可行的。对于任意的 $k\in K$，在已知 $\mathrm{Enc}_{pk}()$ 的情况下推出 $\mathrm{Dec}_{sk}()$ 是计算不可行的。对于任意的 $k\in K$，在已知 pk 的情况下，推出 sk 是计算不可行的。

（6）有效性（实用性）：公钥和私钥的产生，即密钥生成是容易的。即 Gen() 是多项式时间内可计算的。已知公钥 pk 和明文 m，计算密文 $c=\mathrm{Enc}_{pk}(m)$ 也是多项式时间可计算的。

（7）一致性。对每一个 $k=(pk,sk)\in K$，都对应一个加密算法 $\mathrm{Enc}_{pk}:M\to C$ 和解密算法 $\mathrm{Dec}_{sk}():C\to M$，满足对于任意的 $m\in M$，若 $c=\mathrm{Enc}_{pk}(m)$，则 $\mathrm{Dec}_{sk}(c)=m$。

（8）陷门性。若已知私钥 sk，存在多项式时间算法可以有密文 c 计算出明文 m，满足 $c=\mathrm{Enc}_{pk}(m)$，其中私钥 sk 称为陷门信息（trapdoor information）。

1977 年 MIT 的 Ronald L Rivest、Adi Shamir、Leonard Adleman 在 MIT 的技术报告中提出 RSA 方案，正式发表于 1978 年的 *Communication of ACM* 期刊上。①

RSA 利用了单向陷门函数的原理，其示意图如图 3.3 所示，陷门信息是解密密钥即私钥 d（与之类似的陷门是 n 的分解，因为如果知道 n 的分解 p 和 q，就可以从公钥求出私钥 d）。

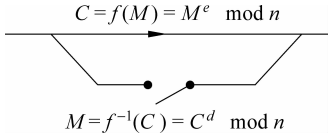

图 3.3　RSA 利用单向陷门函数的原理示意图

RSA 公钥密码方案描述如下：

1. 密钥生成

（1）选取两个大素数 p 和 q（例如长度都接近 512bit）；

（2）计算乘积 $n=p\cdot q$，$\Phi(n)=(p-1)(q-1)$，其中 $\Phi(n)$ 为 n 的欧拉函数；

（3）随机选择整数 $e(1<e<\phi(n))$，要求满足 $\gcd(e,\phi(n))=1$，即 e 与 $\phi(n)$ 互素。

（4）用扩展的 Euclidean 算法计算私钥 d，以满足 $d\cdot e\equiv 1 \mod (\phi(n))$，即 $d\equiv e^{-1} \mod (\phi(n))$。

公钥为 e 和 n，d 是私钥。（两个素数 p 和 q，可销毁，不能泄露。）

2. 加密过程

明文先转换为比特串分组，使每个分组对应的十进制数小于 n，即分组长度小于

① RSA 是第一个实用的公钥密码系统，是目前应用最广泛的公钥密码系统。后来，三位发明者在 2002 年获得了计算机领域的最高奖项——ACM 图灵奖。文献见 R. Rivest, A. Shamir, L. Adleman (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM 21 (2): 120-126。

$\log_2 n$，然后对每个明文分组 m_i 作加密运算，具体过程如下：

(1) 获得接收公钥(e,n)。

(2) 把消息 M 分组长度为 $L(L<\log_2 n)$的消息分组 $M=m_1 m_2 \cdots m_t$。

(3) 使用加密算法 $c_i=m_i^e \bmod n(1\leqslant i\leqslant t)$，计算出密文 $c=c_1 c_2 \cdots c_t$。

3. 解密过程

(1) 将密文 c 按长度 L 分组得 $c=c_1 c_2 \cdots c_t$。

(2) 使用私钥 d 和解密算法 $m_i=c_i^d \bmod n(1\leqslant I\leqslant t)$计算 m_i。

(3) 得明文消息 $M=m_1 m_2 \cdots m_t$。

解密算法的正确性证明如下：

$$c_i^d \bmod n \equiv m_i^{ed} \bmod n \equiv m_i^{k\phi(n)+1} \bmod n$$

分两种情况讨论：

(1) $\gcd(m_i,n)=1$。由欧拉定理，得

$$m_i^{\phi(n)} \equiv 1 \bmod n,\ m_i^{k\phi(n)} \equiv 1 \bmod n,\ m_i^{k\phi(n)+1} \equiv m_i \bmod n$$

于是 $c_i^d \bmod n \equiv m_i \bmod n$。

(2) $\gcd(m_i,n)\neq 1$。由于 $n=p\cdot q$，$\gcd(m_i,n)\mid n$，所以 $\gcd(m_i,n)=p$ 或 q。

不妨设 $\gcd(n,m_i)=p$，$p\mid m_i$，令 $m_i=sp$，$1\leqslant s\leqslant q$。

① $\gcd(m_i,q)=1$，由 Fermat 定理可得 $m_i^{q-1}\equiv 1 \bmod q$，于是$(m_i^{q-1})^{k(p-1)}\equiv 1 \bmod q$，即 $m_i^{k\phi(n)}\equiv 1 \bmod q$。

② 另外，由 $p\mid m_i$，得 $m_i^{ed}\equiv 0\equiv m_i \bmod p$，故 $m_i^{ed}\equiv m_i \bmod p$。

由①②，且 $\gcd(p,q)=1$。由中国剩余定理，$m_i^{ed}\equiv m_i \bmod n$，于是，$c_i^d \bmod n\equiv m_i \bmod n$。

综合(1)和(2)，有 $c_i^d \bmod n\equiv m_i \bmod n$。又 $m_i<n$，故 $m_i \bmod n=m_i$。 ■

例 3.5　取 $p=11$，$q=13$，那么 $n=pq=11\cdot 13=143$，$\phi(n)=(p-1)(q-1)=120$，选取 $e=17$，满足 $\gcd(e,\phi(n))=\gcd(17,120)=1$。

使用扩展的 Euclidean 算法计算 $d=e^{-1}=113 \bmod 120$，所以公钥为$(n,e)=(143,17)$，私钥为 $d=113$。

假设对明文 $m=24$ 进行加密，密文为 $c\equiv m^e\equiv 24^{17}\equiv 7 \bmod 143$。密文 $c=7$ 经公开信道发送到接收方后，接收方用私钥 $d=113$ 对密文解密：$m\equiv c^d\equiv 7^{113}\equiv 24 \bmod 143$。从而恢复明文。

3.3.2　CRT 在 RSA 中的应用

(1) 利用 CRT 进行运算加速。

解密者在计算 $m=c^d(\bmod n)$的过程中，可以分别计算 $m=c^d(\bmod p)$和 $m=c^d(\bmod q)$。然后利用 CRT 计算出 m。

例 3.6　计算 $312^{13}(\bmod 667)$

解答：令 $x=312^{13}$，解密者知道 $667=23\cdot 29$，所以计算等价于求解同余式组

$$\begin{cases} x\equiv a_1 \bmod 23 \\ x\equiv a_2 \bmod 29 \end{cases}$$

利用模重复平方方法(将在 3.3.3 节介绍),得到 $a_1=312^{13}\pmod{23}=8$,$a_2=312^{13}\pmod{29}=4$。再利用 CRT,有

$$M_1=29,M_1^{-1}=4\pmod{23},M_2=23,M_2^{-1}=-5\pmod{29}$$

$$x\equiv 4\cdot 29\cdot 8+(-5)\cdot 23\cdot 4=468\pmod{667}$$

思考 3.4:为什么 CRT 只能对解密加速,不能对加密加速。

解密需要知道 n 的分解,即需要知道 p,q,有 $n=pq$。只有解密者知道 p,q。

另外,因为解密可以加速,并且加密的速度对用户体验的影响可能更大,所以很多情形下 RSA 加密密钥 e 较解密密钥 d 小。

(2) 利用 CRT 进行低加密指数攻击。

如果加密密钥中加密指数 e 很小,例如 $e=3$,设明文为 m,密文 $c\equiv m^3 \bmod n$,如果 m 较小,则 c 有可能小于 n,则 mod n 操作未起作用,故可对 c 直接开 3 次方得到 m。如果 m 较大,mod n 操作起了作用,虽然不能直接开方,仍然有可能得到 m。假设有 3 个用户接收 m,模数分别为 n_1,n_2,n_3。设明文为 m,密文分别是:

$$c_1\equiv m^3 \bmod n_1$$

$$c_2\equiv m^3 \bmod n_2$$

$$c_3\equiv m^3 \bmod n_3$$

一般 $\gcd(n_i,n_j)\neq 1,i\neq j,i,j\in\{1,2,3\}$,否则可通过 $\gcd(n_i,n_j)$ 得到 n_i,n_j 的分解(从而泄露了 p,q),于是由中国剩余定理,可从三个密文同余式求出 $m^3 \bmod n_1n_2n_3$。由于此时 $0<m^3\leqslant n_1n_2n_3$,可直接对 m^3 开立方得到 m。

推而广之,若加密指数为 e,则得到相同明文的 e 个密文即可由该攻击方法恢复出明文。因此,同一消息加密后发送给多个实体时(若此时的加密指数相同),不要使用小的加密指数。

3.3.3 模重复平方算法

RSA 加密和解密时都需要计算模幂。平凡的求模幂方法在幂指数较大时耗时非常长。因此,寻求快速求模幂的算法对于 RSA 的加密解密效率至关重要。模重复平方(也称为平方乘)算法可以加快计算模幂的速度。下面介绍该算法。

要计算 $c=m^e \bmod n$,不妨设 e 的二进制表示为

$$\begin{aligned}e&=e_{k-1}2^{k-1}+e_{k-2}2^{k-2}+\cdots+e_12^1+e_0\\&=2(2(\cdots(2(2(e_{k-1})+e_{k-2})+)\cdots)+e_1)+e_0\end{aligned}$$

于是

$$\begin{aligned}c&\equiv m^e \bmod n\\&\equiv m^{e_{k-1}2^{k-1}+e_{k-2}2^{k-2}+\cdots+e_12^1+e_0} \bmod n\\&\equiv((\cdots\underbrace{((m^{e_{k-1}})^2}m^{e_{k-2}})^2\cdots m^{e_2})^2m^{e_1})^2m^{e_0} \bmod n\end{aligned}$$

从表达式可以看到,如果 $e_i=1$,则 m 即模将需要平方,并且反复进行,因而,称为“模重复平方法”。

根据这一表达式,可以设计计算模幂的快速算法。

算法 3.1：Square-and-Multiply(m,e,n)。

```
/*模重复平方(平方乘)算法,计算 c=m^e mod n。 */
/*输入:m,幂次 e,模 n                        */
/*输出:模幂的结果 c                         */
{
c=1
for i=k-1 to 0
{
     c = c^2 mod n
     If (e_i==1)   c ← c·m mod n
}
Return c
}
```

可以看到,在算法中 $i=k-1$ 时,即首次进入循环时,总满足条件 $e_{k-1}=1$,$c=1\cdot m \bmod n$。然后,每次进入循环后,先平方,从 e 的高位向低位考查,若该位为 1,则乘上 m,否则不乘。最后一次进入循环,$i=0$ 时,若 $e_0=1$,则乘上 m,否则不乘。

例 3.7　计算 $9726^{3533} \bmod 11413$。

解答：$3533=(110111001101)_2$,$m=9726$,表 3.1 给出了计算过程。

表 3.1　模重复平方法的计算过程

i	e_i	c
11	1	$1^2\cdot 9726=9726$
10	1	$9726^2\cdot 9726=2659$
9	0	$2659^2=5634$
8	1	$5634^2\cdot 9726=9167$
7	1	$9167^2\cdot 9726=4958$
6	1	$4958^2\cdot 9726=7783$
5	0	$7783^2=6298$
4	0	$6298^2=4629$
3	1	$4629^2\cdot 9726=10185$
2	1	$10185^2\cdot 9726=105$
1	0	$105^2=11025$
0	1	$11025^2\cdot 9726=5761$

思考 3.5：计算过程中有几次平方,几次乘法。

模平方运算有 12 次,模乘法运算有 8 次。平均而言,有 $\log_2 e$ 次模平方运算和约 $0.5\log_2$ 次,模平方和模乘法均视为模乘法运算,则总计算次数约为 $1.5\log_2 e$ 次模乘法,最

多不超过 $2\log_2 e$ 次模乘法运算。

算法的效率决定了 RSA 加密和解密是否实用。下面给出一个较严格的算法分析。首先来看几个基本模运算的效率，即在 Zn 中的运算，假定 n 为一个 l 比特的整数（$l=\log_2 n$），$0\leqslant m_1, m_2\leqslant n-1$。设 c 为一个正整数，结果如表 3.2 所示。

表 3.2　基本模运算的时间复杂度

运　　算	时间复杂度	运　　算	时间复杂度
$(m_1+m_2) \bmod n$	$O(l)$	$(m_1 m_2) \bmod n$	$O(l^2)$
$(m_1-m_2) \bmod n$	$O(l)$	$(m_1)^{-1} \bmod n$	$O(l^2)$

下面分析模重复平方算法（算法 3.2）的效率，有 k 次循环（即指数 e 的二进制位数），每次循环中总要执行 1 次平方运算（视为模乘），或者加上 1 次的模乘，故 1 次循环执行 2 次模乘或 1 次模乘。k 次循环最多 $2k$ 次模乘。根据表中所示模乘的时间复杂度为 $O(l^2)$，l 为模 n 的长度（即 $l=\log_2 n$），故总时间复杂度为 $O(kl^2)$。另外，通常 $e<n$，故 $k<1$，于是时间复杂度为 $O(l^3)=O((\log_2 n)^3)$。因此，RSA 的加密（或解密）都是在关于明文（或密文）的比特长度的多项式时间内完成。

另外，由于模重复平方算法的循环中模乘的次数等于加密密钥 e 的二进制表示中“1”的个数，故选择二进制表示中“1”较少的那种加密密钥 e 将会加快 RSA 加密的速度，例如 $e=2^{16}+1$，循环中只有 2 次模乘。

思　考　题

［1］计算 $2^{1000000} \pmod{1309}$（提示：CRT＋欧拉定理）。

［2］编写程序实现中国剩余定理（CRT），以韩信点兵为例（韩信带 1500 名兵士打仗，战死四五百人，站 3 人一排，多出 2 人；站 5 人一排，多出 4 人；站 7 人一排，多出 6 人。韩信马上说出人数：1049）。

［3］程序设计：模重复平方法计算 $x^n \pmod m$。计算 RSA，明文 $m=53$，公钥 $e=35$，$n=31\cdot 37$。

［4］计算 $2^{32} \pmod{47}$，$2^{43} \pmod{71}$，并用程序来验算。

［5］编写程序实现 300 十进制位 RSA 公钥密码系统。

第4章 二次同余式和平方剩余

第3章讨论了一次同余式，本章讨论二次同余式。

本章重点是平方剩余的判定，Legendre符号及其计算方法。难点是Rabin公钥密码系统。

4.1 二次同余式和平方剩余

二次同余式的一般形式是：

$$ax^2+bx+c \equiv 0 \pmod{m}$$

其中 $a \not\equiv 0 \pmod{m}$

因为正整数 m 有素因子分解式 $m=p_1^{\alpha_1}p_2^{\alpha_2}\cdots p_k^{\alpha_k}$，所以二次同余式等价于同余式组

$$\begin{cases} ax^2+bx+c \equiv 0 \pmod{p_1^{\alpha_1}} \\ \cdots \\ ax^2+bx+c \equiv 0 \pmod{p_k^{\alpha_k}} \end{cases}$$

因此，只需要讨论模为素数幂 p^α 的同余式

$$ax^2+bx+c \equiv 0 \pmod{p^\alpha}, \quad p \nmid a$$

通过配方，可以进一步变形为

$$(2ax+b)^2 \equiv b^2-4ac \pmod{p^\alpha}$$

令 $y=2ax+b$，有

$$y^2 \equiv b^2-4ac \pmod{p^\alpha}$$

因此，重点关注如下形式的二次同余式：

$$x^2 \equiv a \pmod{p^\alpha}$$

定义 4.1：设 m 为正整数，若同余式

$$x^2 \equiv a \pmod{m}, \quad (a,m)=1$$

有解，则称 a 为模 m 的平方剩余(quadratic residue，也叫做二次剩余)，否则称 a 为模 m 的平方非剩余(quadratic non-residue，二次非剩余)。

思考 4.1：对于 $m=2$，判断某个数是否为模2的平方剩余是平凡的。

下面主要考虑模为奇素数 p 的平方剩余。

例 4.1 求模7的平方剩余。

解答：通过穷举法，可计算出

$$1^1\equiv 1\pmod 7 \quad 2^2\equiv 4\pmod 7 \quad 3^2\equiv 2\pmod 7$$
$$4^2\equiv 2\pmod 7 \quad 5^2\equiv 4\pmod 7 \quad 6^2\equiv 1\pmod 7$$

于是,1,2,4 是模 7 的平方剩余,3,5,6 是模 7 的平方非剩余。

通过观察,发现在计算时可以“成对”计算,即只需要计算一半即$(p-1)/2$个数即可:

$$1^1\equiv 1\pmod 7 \qquad 2^2\equiv 4\pmod 7 \qquad 3^2\equiv 2\pmod 7$$
$$6^2\equiv(-1)^2\equiv 1\pmod 7 \quad 5^2\equiv(-2)^2\equiv 4\pmod 7 \quad 4^2\equiv(-3)^2\equiv 2\pmod 7$$

例 4.2 求模 17 的平方剩余。

解答:

x	1,16	2,15	3,14	4,13	5,12	6,11	7,10	8,9
$a\equiv x^2\pmod{17}$	1	4	9	16	8	2	15	13

模 17 的平方剩余为 1,2,4,8,9,13,15,16;平方非剩余为 3,5,6,7,10,11,12,14。

通过观察,可以发现,平方剩余的个数是简化剩余系元素个数的一半。

一般地,有如下结论:

定理 4.1 在素数模 p 的一个简化剩余系中,恰有$(p-1)/2$个模 p 的平方剩余,$(p-1)/2$个平方非剩余。

在给出证明之前,先看定理示意图(见图 4.1),左边为简化剩余系,右边为平方。可以看到,由于简化剩余系“成对”计算,计算出的平方剩余个数刚好为简化剩余系元素个数的一半。

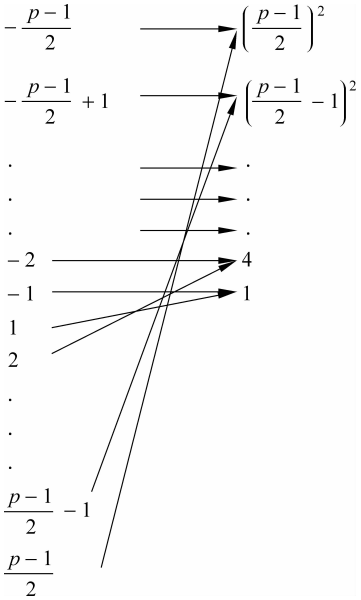

图 4.1 定理 4.1 的示意图

证明: 取模 p 的最小简化剩余系,这里在成对放置

$$-\frac{p-1}{2},-\frac{p-1}{2}+1,\cdots,-2,-1$$

$$\frac{p-1}{2},\frac{p-1}{2}-1,\cdots,+2,+1$$

则 a 是模 p 的平方剩余当且仅当

$$a\equiv\left(-\frac{p-1}{2}\right)^2,\left(-\frac{p-1}{2}+1\right)^2,\cdots,-2,-1,$$

$$\left(\frac{p-1}{2}\right)^2,\left(\frac{p-1}{2}-1\right)^2,\cdots,2,1,\ (\mathrm{mod}\ p)$$

由于$(-i)^2\equiv i^2(\mathrm{mod}\ p)$，所以 a 模 p 的平方剩余当且仅当

$$a\equiv 1^2,2^2,\cdots,\left(-\frac{p-1}{2}-1\right)^2,\left(-\frac{p-1}{2}\right)^2(\mathrm{mod}\ p)$$

又当 $i\neq j,1\leqslant i,j\leqslant(p-1)/2$ 时，$i^2\not\equiv j^2(\mathrm{mod}\ p)$，所以模 p 的平方剩余个数为$(p-1)/2$，模 p 的平方非剩余的个数为 $p-1-(p-1)/2=(p-1)/2$。■

根据定理 4.1，可以给出一个算法输出奇素数 p 的平方剩余和平方非剩余。

思考 4.2：如何根据定理 4.1 给出一个算法计算输出奇素数 p 的平方剩余和平方非剩余。

例 4.3　求方程 E：$y^2\equiv x^3+x+2\ (\mathrm{mod}\ 7)$ 的所有点。

解答：方程 E 其实就是一个椭圆曲线方程。由于模为 7，对 $x=0,1,2,3,4,5,6$，分别求 y。

$$x=0,y^2\equiv 2\ (\mathrm{mod}\ 7),\ y=3,4\ (\mathrm{mod}\ 7)$$
$$x=1,y^2\equiv 4\ (\mathrm{mod}\ 7),\ y=2,5\ (\mathrm{mod}\ 7)$$
$$x=2,y^2\equiv 5\ (\mathrm{mod}\ 7),\ \text{无解}$$
$$x=3,y^2\equiv 4\ (\mathrm{mod}\ 7),\ y=2,5\ (\mathrm{mod}\ 7)$$
$$x=4,y^2\equiv 0\ (\mathrm{mod}\ 7),\ y=0\ (\mathrm{mod}\ 7)$$
$$x=5,y^2\equiv 6\ (\mathrm{mod}\ 7),\ \text{无解}$$
$$x=6,y^2\equiv 0\ (\mathrm{mod}\ 7),\ y=0\ (\mathrm{mod}\ 7)$$

根据该结果，可以画出椭圆曲线图（见图 4.2），这个看上去像个“围棋盘”的图中，如果 $y\neq 0$，则 y 坐标是关于 7/2 对称的。（关于椭圆曲线密码的详述，请见第 9 章。）

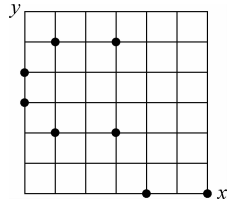

图 4.2　椭圆曲线方程 E：$y^2\equiv x^3+x+2\ (\mathrm{mod}\ 7)$上所有的点

前面（例 4.1 和例 4.2）对平方剩余的判定主要依靠穷举法，下面给出欧拉判定法则，从理论上给出了判别 a 是否为模 p 的平方剩余的方法。该方法将平方剩余的判定问题转化成计算问题。

定理 4.2(Euler 判定法则) 设 p 是奇素数,$(a,p)=1$,则

(1) a 是模 p 的平方剩余的充要条件是

$$a^{\frac{p-1}{2}} \equiv 1 \pmod{p}$$

(2) a 是模 p 的非平方剩余的充要条件是

$$a^{\frac{p-1}{2}} \equiv -1 \pmod{p}$$

当且仅当 a 是模 p 的平方剩余时,二次同余式

$$x^2 \equiv a \pmod{p},\ (a,p)=1$$

有 2 解。

证明:先证明(1)。先证明必要性。假定 $a \equiv y^2 \bmod p$,$a>0$,故 $y \neq 0 \bmod p$。于是根据 Fermat 定理,$(y^2)^{(p-1)/2} \equiv y^{p-1} \equiv 1 \bmod p$。

再证充分性。假定 $a^{(p-1)/2} \equiv 1 \bmod p$,设 b 为一个模 p 的原根(见第 5 章),于是 $a \equiv b^i \bmod p$,对于某个正整数 i 成立。有 $a^{(p-1)/2} \equiv b^{i(p-1)/2} \equiv b^{i(p-1)/2} \bmod p$,由于 b 的阶(见第 5 章)为 $p-1$,因此必有 $(p-1) \mid (i(p-1)/2)$。因此,i 是偶数,于是 a 的平方根为 $\pm b^{i/2} \bmod p$。

下面证明(2)。由于 $a^{p-1} \equiv 1 \pmod{p}$。$a^{\frac{p-1}{2}} \equiv 1 \pmod{p}$ 或者 $a^{\frac{p-1}{2}} \equiv -1 \pmod{p}$。由(1)可知(2)成立。 ■

例 4.4 8 是不是模 17 的平方剩余?

解答:$8^{(17-1)/2} \equiv 8^8 \equiv 1 \pmod{17}$,因此,8 是模 17 的平方剩余。

例 4.5 137 是不是模 227 的平方剩余?

解答:计算 $137^{(227-1)/2} \equiv 137^{113} \pmod{227}$,利用模重复平方方法,得到该值为 -1,因此,137 是模 227 平方非剩余。

推论:设 p 是奇素数,$(a_1,p)=1$,$(a_2,p)=1$,则

(1) 如果 a_1,a_2 都是模 p 的平方剩余,则 a_1a_2 是模 p 的平方剩余;

(2) 如果 a_1,a_2 都是模 p 的平方非剩余,则 a_1a_2 是模 p 的平方非剩余;

(3) 如果 a_1 是模 p 的平方剩余,a_2 是模 p 的平方非剩余,则 a_1a_2 是模 p 的平方非剩余。

显然,定理 4.2 可以转换成一个算法。虽然该算法比较简单,但是有利于加深对概念的理解。

算法 4.1 Euler 判定方法计算 a 是否为模 p 的平方剩余。

```
/* int Euler(a, p)                                  */
/* 输入:整数 a,(a,p)=1,奇素数 p                    */
/* 输出:a 是平方剩余或者 a 是平方非剩余             */
{
int ex = Square-and-Multiply(a,(p-1)/2,p); //调用模重复平方子函数
if (ex==1)
{
      printf("a 是平方剩余");
```

```
        Return 1;
    }
    else
    {
        printf("a 是平方非剩余");
        Return - 1;
    }
    }
```

4.2 Legendre 符号及其计算方法

当 p 不太大时，可以利用算法 4.1 判定某个数是否为平方剩余。但是，当 p 比较大时，该方法的计算量较大，就不实用了。下面引入勒让德符号，给出一种判定模 p 平方剩余的更有效的方法。

定义 4.2　设 p 是素数，a 是整数，Legendre(勒让德)符号定义如下：

$$\left(\frac{a}{p}\right)=\begin{cases}1, & \text{若 } a \text{ 是模 } p \text{ 的平方剩余}\\ -1, & \text{若 } a \text{ 是模 } p \text{ 的平方非剩余}\\ 0, & \text{若 } p \mid a\end{cases}$$

Legendre(勒让德)符号可理解成一个判定函数，函数的返回值为 1，−1，0。

算法 4.2 Legendre 函数计算 Legendre 符号的结果，这一计算过程直接从定义给出，是平凡的，主要目的是帮助初学者加深对 Legendre 符号这一概念的理解。

算法 4.2　Legendre 函数计算 Legendre 符号的结果。

```
/* int Legendre(int a, int p)          */
/* 输入：整数 a,素数 p                 */
/* 输出：Legendre 符号的值             */
int Legendre(int a, int p)
{
    if (p% a==0)                        //p|a
        Return 0;
    else
    {
        if (Euler(a,p)==1)  //调用 Euler 判定函数
            Return 1;
        else
            Return -1;
    }
}
```

其实，如果不要求定理 4.2 中的$(a,p)=1$，并且将 Legendre 符号引入，定理 4.2 可以叙述为定理 4.3。

定理 4.3 （欧拉判定法则）设 p 是奇素数，则对任意整数 a，

$$\left(\frac{a}{p}\right)=a^{\frac{p-1}{2}}(\bmod p)$$

定理 4.3 有两个好处：将 Legendre 符号的计算转变成模幂计算问题；同时这一计算结果可用来判断 a 是否为模 p 的平方剩余。

算法 4.3 计算 Legendre 符号（由定义给出，通过模幂计算的平凡方法）。

```
/* int L (int a, int p)          */
/* 输入：整数 a，奇素数 p        */
/* 输出：Legendre 符号的值       */
int L(int a, int p)
{
    Return Square-and-Multiply(a,(p-1)/2,p);  //调用模重复平方函数
}
```

下面讨论如何给出 Legendre 符号的快速计算方法。

由定理 4.3，可得到如下推论（这些推论有利于 Legendre 符号的快速计算）。

推论 1 设 p 是奇素数，则

(1) $\left(\frac{1}{p}\right)=1$。

(2) $\left(\frac{-1}{p}\right)=(-1)^{\frac{p-1}{2}}$。

由推论 1 可以给出如下计算子函数的定义，帮助初学者理解。

```
int L_ONE(const int a=1, int p)
{
    Retrun 1;
}
```

```
int L_MinusOne(const int a=-1, int p)
{
    Retrun (- 1)^((p-1)/2);
}
```

推论 2 设 p 是奇素数，那么

$$\left(\frac{-1}{p}\right)=\begin{cases}1, & \text{若 } p\equiv 1\ (\bmod 4)\\ -1, & \text{若 } p\equiv 3\ (\bmod 4)\end{cases}$$

证明：根据 Euler 判别法则，有

$$\left(\frac{-1}{p}\right)=(-1)^{\frac{p-1}{2}}$$

若 $p\equiv 1(\bmod 4)$，则存在正整数 k，使得 $p=4k+1$，于是$(p-1)/2=2k$。从而

$$\left(\frac{-1}{p}\right)=(-1)^{2k}=(-1)^{2k}=1$$

若 $p\equiv 1(\bmod 4)$，则存在正整数 k，使得 $p=4k+1$，于是$(p-1)/2=2k+1$。从而

$$\left(\frac{-1}{p}\right)=(-1)^{2k+1}=(-1)^{2k+1}=-1$$ ■

定理 4.4　设 p 是奇素数，则

(1) $\left(\frac{a}{p}\right)=\left(\frac{a+p}{p}\right)$。

(2) $\left(\frac{ab}{p}\right)=\left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$。

(3) 若$(a,p)=1$，则 $\left(\frac{a^2}{p}\right)=1$。

思考 4.3　读者能否给出一些例子说明上述定理。

定理 4.5　设 p 是奇素数，则

$$\left(\frac{2}{p}\right)=(-1)^{\frac{p^2-1}{8}}$$

例 4.6　证明 2 是模 17 的平方剩余。

证明：根据定理 4.5，

$$\left(\frac{2}{17}\right)=(-1)^{\frac{17^2-1}{8}}=(-1)^{2\cdot 18}=1$$

因此，2 是模 17 的平方剩余。 ■

定理 4.5 可以得到如下子函数(比较简单，仅为加深理解)。

```
int L_TWO(const int a=2, int p)
{
    Retrun (- 1)^((p^2-1)/8) ;
}
```

定理 4.6(二次互反律)　设 p,q 是互素的奇素数，则

$$\left(\frac{p}{q}\right)=(-1)^{\frac{p-1}{2}\frac{q-1}{2}}\left(\frac{q}{p}\right)$$

写成一个子函数，可以得到：

```
int L_MutualInverse(p,q)
{
    Retrun (- 1)^((p-1)/2 . (q-1)/2) L (q,p);
}
```

二次互反律提供了 Lengendre 符号的递归计算方式。

例 4.7 证明 3 是模 17 的平方非剩余。

证明：

$$\left(\frac{3}{17}\right)=(-1)^{\frac{3-1}{2}\cdot\frac{17-1}{2}}\left(\frac{17}{3}\right)=\left(\frac{-1}{3}\right)=(-1)^{\frac{3-1}{2}}=-1$$

因此，3 是模 17 的平方非剩余。 ■

例 4.8 判断同余式 $x^2\equiv 137 \pmod{227}$是否有解。

解答：因为 227 是奇素数，根据定理 4.4，

$$\left(\frac{137}{227}\right)=\left(\frac{-90}{227}\right)=\left(\frac{-1}{227}\right)\left(\frac{2\cdot 3^2\cdot 5}{227}\right)=-\left(\frac{2}{227}\right)\left(\frac{5}{227}\right)$$

$$\left(\frac{2}{227}\right)=(-1)^{\frac{227^2-1}{8}}=(-1)^{\frac{226\cdot 228}{8}}=-1$$

$$\left(\frac{5}{227}\right)=(-1)^{\frac{5-1}{2}\frac{227-1}{2}}\left(\frac{227}{5}\right)=\left(\frac{2}{5}\right)=(-1)^{\frac{5^2-1}{8}}=-1$$

因此，$\left(\frac{137}{227}\right)=-1$。

故同余式 $x^2\equiv 137 \pmod{227}$无解。

由上述的定理和推论，可以得到如下计算 Legendre 符号的算法 4.4。

算法 4.4 Legendre(勒让德)符号的计算方法。

/* int L_FAST(int a, int p)
/* 输入：奇素数 p≥3，整数 a，且 0≤a< p */
/* 输出：勒让德符号 $\left(\frac{a}{p}\right)$ */
1. 如果 a=0，则返回 0；
2. 如果 a=1，则返回 1；
3. 令 $a=2^e a1$，其中 a1 为奇数；
4. 如果 e 是偶数，则令 s=1；否则，如果 p=1 或 7 (mod 8)，则令 s←1；如果 p=3 或者 5 (mod 8)，则令 s←-1；
5. 如果 p=3 (mod 4)，a1=3(mod 4)，则令 s←-s；
6. 令 p1 ← p mod a1；
7. 如果 a1=1，则返回 s；否则返回 s·L_FAST(p1,a1)。

例 4.9 判断同余式

$$x^2\equiv -1 \pmod{365}$$

解答：365 可分解成 2 个素数的乘积，即 365=5·73，原同余式等价于

$$\begin{cases}x^2\equiv -1 \pmod 5\\ x^2\equiv -1 \pmod{73}\end{cases}$$

因为

$$\left(\frac{-1}{5}\right)=\left(\frac{-1}{73}\right)=1$$