21世纪高等学校计算机规划教材

现代密码学

Modern Cryptography

作　者：何大可 彭代渊 唐小虎 何明星 梅其祥

出版社：人民邮电出版社

现代密码学

Modern Cryptography

第4章 公钥密码体制

孙玉花

中国石油大学 理学院

Sunyuhua_1@163.com

2019年9月

第4章 公钥密码体制

- 4.0 公钥密码数学基础
- 4.1 公钥密码体制概述
- 4.2 RSA公钥密码体制
- 4.3 Rabin公钥密码体制
- 4.4 背包公钥密码体制
- 4.5 离散对数公钥密码体制

4.0 公钥密码数学基础

一、同余与模运算

二、快速模指数运算

三、模乘法逆元问题

四、著名的小费马定理和欧拉定理

五、本原根与离散对数

六、著名的中国剩余定理

七、二次剩余

八、单向函数

一、同余与模运算

- 两个整数a, b分别被m除，如果所得的余数相同，则称a与b对模m是同余的，记为$a \equiv b \pmod{m}$，正整数m称为模。求余运算称为模运算。
- $a \bmod m$将a映射到0到$m-1$之间。
- 给定整数m，将{0，1，…，m-1}记为Z_m。

一、同余与模运算

- 同余具有下面的性质：
  - 若$a \equiv b \pmod{m}$，则$m|(b-a)$。反过来，若$m|(b-a)$，则$a \equiv b \pmod{m}$
  - $a=km+b$ (k为整数)，等价于$a \equiv b \pmod{m}$
  - 每个整数恰与0,1,…，m-1这m个整数中的某一个对模m同余
  - 同余关系是一种等价关系
  - $a \equiv b \pmod{m}$当且仅当$a \bmod m = b \bmod m$。（由定义推出）

一、同余与模运算

- 给定整数m，将{0，1，…，m-1}记为Z_m，称为模m的非负最小完全剩余系。
- 将{0，1，…，m-1}中所有与m的公因子为1的整数取出，记为Z_m^*，称为模m的既约剩余系，且有$|Z_m^*| = \varphi(m)$。
- 若$ax \equiv b \pmod m$满足a，b为整数，m不是a的因子，则称该方程为模m的一次同余方程。若$\gcd(a, m)=1$,该方程有唯一解。

一、同余与模运算

- 模m求余运算称为模运算, 下面是模运算的一些性质。

- $(a+b) \bmod m = ((a \bmod m) + (b \bmod m)) \bmod m$
- $(a-b) \bmod m = ((a \bmod m) - (b \bmod m)) \bmod m$
- $(a \times b) \bmod m = ((a \bmod m) \times (b \bmod m)) \bmod m$
- $(a \times (b+c)) \bmod m = ((a \times b) \bmod m) + ((a \times c) \bmod m)) \bmod m$

一、同余与模运算

- 例如 **11 mod 8 = 3; 15 mod 8 =7,**

  那么 (11 mod 8)+ (15 mod 8) mod 8 = (3+7) mod 8 = 2

  (11+15) mod 8 = 26 mod 8 =2

- **在模运算中，加法单位元是0，(0+*a*) mod *m* = a mod *m***
- **乘法单位元是1，(1×*a*) mod *m* = *a* mod *m***

一、同余与模运算

- 对$a \in Z_m$，存在$b \in Z_m$，使得$a+b \equiv 0 \pmod m$，则b是a的加法逆元，
- 记$b = -a \bmod m$。
- 对$a \in Z_m$，存在$b \in Z_m$，使得$a \times b \equiv 1 \pmod m$，则称$b$为$a$的乘法逆元$a^{-1}$。
- 模m加法一定存在逆元，模m乘法不一定存在逆元。若$\gcd(a, m) = 1$，a的模m下的乘法逆元一定存在，反之不存在。

一、同余与模运算

- 定理 (乘法消去律)对于$ab \equiv ac$（$\mathrm{mod}\ m$）来说，若$\gcd(a, m)=1$则$b \equiv c(\mathrm{mod}\ m)$。
- 定理4(加法消去律)如果$a+b \equiv a+c$（$\mathrm{mod}\ m$），则$b \equiv c(\mathrm{mod}\ m)$
- 加法消去律是没有条件，但乘法消去律的条件是$\gcd(a, m)=1$，即a和m互素
- 例如 $6\times3\equiv6\times7\equiv2 \bmod 8$，但$3\equiv7 \bmod 8$不成立

模8运算的例子

- 模8的加法和乘法运算与普通运算一样,只是将所得的值取模8后的余数

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

一、同余与模运算

×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

模8的加法逆元和乘法逆元

- 对每一个x都有一个对应的y，使得$x+y \equiv 0 \bmod 8$，则y是x的加法逆元。如对2，有6，使得$2+6 \equiv 0 \bmod 8$，那么6是2的加法逆元
- 如果对x，存在y，使得$x \times y \equiv 1 \bmod 8$，则$y$为$x$的乘法逆元。如$3 \times 3 \equiv 1 \bmod 8$，因此3的乘法逆元是3。

a	−a	a^{-1}
0	0	—
1	7	1
2	6	—
3	5	3
4	4	—
5	3	5
6	2	—
7	1	7

二、快速模指数运算

快速的模幂运算

- 在非对称密码体制（公钥密码体制）中常常涉及指数模运算，如计算$73^{327} \bmod 37$
- 一种方法是利用前面介绍的模运算性质$(a \times b) \bmod m = ((a \bmod m) \times (b \bmod m)) \bmod m$，将指数模运算可以看做是多次重复乘法，并且在计算中间结果时就取模

二、快速模指数运算

- 例如：计算$11^7 \bmod 13$，可以按照下面的思路：

$11^2=121\equiv 4 \bmod 13$

$11^4=(11^2)^2\equiv 4^2 \bmod 13 \equiv 3 \bmod 13$

$11^7=11\times 11^2\times 11^4\equiv 11\times 4\times 3 \bmod 13 \equiv 132 \bmod 13 \equiv 2 \bmod 13$

二、快速模指数运算

快速求$m^e \bmod n$算法

$a \leftarrow e,\ b \leftarrow m,\ c \leftarrow 1$

while (a>0)

{ if (a是奇数)

$a \leftarrow (a-1),\ c \leftarrow (c \times b) \bmod n;$

else $a \leftarrow (a \div 2),\ b \leftarrow (b \times b) \bmod n;$

}

c为所求。

二、快速模指数运算

表 2.3　快速计算 $30^{37} \bmod 77$ 的过程

a	b	c
37	30	1
36	与前一次值相同	$(30\times1) \bmod 77 = 30$
18	$(30\times30) \bmod 77 = 53$	与前一次值相同
9	$(53\times53) \bmod 77 = 37$	与前一次值相同
8	与前一次值相同	$(37\times30) \bmod 77 = 32$
4	$(37\times37) \bmod 77 = 60$	与前一次值相同
2	$(60\times60) \bmod 77 = 58$	与前一次值相同
1	$(58\times58) \bmod 77 = 53$	与前一次值相同
0	与前一次值相同	$(53\times32) \bmod 77 = 2$

由最后一行可知，c=2，即 $30^{37} \bmod 77 = 2$。

二、快速模指数运算

快速求$m^e \bmod n$算法二（平方—乘算法）

$b \leftarrow m$，$e=a_k a_{k-1} \ldots a_1 a_0$（二进制展开）；$c \leftarrow 1$

for i=0 downto k do

{ if (i=0 and a_i=1) then c=m;

else{ $b \leftarrow (b \times b) \bmod n$;

If $a_i=1$ $c \leftarrow (c \times b) \bmod n$;

}

}

return c;

三、模乘法逆元

- 在密码学特别是非对称密码体制中，常常需要求模逆元，求模逆元就是求乘法逆元。即寻找一个x，使得$a \times x \equiv 1 \bmod m$成立
- 利用扩展欧几里德算法能够计算出模m下a的乘法逆元。

三、模乘法逆元

- 定理 （欧几里德算法）给定整数a和b，且$b>0$，重复使用带余除法，即每次的余数为除数去除上一次的除数，直到余数为0，这样可以得到下面一组方程：

  $a = bq_1+r_1,\ 0 < r_1 <b,\ \ b = r_1q_2+r_2,\ 0 < r_2 < r_1,$

  $r_1 = r_2q_3+r_3,\ 0 < r_3 < r_2,\ \ \ \ldots\ldots$

  $r_{j-1} = r_jq_{j+1}$，r_j就是a和b的最大公因子 。

由辗转相除定理还可得到下面结论：

对于不全为零的两个整数a,b存在整数s,t使得$\gcd(a,b)=sa+tb$

三、模乘法逆元

例1 求gcd (1970，1066)

- 用欧几里德算法的计算过程如下：
- **1970＝1×1066+904；　1066＝1×904+162**

  904＝5×162+94；　　162=1×94+68

  94＝1×68+26；　　68＝2×26+16

  26＝1×16+10；　　16＝1×10+6

  10＝1×6+4；　　6=1×4+2

  4＝2×2+0，因此gcd (1970，1066) = 2。

三、模乘法逆元

- 扩展欧几里德算法可以计算满足前面的组合系数*s*和*t*。
  - r1←a; r2←b;s1←1; s2←0; t1←0; t2←1;
  - while(r2>0)
  - { q=r1/r2;
  - r=r1-q×r2; r1← r2; r2←r;
  - s=s1-q×s2; s1← s2; s2←s;
  - t=t1-q×t2; t1← t2; t2←t;
  - }
  - gcd(a, b)←r1, s ←s1, t ←t1;
  - 满足gcd(a, b)=s×a+t×b

四、著名的小费马定理和欧拉定理

- 费马定理和欧拉定理在公钥密码体制中占非常重要的地位
- 定理2.5 (费马定理Fermat) 若p是素数, 且a是正整数，且 $\gcd(a, p) = 1$，则： $a^{p-1} \equiv 1 (\mathrm{mod}\ p)$
- 费马定理推论：若p是素数,对任意正整数a， $a^p \equiv a (\mathrm{mod}\ p)$，此时不要求$\gcd(a, p) = 1$。

四、著名的小费马定理和欧拉定理

- **设$\varphi(n)$为1到$n-1$之间且与n互素的正整数个数，则称其为欧拉(Euler)函数**

- **(欧拉定理)** 对于任何互素的两个整数a和n，有

$$a^{\varphi(n)} \equiv 1 \bmod n$$

欧拉定理推论：对任意正整数a，n是互不相同的素数之积，有

$$a^{\varphi(n)+1} \equiv a \bmod n ,$$

进而，对任意的正整数k, 有 $a^{k\varphi(n)+1} \equiv a \bmod n$ ，

此时不要求$\gcd(a, n) = 1$。

四、著名的小费马定理和欧拉定理

欧拉函数$\varphi(n)$的几条性质：

- n为素数， $\varphi(n)=n-1$;
- 若p为素数，n为正整数，则$\varphi(p^n)=(p-1)p^{n-1}$
- gcd(m, n) =1, $\varphi(mn)=\varphi(m)\times\varphi(n)$

五、本原根与离散对数问题

- 本原根
- 假设gcd (a, n) =1,
- 如果m是使$a^m \equiv 1 \bmod n$成立的最小正整数，则称它是a对模n的指数，或者称为a关于模n的乘法阶，记为$\mathrm{Ord}_n a$。

五、本原根与离散对数问题

- 若$\mathrm{Ord}_n a = \varphi(n)$，则称$a$是模$n$的本原根(primitive root)，也称模$n$的乘法生成元。
- 定理 当a是模n的本原根，则$1，a，\ldots，a^{\varphi(n)-1}$构成模$n$的既约剩余系, 也即：$Z_n^* = \{1，a，\ldots，a^{\varphi(n)-1}\}$。

五、本原根与离散对数问题

求模7和模15的本原根

- **对于模7而言，满足gcd (*a*, *n*) =1的*a*是{1,2,3,4,5,6}，将它们的指数列表如下**

a	1	2	3	4	5	6
$\mathrm{Ord}_7 a$	1	3	6	3	6	2

- 从上表可以看到，当*a*是3和5时，$\mathrm{Ord}_7 a = \varphi(7)$，因此，3和5是模7的本原根。

五、本原根与离散对数问题

对于模15而言，满足gcd $(a, n) = 1$的a是

{1,2,4,7,8,11,13,14},

将它们的指数列表如下：

a	1	2	4	7	8	11	13	14
$\mathrm{Ord}_7 a$	1	4	2	4	4	2	4	2

- **上表中不存在一个a，使$\mathbf{Ord}_{15} a = \varphi(15)$，所以模15没有本原根**
- **定理2.8 模m的本原根存在的必要条件是$m = 2, 4, p^a$, 或者$2\ p^a$，此处p是奇素数**

五、本原根与离散对数问题

本原根的测试

- 通常找出一个本原根不是一件容易的问题。
- 对一个素数p，如果知道$p-1$的因子，该问题就变得容易。
- 测试方法:令$q_1,q_2,\ldots,q_n$是$p-1$的素因子，对于所有的$q_1,q_2,\ldots,q_n$, 计算$a^{(p-1)/q} \pmod p$，如果对某个素因子q，其结果为1，那么a 不是一个本原根。如果对所有素因子q，其结果都不为1，那么a 是一个本原根。

五、本原根与离散对数问题

- 例 假设p=11, 检验2和3是否是一个本原根。

解： 当p=11时, p-1=10，p-1有两个素因子2和5，现测试2是否是一个本原根。

$2^{(11-1)/5} \pmod{11} = 4$； $2^{(11-1)/2} \pmod{11} = 10$

计算结果没有1，所以2是本原根。

测试3是否是本原根

$3^{(11-1)/5} \pmod{11} = 9$； $3^{(11-1)/2} \pmod{11} = 1$

所以3不是本原根。

五、本原根与离散对数问题

- 模运算用于指数计算可以表示为$a^x \bmod n$，我们称为模指数运算
- 模指数运算的逆问题就是找出一个数的离散对数，即求解x，使得 $a^x \equiv b \bmod n$
- 对于一个整数b和素数n的一个本原根a，可以找到唯一的指数x，使得$b \equiv a^x \bmod n$，其中$0 \le x \le n-1$，指数x称为b的以a为基数的模n的离散对数
- 离散对数是许多公钥算法的基础。

六、著名的中国剩余定理（孙子定理）

中国古代著名算题。原载《孙子算经》卷下第二十六题：“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?”当时虽已有了答案23，但它的系统解法是秦九韶在《数书九章·大衍求一术》中给出的。大衍求一术（也称作“中国剩余定理”）是中国古算中最有独创性的成就之一，属现代数论中的一次同余式组问题。

六、著名的中国剩余定理（孙子定理）

这个定理的最初形式是由一世纪的中国数学家孙子发现的。

(1) 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，

设$m = m_1 m_2 \cdots m_k$，$m = m_i M_i \ (i = 1, \cdots, k)$，则同余式

$$(x \bmod m_i) = a_i, \ i = 1, 2, \cdots k$$

有唯一解，

$$x \equiv M_1' M_1 a_1 + M_2' M_2 a_2 + \cdots + M_k' M_k a_k \pmod{m},$$

其中$M_i' M_i \equiv 1 \pmod{m_i} (i = 1, 2, \cdots, k)$，在此，$x$小于$m$，换句话说，一个数（小于一些素数之积）被它的余数模这些素数唯一确定。

(2) 中国剩余定理的一个推论可用于求出一个简化的问题的解：

如果 p 和 q 都是素数，p 小于 q，那么存在一个唯一的 x 小于 $p \times q$，使得 $x \equiv a \pmod{p}$ 且 $x \equiv b \pmod{q}$.

七、二次剩余

如果 p 是素数，且 a 小于 p，如果方程

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

有解，就称 a 是对模 p 的二次剩余。 当p时素数时，该方程至多有两个解。

例如 $p = 7$ 时，有

$1^2 = 1 \equiv 1 \pmod{7}$

$6^2 = 36 \equiv 1 \pmod{7}$

$2^2 = 4 \equiv 4 \pmod{7}$

$5^2 = 25 \equiv 4 \pmod{7}$

$3^2 = 9 \equiv 2 \pmod{7}$

$4^2 = 16 \equiv 2 \pmod{7}$

八、单向函数

- 单向和陷门单向函数是公钥密码学的核心，可以说公钥密码体制的设计就是陷门单向函数的设计
- 定义 (单向函数) 一个可逆函数$f: A \to B$，若它满足：

  (1) 对所有$x \in A$，易于计算$f(x)$。

  (2) 对几乎所有$x \in A$，由$f(x)$求x极为困难，以至于实际上不可能做到，则称f为单向函数(One-way Function)

八、单向函数

- 定义 (单向陷门函数)

一个“可逆”函数F若满足下列二条件，则称F为单向陷门函数 (One-way Trapdoor Function)：

- 对于所有属于F中定义域的任一x，容易计算$F(x)=y$；
- 对于几乎所有属于F中值域的任一y，除非获得陷门信息(trapdoor)，则求出x，使得 $x = F^{-1}(y)$在计算上不可行，F^{-1}为F的逆函数

- 单向函数是求逆困难的函数，而单向陷门函数是在不知陷门信息下求逆困难的函数，当知道陷门信息后，求逆是易于实现的。

八、单向函数

- 目前，还不能从理论上证明单向函数是存在的。
- 现实中却存在几个候选单向函数. 说他们是“候选”，是因为他们表现出了单向函数的性质，但还没有办法从理论上证明它们一定是单向函数
- 常见的候选单向函数：

  --因数分解问题

  --背包问题

  --离散对数

八、单向函数

单向函数举例

背包问题。已知向量

$$A=(a_1, a_2, \ldots, a_N),\quad a_i\text{为正整数},$$

称其为背包向量，称每个a_i为物品重量。给定向量

$$x=(x_1, x_2, \ldots, x_N),\quad x_i\in\{0, 1\},$$

求和式（称为背包重量）

$$S= a_1x_1+ a_2x_2+\ldots +a_Nx_N$$

容易，只需要不超过$N-1$次加法。但已知A和S，求x则非常困难，称其为背包问题，又称作子集和(Subset-Sum)问题。一般只能用穷举搜索法，有2^N种可能。N大时，相当困难。

八、单向函数

单向函数举例

背包问题的特例：超递增背包问题。将物品重量从小到大排列：a_1，a_2，a_3，…，a_N。称该背包问题为超递增背包问题，如果：

$a_1<a_2$；

$a_1+a_2<a_3$；

$a_1+a_2+a_3<a_4$；

…

$a_1+a_2+a_3+\ldots+a_{N-1}<a_N$。

（超递增背包问题是容易解决的。）

八、单向函数

单向函数举例

定理 设超递增背包重量为S。如果k满足$a_k<S<a_{k+1}$，则a_k是背包中的最大物品重量。

定理的证明

首先，背包中没有大于a_k的物品重量。

其次，背包中确有等于a_k的物品重量。

证明完毕。

注意到，寻找k满足$a_k<S<a_{k+1}$只需要对比N次。

八、单向函数

单向函数举例

超递增背包问题的解决方法

解决方法是可行的。设背包重量S，步骤如下。

（1）穷举：找k满足$a_k<S<a_{k+1}$。（这说明背包中的最大物品重量是a_k）

（2）记忆：存储这个k。

（3）卸载：如果$S>0$，则令$S:=S-a_k$，返回（1）。如果$w=0$，则到（4）。

（4）输出前面存储的所有的k，停止。

八、单向函数

单向函数举例

离散对数DL。给定一大素数p（比如，p在2^{1024}数量级），称$\log_2 p$为素数p的长度。

{1, 2, …, p-1}关于mod$p$乘法构成了一乘群Z_p^*，它是一个$p-1$阶循环群。该循环群的生成元一共有$\varphi(p-1)$个。

- 设一个生成元为整数g，$1<g<p-1$。
- 设一个整数x，$1<x<p-1$。
- 设y满足$y=g^x \bmod p$。

八、单向函数

单向函数举例

已知x，g，p， 求$y=g^x \bmod p$容易。

这是因为，采用折半相乘，只需要不超过$2\log_2 p$次的$\bmod p$乘法运算。

（实际上只需要不超过$2\log_2 x$次的$\bmod p$乘法运算。如

$$x=15=1111_2,$$

$$g^{15} \bmod p=(((g)^2 g)^2 g)^2 g \bmod p,$$

要用6次$\bmod p$乘法）

八、单向函数

单向函数举例

若已知y，g，p，求x满足$y=g^x \bmod p$，称为求解离散对数问题。记为$x=\log_g y \bmod p$。

求解离散对数问题的“最笨的方法”当然就是穷举，对每一个$x \in \{0, 1, 2, \ldots, p-1\}$检验是否$y=g^x \bmod p$。穷举求解法的运算次数约为$(p-1)/2$。许多求解离散对数问题的算法比穷举快得多，比如Shanks算法，Pohlig-Hellman算法等。