讲课内容

- 整除性和除法
- Euclid算法
- 模运算
- 群、环和域
- 有限域G(p)
- 多项式运算
- 有限域GF(2^n)

简介

- 有限域（finite fields）
- 在密码领域的重要性日益突出
  - AES, Elliptic Curve, IDEA, Public Key
- 对“数”的操作
- 概念：群(group)、环(ring)和域(field)

群

- **Group,**定义了二元运算的集合，记为{G, ·}
- 集合上的二元运算结果仍在该集合中(封闭性)
- 遵循:
  - 封闭性：a,b属于G，则a.b属于G
  - 结合律: $(a.b).c = a.(b.c)$
  - 单位元 e: $e.a = a.e = a$
  - 逆元 a^{-1}: $a.a^{-1} = e$
- 有限群、无限群
- 如果满足交换律 $a.b = b.a$

  则构成阿贝尔群(**Abelian group)**

循环群

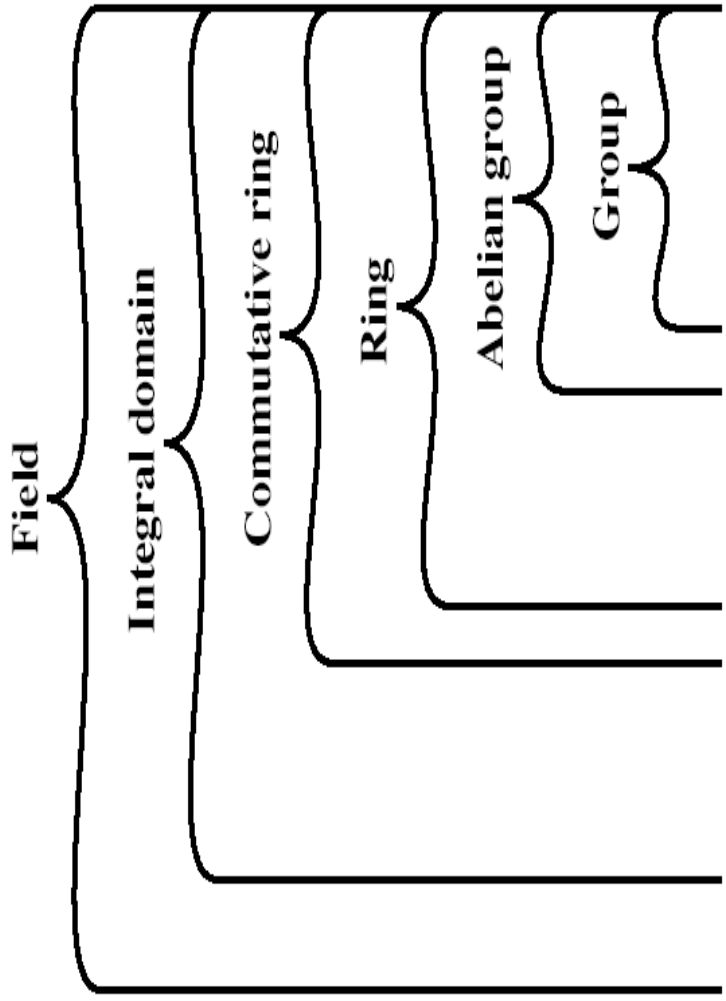
- 定义求幂运算为重复运用群中的运算
  - 如: $a^3 = a.a.a$
- 定义: $e=a^0$
- 称一个群为循环群，如果：群中每个元素都是一个固定元素a的幂，即
  $b = a^k$ （for some a and every b in group）
- a 称为群的一个生成元（generator）

环

- Ring，一个集合 ,记为{R, ＋，×}
- 定义了两种运算：加法和乘法
- 对加法，构成阿贝尔群
- 对乘法满足:
  - 封闭性
  - 结合律: a×(b×c) = (a×b)×c
  - 分配律: a×(b+c) = a×b + a×c
- 如果乘法满足交换律，则称交换环**commutative ring**
- 如过乘法有单位元且无零因子，则称整环**integral domain**

域

- Field， 集合 ， 记为 {F, ＋， ×}
- 两种运算:
  - 对加法，构成阿贝尔群
  - 对乘法，构成阿贝尔群（除0外）
  - 环
- 作加、减、乘和除法（除0外）运算，结果仍在集合中
- 继承关系：群 -> 环 -> 域 P69图4.1



(A1) Closure under addition:	If a and b belong to S, then $a + b$ is also in S
(A2) Associativity of addition:	$a + (b + c) = (a + b) + c$ for all a, b, c in S
(A3) Additive identity:	There is an element 0 in R such that $a + 0 = 0 + a = a$ for all a in S
(A4) Additive inverse:	For each a in S there is an element $-a$ in S such that $a + (-a) = (-a) + a = 0$
(A5) Commutativity of addition:	$a + b = b + a$ for all a, b in S
(M1) Closure under multiplication:	If a and b belong to S, then ab is also in S
(M2) Associativity of multiplication:	$a(bc) = (ab)c$ for all a, b, c in S
(M3) Distributive laws:	$a(b + c) = ab + ac$ for all a, b, c in S $(a + b)c = ac + bc$ for all a, b, c in S
(M4) Commutativity of multiplication:	$ab = ba$ for all a, b in S
(M5) Multiplicative identity:	There is an element 1 in S such that $a1 = 1a = a$ for all a in S
(M6) No zero divisors:	If a, b in S and $ab = 0$, then either $a = 0$ or $b = 0$
(M7) Multiplicative inverse:	If a belongs to S and $a \neq 0$, there is an element a^{-1} in S such that $aa^{-1} = a^{-1}a = 1$

模运算

- 定义：模运算 “a mod n” 表示用n去除a所得的余数

- 术语“同余”: a = b mod n
  - 用*n*去除a 和 b，他们有相同的余数
  - 如： 100 = 34 mod 11

- b 称作a模n的余数
  - 整数总可以写作: a = qn + b
  - 通常选择最小的非负整数作为余数，即
    0 <= b <= n-1

因子

- 整除：a=mb (a,b,m 都是整数)
- b是一个因子(没有余数)
- 记作： b|a
- b是a的一个因子
- 如：1,2,3,4,6,8,12,24 都是24 的因子

模算术运算

- 是一种“时钟运算”（clock arithmetic)

- 一些性质：
  - [(a mod n)＋(b mod n)] mod n = (a＋b) mod n
  - [(a mod n)－(b mod n)] mod n = (a－b) mod n
  - **[(a mod n)×(b mod n)] mod n = (a×b) mod n**

可为大整数求余提供便利

模算术运算

- 剩余集 $Z_n = \{0, 1, \ldots, n-1\}$
- 剩余类 [0]、[1]、…、[n-1]
- if $(a+b)=(a+c) \bmod n$

  then $b=c \bmod n$
- if $(a.b)=(a.c) \bmod n$

  then $b=c \bmod n$ only if $(a,n)=1$

模8加法

例：$Z_8=\{0,1,\ldots,7\}$ 上的模加法。

- **模加法＋：** 对每一a，都有一b，使得 a＋b≡0 mod 8

+	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

模8乘法

例：$Z_8=\{0,1,\ldots,7\}$ 上的模乘法。

- 模乘法×：并非每一a都有乘法逆元b, 使得a×b≡1 mod 8

×	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

1、3、5、7
有逆元

与**8**互素的有逆元？
YES!

模8的加法、乘法逆元

w	$-w$	w^{-1}
0	0	—
1	7	1
2	6	—
3	5	3
4	4	—
5	3	5
6	2	—
7	1	7

Table 4.2 Properties of Modular Arithmetic for Integers in Z_n

Property	Expression
Commutative laws 交换律	$(w + x) \bmod n = (x + w) \bmod n$ $(w \times x) \bmod n = (x \times w) \bmod n$
Associative laws 结合律	$[(w + x) + y] \bmod n = [w + (x + y)] \bmod n$ $[(w \times x) \times y] \bmod n = [w \times (x \times y)] \bmod n$
Distributive laws 分配律	$[w \times (x + y)] \bmod n = [(w \times x) + (w \times y)] \bmod n$ $[w + (x \times y)] \bmod n = [(w + x) \times (w + y)] \bmod n$
Identities 恒等式	$(0 + w) \bmod n = w \bmod n$ $(1 \times w) \bmod n = w \bmod n$
Additive inverse ($-w$) 加法逆元	For each $w \in Z_n$, there exists a z such that $w + z \equiv 0 \bmod n$

最大公因子(GCD)

- **GCD (a,b) ＝ GCD (|a|,|b|)**
- **如：GCD(60,24) = GCD(－60,24) = GCD(60,－24)＝12**
- **如果GCD(a,b) = 1，则称a和b互素**
  - **如： GCD(8,15) = 1**
- **注意： GCD(a,0) = |a|**

欧几里得算法
Euclidean Algorithm

- 求最大公因子的一个有效方法：欧几里得算法
- 算法基于:
  - GCD(a,b) = GCD(b, a mod b)
- 计算GCD(a,b)的欧几里得算法:

```
EUCLID(a,b)
1. A = a; B = b
2. if B = 0 return  A = gcd(a, b)
3. R = A mod B
4. A = B
5. B = R
6. goto 2
```

求GCD(1970,1066)

```
1970 = 1 x 1066 + 904   gcd(1066, 904)
1066 = 1 x 904 + 162    gcd(904, 162)
904 = 5 x 162 + 94      gcd(162, 94)
162 = 1 x 94 + 68       gcd(94, 68)
94 = 1 x 68 + 26        gcd(68, 26)
68 = 2 x 26 + 16        gcd(26, 16)
26 = 1 x 16 + 10        gcd(16, 10)
16 = 1 x 10 + 6         gcd(10, 6)
10 = 1 x 6 + 4          gcd(6, 4)
6 = 1 x 4 + 2           gcd(4, 2)
4 = 2 x 2 + 0           gcd(2, 0)
```

有限域

- 一个元素个数有限的域称为有限域或伽罗华域（Galois field）
- 有限域中元素的个数为一个素数或者一个素数的幂，记为GF(p)或GF(p^n)，其中p为素数。
- 有限域中运算满足交换律、结合律和分配律。
- 加法的单位元是0，乘法的单位元是1，每个非零元素都有一个唯一的乘法逆元。

- 密码学中用到很多有限域中的运算，因为可以保持数在有限的范围内，且不会有取整的误差。
- 常用的有限域：
  - GF(p)
  - GF(2^n)

伽罗华域GF(p)

- **GF(p)** 是整数集合Z_p={0,1, … , p-1}具有模素数p的代数运算
- Z_p中的整数模运算的性质（表4.2，P73）：交换律、结合律、分配律、恒等式、加法逆元
  - Z_n中的任一整数有乘法逆元，当且仅当该整数与n互素。
  - p为素数， Zn中所有的非零整数都与p互素，因此Zn中所有非零整数都有乘法逆元。
- 乘法逆元(w^{-1})： 任意w∈ Z_n中,w≠0,存在z ∈ Zn ,使得w×z ≡ 1 mod p，z就是w的乘法逆元w^{-1}

GF(7) 乘法

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

求逆算法

EXTENDED EUCLID(m, b)

1. (A1, A2, A3)=(1, 0, m);
   (B1, B2, B3)=(0, 1, b)

2. **if** B3 = 0
   return A3 = gcd(m, b); no inverse

3. **if** B3 = 1
   return B3 = gcd(m, b); B2 = b^{-1} mod m

4. Q = A3 div B3

5. (T1, T2, T3)=(A1 – Q B1, A2 – Q B2, A3 – Q B3)

6. (A1, A2, A3)=(B1, B2, B3)

7. (B1, B2, B3)=(T1, T2, T3)

8. **goto** 2

求GF(1759)中550的乘法逆元

Q	A1	A2	A3	B1	B2	B3
—	1	0	1759	0	1	550
3	0	1	550	1	–3	109
5	1	–3	109	–5	16	5
21	–5	16	5	106	–339	4
1	106	–339	4	–111	355	1

1 x 1759 + (-3) x 550 = 109

(-5) x 1759 + 16 x 550 = 5

106 x 1759 + (-339) x 550 = 4

(-111)x 1759 + 355 x 550 = 1

所以 **$550^{-1} = 355 \bmod 1759$**

扩展Euclid算法

- 扩展Euclid算法
- 做欧几里德算法的计算序列
- $r_0 = q_1 r_1 + r_2$ (令$r_0 = n$，$r_1 = a$)
- $r1 = q_2 r_2 + r_3$
- …
- $r_{m-2} = q_{m-1} r_{m-1} + r_m$ (1)
- $r_{m-1} = q_m r_m + r_{m+1}$ (0)
- 记 $t_0 = r_{m+1}$，$t_1 = r_m$，…
- 依 $t_j = t_{j-2} - q_{i-1}\, t_{j-1} \bmod r_0$，…
- 得 t_m
- 逆 r_1的逆

扩展Euclid算法举例

- **22×□≡ 1 mod 31**

31＝*1*×22＋9	*-1*×22 ≡ 9	即 30×22≡ 9 mod 31
22＝*2*×9＋4	22-*2*×(30×22) ≡ 4	即 3×22≡ 4 mod 31
9＝*2*×4＋1	30×22-*2*×(3×22) ≡ 1	即 24×22≡ 1 mod 31

- **28×□≡ 1 mod 75**

75＝*2*×28＋19	*-2*×28≡19	即 73×28≡19 mod 75
28＝*1*×19＋9	28-*1*×(73×28)≡9	即 3×28≡ 9 mod 75
19＝*2*×9＋1	(73×28)-*2*×(3×38)≡1	即67×28≡ 1 mod 75

- **269×□≡ 1 mod 349**

349＝*1*×269＋80	*-1*×269≡80	即 -1×269
269＝*3*×80＋29	269-*3*×(-1×269)≡29	即 4×269
80＝*2*×29＋22	(-1×269)-*2*×(4×269)≡22	即 -9×269
29＝*1*×22＋7	(4×269)-*1*×(-9×269)≡7	即 13×269
22＝*3*×7＋1	(-9×269)-*3*×(13×269)≡1	即-48×269 得 301

```
int module_reverse(int a, int m)
{
    int b, b1=1, b2=0;  // 逆元
    int r, r1=a, r2=m;  //
    do {
        r = r2 % r1;      //  y-kx = r
        b = (b2-(r2/r1)*b1)%m;
        r2 = r1; b2 = b1;
        r1 = r;           b1 = b;
    } while (r>1);

    if (r==0)  // r1中就是gcd,
        return 0;
    if (b<0)
        b += m;
    return b;
}
```

扩展的Euclid算法求乘法逆元

- 扩展的Euclid算法求乘法逆元：

如果gcd(a,n)=1，则a 在模n下有乘法逆元(不妨设a<n)，即存在x<n，使得

$$ax \equiv 1 \bmod n$$

即

$$x = a^{-1} \bmod n$$

~~~ ExtendedEuclid(a,n) ~~~

$g_{-1} = n, \quad u_{-1} = 1, \quad v_{-1} = 0$

$g_0 = a, \quad u_0 = 0, \quad v_0 = 1$

do

$q = \lfloor g_{i-2} / g_{i-1} \rfloor$

$g_i = g_{i-2} - q \cdot g_{i-1} = g_{i-2} \bmod g_{i-1}$

$u_i = u_{i-2} - q \cdot u_{i-1}$

$v_i = v_{i-2} - q \cdot v_{i-1}$

until $g_k = 0$

$\gcd(n, a) = g_{k-1}$

if $g_{k-1} = 1$, then $a^{-1} \bmod n = v_{k-1}$

扩展的Euclid算法求乘法逆元：举例

求 $15^{-1} \bmod 34$

$$q = \lfloor g_{i-2} / g_{i-1} \rfloor$$

$$g_i = g_{i-2} - q \cdot g_{i-1}$$

$$u_i = u_{i-2} - q \cdot u_{i-1}, \quad v_i = v_{i-2} - q \cdot v_{i-1}$$

循环次数	q	g_i	u_i	v_i	$g_i = u_i \times n + v_i \times a$
-1	-	**34**	1	0	34 = 1×34 + 0×15
0	-	**15**	0	1	15 = 0×34 + 1×15
1	2	4	1	-2	4 = 1×34 + (-2)×15
2	3	3	-3	7	3 = (-3)×34 +7×15
3	1	**1**	4	**-9**	1 = 4×34 + (-9)×15
4	3	0	-	-	

gcd(34,15) = 1

$15^{-1} \bmod 34 = -9 = 25$

多项式运算

- n次多项式

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 = \sum a_i x^i$$

  - a_i组成的集合称为系数集

- 讨论三种多项式运算
  - 使用代数基本规则的普通多项式运算
  - 系数运算是模**p**运算的多项式运算，即系数在$\mathbf{Z_p}$中
  - 系数在$\mathbf{Z_p}$中，且多项式被定义为模一个**n**次多项式**m(x)**的多项式运算

普通多项式运算

- 对应系数相加减（＋，－）
- 系数依次相乘（×）
- 如

  $f(x) = x^3 + x^2 + 2$，　$g(x) = x^2 - x + 1$

  $f(x) + g(x) = x^3 + 2x^2 - x + 3$

  $f(x) - g(x) = x^3 + x + 1$

  $f(x) \times g(x) = x^5 + 3x^2 - 2x + 2$
- 注意：定义在整数集上的多项式不支持除法运算，整数集不是域

系数在模Z_p中的多项式运算

- 系数是域F的元素时，构成多项式环（不构成整环，因为有可能有零因子）
- 系数是Z_p的元素的多项式
- 最感兴趣的是mod 2
  - 所有系数是0 或 1
  - 例如：$f(x) = x^3 + x^2$ 和 $g(x) = x^2 + x + 1$

    $f(x) + g(x) = x^3 + x + 1$

    $f(x) \times g(x) = x^5 + x^2$

多项式的因式

- 对于任何多项式:
  - $f(x) = q(x)\, g(x) + r(x)$
  - $r(x)$ 称为余式
  - $r(x) = f(x) \bmod g(x)$
- 若没有余式，则称$g(x)$整除$f(x)$
- 不可约（既约）多项式，也叫素多项式。
- 用一个不可约多项式作为模，则可构成一个域（可以定义除法了）

多项式的最大公因式

- *c(x)* = GCD(*a(x), b(x)*)，如果*c(x)* 是能够同时整除 *a(x), b(x)* 的多项式中次数最高的一个
- 欧几里得算法:

EUCLID[*a*(*x*), *b*(*x*)]

1. A(*x*) = *a*(*x*); B(*x*) = *b*(*x*)

2. if B(*x*) = 0 **return** A(*x*) = gcd[*a*(*x*), *b*(*x*)]

3. R(*x*) = A(*x*) mod B(*x*)

4. A(*x*) ¨ B(*x*)

5. B(*x*) ¨ R(*x*)

6. goto 2

多项式模运算

- 在GF(2^n) 中
  - 系数是 mod 2 的多项式
  - 次数低于n
  - 可用n次**素多项式去**模约以降次
- 构成一个有限域
- 非零元总有逆元
  - 可用扩展的欧几里得算法计算

Example GF(2^3)

Table 4.6 Polynomial Arithmetic Modulo ($x^3 + x + 1$)

	+	000 0	001 1	010 x	011 $x+1$	100 x^2	101 x^2+1	110 x^2+x	111 x^2+x+1
000	0	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
001	1	1	0	$x+1$	x	x^2+1	x^2	x^2+x+1	x^2+x
010	x	x	$x+1$	0	1	x^2+x	x^2+x+1	x^2	x^2+1
011	$x+1$	$x+1$	x	1	0	x^2+x+1	x^2+x	x^2+1	x^2
100	x^2	x^2	x^2+1	x^2+x	x^2+x+1	0	1	x	$x+1$
101	x^2+1	x^2+1	x^2	x^2+x+1	x^2+x	1	0	$x+1$	x
110	x^2+x	x^2+x	x^2+x+1	x^2	x^2+1	x	$x+1$	0	1
111	x^2+x+1	x^2+x+1	x^2+x	x^2+1	x^2	$x+1$	x	1	0

(a) Addition

	×	000 0	001 1	010 x	011 $x+1$	100 x^2	101 x^2+1	110 x^2+x	111 x^2+x+1
000	0	0	0	0	0	0	0	0	0
001	1	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
010	x	0	x	x^2	x^2+x	$x+1$	1	x^2+x+1	x^2+1
011	$x+1$	0	$x+1$	x^2+x	x^2+1	x^2+x+1	x^2	1	x
100	x^2	0	x^2	$x+1$	x^2+x+1	x^2+x	x	x^2+1	1
101	x^2+1	0	x^2+1	1	x^2	x	x^2+x+1	$x+1$	x^2+x
110	x^2+x	0	x^2+x	x^2+x+1	1	x^2+1	$x+1$	x	x^2
111	x^2+x+1	0	x^2+x+1	x^2+1	x	1	x^2+x	x^2	$x+1$

(b) Multiplication

计算上的考虑

- 因为系数是0或1, 所以可以用一个比特串来表示任何多项式
- 加法：比特串的逐位XOR
- 乘法：移位和XOR

例子：GF(2^3)

- (x^2+1) 表示为101_2 ，(x^2+x+1) 表示为111_2
- 加法
  - $(x^2+1) + (x^2+x+1) = x$
  - 101 XOR $111 = 010_2$
- 乘法
  - $(x+1).(x^2+1) = x.(x^2+1) + 1.(x^2+1)$
    $= x^3+x+x^2+1 = x^3+x^2+x+1$
  - $011.101 = (101)<<1$ XOR $(101)<<0 =$
    1010 XOR $101 = 1111_2$
- 多项式模运算
  - (x^3+x^2+x+1) mod $(x^3+x+1) = 1.(x^3+x+1) + (x^2) = x^2$
  - 1111 mod $1011 = 1111$ XOR $1011 = 0100_2$

生成元

- 有限域的一种定义
- 生成元g
  - 在域F中有0, g^0, g^1, …, g^{q-2}
- 用不可约多项式的根可以产生生成元
- 生成元的指数相加，就可定义乘法运算

本章小结

- 群、环和域的概念
- 整数的模运送
- 求最大公因数（式）的欧几里得算法
- 有限域 GF(p)
- 在GF(2^n) 中的多项式运算