信息安全基础综合设计实验

Lecture 03

李经纬

电子科技大学

课程回顾

内容回顾

- 计算模指数a^e mod m：**二进制分治算法**
  - 二进制分拆e为$\underline{d_{n-1}d_{n-2}\ldots d_0}$
  - 分治计算a^{D_i} mod m, $D_i = d_i * 2^i$
  - 归并
- 测试n是否为素数：**基于Eratosthenes筛选法**
  - 确定待筛选集合{2, 3,…, $n^{1/2}$}
  - 筛选{2, 3,…, $n^{1/2}$}内所有素数
  - 判断素数是否整除n

模指数运算

- 计算模指数a^e mod m
  - 思路：二进制拆分指数e为$d_{n-1}d_{n-2}...d_1d_0$，分治计算$a^{d_i * 2^i}$ mod m

```
int r = 1 // 为计算结果
while (e != 0){
    if (e % 2 == 1)
        r = r * a mod m // 即时归并
    a = a * a mod m // a^(2^(i+1)) mod m = (a^(2^i) mod m)^2 mod m
    e = e/2
}
return r
```

素性测试 I

- 确定待筛选集合set[0...m], $m = a^{1/2}-1$
  - 如果i在set中，则set[i] = 1，否则set[i] = 0
- Eratosthenes筛选：从2开始依次删除集合中剩余各数的整倍数

```
int set[0…m] = {1}; //筛选集合，1代表存在于集合中
for (int i=2; i < m; i++){
    j = i * i; //第一个可能删除的数是i^2
    while (j <= m) {
        set[j] = 0; //0代表从集合中删除
        j += i;
    }
}
```

- 筛选后元素关于n的整除性判断

数论基础实验——素性测试 II

素数的两个性质

- 性质 I：任意素数n可表示为**$n = 2^k q + 1$**，$k >= 0$，q为奇数
  - 特殊情况：$n = 2$时，$2 = 2^0 * 1 + 1$
- 性质 II：n是素数，a是小于n的正整数，则**$a^2 \bmod n = 1$当且仅当$a \bmod n = 1$或$a \bmod n = n - 1$**
  - 充分性：$a^2 \bmod n = (a \bmod n) * (a \bmod n) \bmod n = 1$
  - 必要性：$a^2 \bmod n = 1$，则$a^2 - 1 \bmod n = 0$；

    即$(a+1)(a-1) \bmod n = 0$；

    由于n为素数，因此$a+1 \bmod n = 0$或$a-1 \bmod n = 0$；

    即$a \bmod n = n-1$或$a \bmod n = 1$

Miller-Rabin算法原理

- $n = 2^k q + 1$（$k > 0$，q为奇数）是大于2的**素数**，a是大于1且小于n-1的整数，如下**两个条件之一**成立
  - $a^q \bmod n = 1$
  - 存在j（$j >= 1$且$j <= k$），满足$a^{2^{j-1}q} \bmod n = n-1$

Miller-Rabin算法原理

- $n = 2^k q + 1$（$k > 0$，q为奇数）是大于2的**素数**，a是大于1且小于n-1的整数，如下**两个条件之一**成立
  - $a^q \bmod n = 1$
  - 存在j（$j >= 1$且$j <= k$），满足$a^{2^{j-1}q} \bmod n = n-1$

- 费马小定理：a^{n-1} **mod** $n = a^{2^k q}$ **mod** $n = 1$
- 序列：**$a^q \bmod n, a^{2q} \bmod n, \ldots, a^{2^{k-1}q} \bmod n, a^{2^k q} \bmod n$**

Miller-Rabin算法原理

- $n = 2^k q + 1$（$k > 0$，q为奇数）是大于2的**素数**，a是大于1且小于n-1的整数，如下**两个条件之一**成立
  - $a^q \bmod n = 1$
  - 存在j（$j >= 1$且$j <= k$），满足$a^{2^{j-1}q} \bmod n = n-1$

> - 费马小定理：a^{n-1} **mod** $n = a^{2^k q}$ **mod** $n = 1$
> - 序列：$a^q \bmod n, a^{2q} \bmod n, \ldots, a^{2^{k-1}q} \bmod n, a^{2^k q} \bmod n$
>   - **后一项恰为前一项的平方**：$a^{2^i q}$ **mod** $n = [(a^{2^{i-1}q}$ **mod** $n)^2]$ **mod** n
>   - **最后一项为1**

Miller-Rabin算法原理

- $n = 2^k q + 1$（$k > 0$，q为奇数）是大于2的**素数**，a是大于1且小于n-1的整数，如下**两个条件之一**成立
  - $a^q \bmod n = 1$ **序列所有项均为1**
  - 存在j（$j >= 1$且$j <= k$），满足$a^{2^{j-1}q} \bmod n = n-1$ **序列存在一项为n-1，使之后所有项均为1**

- 费马小定理：a^{n-1} **mod** $n = a^{2^k q}$ **mod** $n = 1$
- 序列：a^q **mod n**, a^{2q} **mod n**,…, $a^{2^{k-1}q}$ **mod n**, $a^{2^k q}$ **mod n**
  - **后一项恰为前一项的平方**：$a^{2^i q}$ **mod** $n = [(a^{2^{i-1}q}$ **mod** $n)^2]$ **mod** n
  - **最后一项为1**

Miller-Rabin素性测试算法

- **Miller-Rabin**(n)
  - 确定整数k和q，满足**$n = 2^k q + 1$**
  - **随机**选择整数a，满足a > 1且a < n-1
  - 如果**$a^q \bmod n = 1$**，返回**“不确定”**(可能是素数)
  - 如果存在**$a^{2^{j-1}q} \bmod n = n-1$** (j = 1, 2,..., k)，返回**“不确定”**
  - 返回“合数”
- 通过Miller-Rabin素性测试的数**不一定**是素数；无法通过Miller-Rabin素性测试的数一定不是素数（必要条件）

Miller-Rabin素性测试示例 I

- 判断29是否是素数
  - 确定k和q：$29 = 2^2 * 7 + 1$，因此，$k = 2, q = 7$
  - 随机选择a：$a = 2$
  - 序列：$2^7 \bmod n, 2^{2*7} \bmod n, 2^{4*7} \bmod n$
  - 判定I：$a^q \bmod n = 12$，既不等于n-1，又不等于1
  - 判定II：序列第二项$a^{2q} \bmod n = 28 = n-1$，返回“不确定”

Miller-Rabin素性测试示例 II

- 判断221是否是素数
  - 确定k和q：$221 = 2^2 * 55 + 1$，因此，$k = 2$，$q = 55$
  - 随机选择a：$a = 5$
  - 判定I：$a^q \bmod n = 5^{55} \bmod 221 = 112$，既不等于n-1，又不等于1
  - 判定II：$5^{2q} \bmod n = (5^{55})^2 \bmod 221 = 168$，返回“合数”

Miller-Rabin素性测试示例 II

- 判断221是否是素数
  - 确定k和q：$221 = 2^2 * 55 + 1$，因此，$k = 2$，$q = 55$
  - 随机选择a：$a = 5$
  - 判定I：$a^q \bmod n = 5^{55} \bmod 221 = 112$，既不等于n-1，又不等于1
  - 判定II：$5^{2q} \bmod n = (5^{55})^2 \bmod 221 = 168$，返回“合数”
- 随机选择a = 21？

Miller-Rabin素性测试示例 II

- 判断221是否是素数
  - 确定k和q：$221 = 2^2 * 55 + 1$，因此，$k = 2$，$q = 55$
  - 随机选择a：$a = 5$
  - 判定I：$a^q \bmod n = 5^{55} \bmod 221 = 112$，既不等于n-1，又不等于1
  - 判定II：$5^{2q} \bmod n = (5^{55})^2 \bmod 221 = 168$，返回“合数”
- 随机选择a = 21？
  - $21^{55} \bmod 221 = 200$；$(21^{55})^2 \bmod 221 = 220$，”不确定?”

Miller-Rabin素性测试示例 II

- 判断221是否是素数
  - 确定k和q：$221 = 2^2 * 55 + 1$，因此，$k = 2$，$q = 55$
  - 随机选择a：$a = 5$
  - 判定I：$a^q \bmod n = 5^{55} \bmod 221 = 112$，既不等于n-1，又不等于1
  - 判定II：$5^{2q} \bmod n = (5^{55})^2 \bmod 221 = 168$，返回“合数”
- 随机选择a = 21？
  - $21^{55} \bmod 221 = 200$；$(21^{55})2 \bmod 221 = 220$，”不确定?”

非确定测试：选择不同的a，测试结果不完全相同（**多次测试**）

数论基础实验——乘法逆元

乘法逆元

- 定义：对于整数a和m，如果存在整数b，满足a * b mod m = 1，则称b为a关于模m的乘法逆元，记为a^{-1}
- 乘法逆元目标：**已知a和m，求解a^{-1}**
- 用途：现代密码学加解密常涉及求解乘法逆元

存在条件

- a存在关于模m的乘法逆元的充要条件是，a和m的最大公约数为1（或a和m互素），记为gcd(a, m) = 1

- a与m互素，则存在整数k_1, k_2，满足k_1*a + k_2*m = 1
  - **等式两边同取mod m**： k_1*a = 1 mod m，即k_1是a关于模m的乘法逆元

存在条件示例

- 分析6和5关于模8的乘法逆元
  - gcd(6, 8) = 2, gcd(5, 8) = 1

Z_8	0	1	2	3	4	5	6	7
乘以6	0	6	4	2	0	6	4	2
乘以5	0	5	2	7	4	1	6	3

6不存在关于模8的乘法逆元

5关于模8的乘法逆元为5（本身）：5*5 + (-3)*8 = 1

存在条件示例

➢分析6和5关于模8的乘法逆元

- gcd(6, 8) = 2, gcd(5, 8) = 1

Z_8	0	1	2	3	4	5	6	7
乘以6	0	6	4	2	0	6	4	2
乘以5	0	5	2	7	4	1	6	3

6不存在关于模8的乘法逆元

5关于模8的乘法逆元为5（本身）：5*5 + (-3)*8 = 1

➢**如何求解乘法逆元？**

- k_1*a + k_2*m = 1，求解k_1

欧几里德算法原理

- 最大公约数：gcd(a, m)是a和m的因子，并且a和m的任意因子都是gcd(a, m)的因子

- 欧几里德算法：辗转相除，求最大公约数

$a = q_1 * m + r_1$, $r_1 >= 0$且$r_1 < m$

如果$r_1 = 0$, gcd(a, m) = m；否则, **gcd(a, m) = gcd(m, r_1)**

$m = q_2 * r_1 + r_2$, $r_2 >= 0$且$r_2 < r_1$，意味着**gcd(m, r_1) = gcd(r_1, r_2)**

……辗转相除

$r_{n-1} = q_{n+1} * r_n + 0$, 那么**gcd(a, m) = gcd(m, r_1) = gcd(r_1, r_2) =…= r_n**

欧几里德算法示例 I

- 求7与96的最大公约数
  - 假设$r_{-1} = 7, r_0 = 96$

i	r_i	q_i	**公式**
-1	7	0	7 = 0*96 + 7
0	96	13	96 = 13*7 + 5
1	7	1	7 = 1*5 + 2
2	5	2	5 = 2*2 + 1
3	2	2	2 = 2*1
4	1		

$r_4 = 1$, 因此gcd(7, 96) = 1

欧几里德算法示例 II

- 求270与96的最大公约数
  - 假设r_{-1} = 270, r_0 = 96

i	r_i	q_i	**公式**
-1	270	2	270 = 2*96 + 78
0	96	1	96 = 1*78 + 18
1	78	4	78 = 4*18 + 6
2	18	3	18 = 3*6

gcd(270, 96) = 6

扩展欧几里德算法

- 在欧几里德算法的基础上，计算辗转相除的系数

$a = q_1*m + r_1$		$1*a + (-q_1)*m = r_1$
$m = q_2*r_1 + r_2$	$m + (-q_2)*r_1 = r_2$	$(-q_2)*a + (1+q_1q_2)*m = r_2$
$r_1 = q_3*r_2 + r_3$	$r_1 + (-q_3)*r_2 = r_3$	$(1+q_2q_3)*a + (1-q_1-q_1q_2q_3)*m = r_3$
…	…	…
$r_{n-1} = q_{n+1}*r_n$	…	$k_1*a + k_2*m = r_n = 1$

计算系数k_1, k_2，k_1为a关于模m的乘法逆元

$$\boxed{r_{i-1} + (-q_{i+1})*r_i = r_{i+1}}$$

欧几里德扩展算法示例 I

- 求7关于模96的逆元: 假设$r_{-1} = 7, r_0 = 96$

i	r_i	q_i	**公式**		
-1	7	0	7 = 0*96 + 7		**1***7 + **0***96 = 7
0	96	13	96 = 13*7 + 5	1*96 + (-13)*7 = 5	**(-13)***7 + **1***96 = 5
1	7	1	7 = 1*5 + 2	7 + (-1)*5 = 2	**(14)***7 + **(-1)***96 = 2
2	5	2	5 = 2*2 + 1	5 + (-2)*2 = 1	**(-41)***7 + **3***96 = 1
3	2	2	2 = 2*1		
4	1				

$k_1 = -41, k_2 = 3$

可选：将k_1变换为Z_{96}中的元素55

欧几里德扩展算法示例 II

- 求270关于模96的乘法逆元: 假设r_{-1} = 7, r_0 = 96

i	r_i	q_i	公式		
-1	270	0	270 = 2*96 + 78		1*270 + (-2)*96 = 78
0	96	1	96 = 1*78 + 18	1*96 + (-1)*78 = 18	(-1)*270 + 3*96 = 18
1	78	4	78 = 4*18 + 6	78 + (-4)*18 = 6	5*270 + (-14)*96 = 6
2	18	3	18 = 3*6	18 + (-3)*6 = 0	

由于gcd(270, 96) != 1，不存在乘法逆元

k_1 = 5, k_2 = -14

课堂作业

素性测试 II

- 基于Miller-Rabin算法进行素性判定
- 函数头：

```
std::string miller_rabin_prime_test(\
                    unsigned int n, \ // 被测试数
                    unsigned int a)   // 测试随机值
// 返回: "not_prime" - 表示一定不是素数
//        "uncertain" - 表示不一定是素数，即无法确定

std::string miller_rabin_multiple_test(\
                    unsigned int n, \ // 被测试数
                    unsigned int repeat_times) // 测试轮数
// 每一轮随机选择a进行测试，存在某一轮无法通过，返回"not_prime";
//                        repeat_times轮均能够通过测试，返回“uncertain”
```

乘法逆元

- 基于扩展欧几里德算法计算乘法逆元
- 函数头：

```
int euclid_mod_reverse(int a, int m)
// 参数：
//      a - 需要求逆元的数
//      m - 模
// 返回值：std::int
//      返回值说明：返回a关于m乘法逆元；不存在返回-1
```