Crypto_P3

P3 调整说明

为了准备 mst，最近课程优先讲解往年 mst 考察内容。

按照经验，往年 mst 重点：

- 概念等基础知识
- 分组加密模式
- 扩展欧几里得
- RSA

数学基础

- 整数与可除性
- GCD
- EGCD
- 欧拉定理
- 费马小定理

整除相关计算

`a|b` 读作a整除b，这时 b = k*a，那么有以下结论

- 如果`a|1`那么 a = 1 或 a = -1
- 任意a，满足`a|0`
- 如果`a|b`,`b|c` 那么 `a|c`
- 如果`a|b`,`a|c` 那么 `a|(sb+tc)`

$$ a\ \equiv \ b \ (mod\ c) $$

- n|(a-b) -> a=b mod n
    - b = k1*n + c a = (k1+(a-b)/n)*n + c
- a = b mod n, b=c mod n -> a=c mod n
- a+b mod n = ((a mod n) + (b mod n)) mod n

上面这个性质对加、减、乘均成立，但是：

```
18 = 48 mod 10
3 != 8  mod 10
```

对于除法：

```
ac = bc mod m
a = b mod (m/gcd(c,m))
```

例题：计算 3 ^ 123456789 mod 80

```
3^4 = 81
3^4 = 1 (mod 80)
3^4 * 3^123456785 (mod 80)
1 * 3^123456785 (mod 80)
3^(123456789 % 4) (mod 80)
---
关注点转移到：123456789 % 4
1234567*100 + 89 % 4
20 * 4 + 9 % 4
2*4 + 1 % 4
---
3 ^ 1 % 80 = 3
```

算数基本定理

- 任意正整数可以表示成若干个 1 的加法，因此 1 是加法的基本单位。
- 素数则是乘法的基本单位，任意正整数可以分解成若干素数的乘积。

```
m = p1^a1 * p2^a2 ... pn^an
```

任意正整数都可以得到这样的分解，并且已知正整数的分解后，恢复这个整数是容易的。 但是如果只知道整数，要得到它的分解，这是一个数学上的困难问题。

```python
def fac(n):
    result = []
    while n != 1:
        for i in range(2, n + 1):
            if n % i == 0:
                result.append(i)
                n //= i
                break
    return result
```

欧拉定理

如果 `gcd(a, n) == 1`:

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

其中 φ 称为欧拉函数。

欧拉函数

- 定义欧拉函数 phi(n) 为小于n且和n互素的整数数量
- 7：1 2 3 4 5 6 phi(7) = 6
- 6: 1 5 phi(6) = 2
- 如果p是素数， phi(p) = p-1
- 如果n = p1*p2，phi(n) = (p1-1)(p2-1)
- 如果n = p1*p2...，phi(n) = (p1-1)....
- 如果存在相同的，n = (p1^i)*p1, f(n) = (p1^i)*(p1-1)

也可以直接使用下面的计算方式：

$$ \varphi(m) \equiv m\prod_{i=1}^n (1-\frac{1}{p_i}) $$ 其中p为m的分解：m = p1^a1 p2^a2 ... pn^an

例题：n = 72283 求 phi(n) ?

```
显然：72283 = 41*41*43
phi(72283) = (41-1)*41*(43-1) = 68880
```

证明

如果 a 和 p1 ... pn 均和 m 互素，那么 ap1 ... apn 均和 m 互素。 如果 p1 ... pn 恰好覆盖了 0-m 之间所有和 m 互素的数，则ap1 ... apn 对m取余后同样完成覆盖。 反证法，如果 api1 = api2 mod m，那么pi1 = pi2 mod m，这与前提矛盾。

```
api1 = k1m+b
api2 = k2m+b
a(pi1 - pi2) = (k2-k1)m
```

那么

$$ \prod_{i=1}^npi \equiv \prod_{i=1}^n a\ p_i (mod \ m) $$

$$ \prod_{i=1}^np_i \equiv a^{\varphi(n)} \prod_{i=1}^n p_i (mod \ m) $$

$$ a^{\varphi(m)} \equiv 1 (mod \ m) $$

费马小定理

- 费马小定理为欧拉定理特殊形式
  - 若m为素数，则phi(m) = m-1
  - p = m
  - pow(a, p-1, p) = 1

欧几里得算法

欧几里得算法

- 对于 A、B，要求 GCD(A，B)
- 考虑 A = k*B + R
- GCD(A，B) = GCD(k*B + R, B) = GCD(B, R)

扩展欧几里得算法

```
gcd(a, b) == gcd(b, a%b)


假设a,b最大公因数为T,gcd递归下降的终点为(1*T+y*0 == T) 1 * T+ y * 0 == T, 在gcd算法向上回溯的过程中总有:
# a 和 b 是每一次调用 gcd 时的入参
# x 和 y 为对应系数
T = nx*a + ny*b
T = x*b + y*(a%b)
这里 a%b 写成 a-k*b，其中 k = a//b,则
T = b*x + a*y - k*b*y
T = [y]*a + [(x-k*y)]*b
那么 nx, ny = y, (x-a//b*y),由此递推计算x, y
```

例题：a = 96 b = 35 ，应用 XGCD

a	b	x	y	等式
96	35			
35	26			
26	9			
9	8	1	-1	9*1+8*(-1) == 1
8	1	0	1	8*0+1*1 == 1
1	0	1	0	1*1+0*0 == 1

a	b	x	y
96	35	-4	11
35	26	3	-4
26	9	-1	3
9	8	1	-1
8	1	0	1
1	0	1	0

求逆元

我们的目标是对于 A 和 N 找到 A 的逆元 B，使得 `A*B = 1 (mod N)`。扩展欧几里得算法对于给定 (a,b) 扩展欧几里得算法可以计算出 `a*x + b*y = gcd(a, b)`，那么，只要`gcd(A, N) == 1`，令 (a,b) 为 (A, N)，可以计算 `A*x +b*N = 1`，则 `A*x = 1 (mod N)`，x 即为一个满足条件的 B。

代码实现

```
def XGCD(a, b):
    if (b == 0):
        return a, 1, 0
    gcd, x, y = XGCD(b, a%b)
    return gcd, y, x-a//b*y


def inverse(a, n):
    gcd, x, y = XGCD(a, n)
    return x%n if gcd == 1 else None
```

非对称

1. 密钥配送问题
2. 共享密钥数量爆炸
3. 需要签名机制

RSA

- RSA介绍
- RSA计算
- RSA证明
- RSA攻击

RSA介绍

- 利用了大数分解的困难性

流程

1. 生成 p, q
2. 计算 n = p*q
3. 计算 n 的欧拉函数, phin = (p-1)*(q-1)
4. 生成公钥 e，和 phin 互素即可
5. 计算私钥 d，满足 e*d % phin = 1, 例如 d = inverse(e, phin)
6. 丢弃 phin p q
7. 保留 e, n 作为公钥， d 作为私钥

- 加密: C = pow(P, e, n)
- 解密: P = pow(C, d, n)
- P2 = pow(pow(P, e, n), d, n)
- P2 = pow(P, e*d, n)

- P2 = pow(P, k*phin + 1, n)
- P2 = pow(pow(P, phin, n), k, n)*pow(P, 1, n)
- P2 = 1*P

(P ^ e % n) ^d %n (P ^ e) ^ d % n (P ^ e*d) % n