Crypto_MST

- 基础知识回顾
- 如何利用扩展欧几里得求逆元
- 欧拉定理与大数运算
- RSA 标准流程及其变体分析
- 不同分组加密模式的比较

结构

主要参考-21年：

- 基础知识 - 20分
- 数论有关计算 - 10分
- RSA - 10分
- 分组加密模式 - 10分

补充参考-20年：

- 基础知识 - 20分
- 数论有关计算 - 10分
- RSA - 10分
- 分组加密模式 - 10分

补充参考-17年：

- 数论有关计算 - 10分
- 扩展欧几里得 - 10分
- RSA - 20分
- 分组加密模式 - 10分

基础知识

1. CIA代表什么? 机密性(Confidentiality)，完整性(Integrity)，可用性(Availability)

2. 所有加密算法基于哪两条原则? substitution and transposition/permutation

3. 各分组模式的应用场景?

    - ECB - 单个数据加密
    - [CBC - 分组加密建议模式；认证]
    - CFB - 用于认证和数据流的加密
    - OFB - 噪声比较大的信道传输
    - CTR - 分组加密建议模式；高速

4. RSA 基于哪些问题?

    - RSA 问题
        - 目前没有证明，解决RSA必须要大数分解

- GCD 问题 不是一个困难的问题 欧几里得算法
- DLP DH密钥交换场景使用
- modExp 问题 不是一个困难问题 模平方重复算法
- 大数分解 困难问题
  - 如果可以快速大数分解，则可以快速解RSA

5. RSA加密?

```
--- 为什么需要公钥密码
1. 密钥配送问题；ok
2. 共享密钥数量膨胀问题；ok
3. 需要数字签名机制；  ok

PU - e, n
PR - d
A: PU_A(公开), PR_A
B: PU_B(公开), PR_B

加密: C = M^E % N
解密: M = C^D % N

A 想发消息给 B;
    A 使用PU_B加密
    B 使用PR_B解密

XX 算法:
C = M * E % N
C * E^(-1) = M * E * E^(-1) % N

C ^ (E^(-1)) = M ^ E ^ (E^(-1)) % N

签名:
B 想签名一个消息 M

B: S = M^D % N
任何人，要验证签名: M1 = S^E % N
M1 == M
```

8. RSA签名?

9. GCD计算?

```
a,b = b,a%b
直到b=0, 返回a
```

10. 凯撒密码?

```
```

```
unimelb
wpkognd
```

11. 同余方程组
    - x % 7 == 2
    - x % 5 == 3
    - mod 35
    - 2, 9, 16, [23], 30
    - 3, 8, 13, 18, [23], 28, 33

数论有关计算

1. EGCD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 +N*y = 1`，则 `A*x = 1 (mod N)`，x 即为一个满足条件的 B。

a	b	x	y
a	b	nx	ny
b	a%b	x	y

- 1 = b*x +(a-k*b)*y
- 1 = b*x + a*y - k*b*y
- 1 = a*[y] + b*[(x-k*y)]
- nx = y
- ny = x-y*(a//b)
- ps. k = a//b

手算列表格： 11

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

a	b	x	y
a	b	nx	ny

a	b	x	y
b	a%b	x	y

- 1 = b*x +(a-k*b)*y
- 1 = b*x + a*y - k*b*y
- 1 = a*[y] + b*[(x-k*y)]
- nx = y
- ny = x-y*(a//b)
- ps. k = a//b

2. 计算

- a*b % c = (a%c * b%c) % c
- a+b % c = (a%c + b%c) % c

20年

```
pow(pow(1271, 36000075, 111) + 36, 28, 111)
已知 pow(a, phi(111), 111) == 1
phi(111) = phi(3) * phi(37) = 72
所以 pow(1271, 72, 111) == 1
pow(1271, 36000075, 111) = pow(1271, 3, 111) = pow(50, 3, 111)
pow(pow(50, 3, 111) + 36, 28, 111)
pow(50, 28, 111)
16 * pow(50, 8, 111)
= 70
```

- 1271^72 % 111 = 1
- 1271^36000075 % 111
  - 1271^(72*500001+3) % 111
  - 1271^3 % 111
  - 50^3 % 111
- (50^3+36)^28 % 111
- 50^28 % 111
- 50^10 * 50^10 * 50^8 %111
- 4*4*50^8 %111
- 70

(1271^9897801 + 41)^1342 % 101

注意：101是素数，phi(101) = 100

(1271^1 + 41)^42 % 101 (100)^42 % 101 <=> (-1)^42 % 101 1

```
(1271^9897801 + 41)^1342 % 101
(1271 % 101 + 41 % 101)^42 % 101
```

```
(100)^42 % 101
1
```

RSA

1. n 的分解

- a^2 % n = 1
- a^2-1 % n = 0
- a^2-1 = k*n

```
a^2 % n = 1
a^2 - 1 % n = 0
a^2 - 1 = kn
(a+1)*(a-1) = kn

- gcd(a+1, n)
- gcd(a-1, n)
```

2. RSA 参数基本计算

- p = 17
- q = 41

```
1. 生成 p, q
已知
2. 计算 n = p\*q
n = 697
3. 计算 n 的欧拉函数, phin = (p-1)\*(q-1)
phin = 640
4. 生成公钥 e, 和 phin 互素即可
令 e = 3, 显然 gcd(3, 640) == 1
5. 计算私钥 d, 满足 e*d % phin = 1, 例如 d = inverse(e, phin)
一个满足条件的d = inverse(3, 640) = 427
6. 丢弃 phin p q
OK
7. 保留 e, n 作为公钥, d 作为私钥
公钥: 3, 697
私钥: 427
```

3. 三因子RSA

- e = 13
- d = 15637

```
1. 生成 p, q, r
题目给出 p, q, r = 23, 29, 31
2. 计算 n = p\*q\*r
n = 20677
3. 计算 n 的欧拉函数, phin = (p-1)\*(q-1)\*(r-1)
phin = 18480
4. 生成公钥 e，和 phin 互素即可
e = 13
5. 计算私钥 d，满足 e*d % phin = 1, 例如 d = inverse(e, phin)
d = 15637
6. 丢弃 phin p q
OK
7. 保留 e, n 作为公钥， d 作为私钥
公钥：13, 20677
私钥：15637
```

```
C^d = M^(e*d) # 确定加解密正常
e*d = k*fn+1

======
p = q = r = 31 # 31*31*30注意一下，欧拉函数计算
```

分组模式

1. 模式

- ECB
- CBC
- OFB
- CFB
- CTR

2. nonce(IV的作用)

随机性

3. 加解密公式

4. 错误传递