# 证明中国剩余定理

- 中国剩余定理：
- 设$m_1, m_2, m_3 \cdots, m_r$两两互素，则以下同余方程组

$$x \equiv a_i \pmod{m_i},\ i=1,2,3,\cdots r \quad (a)$$

模 $M=m_1 m_2 m_3 \cdots m_r$ 的唯一解为

- $x = \sum_{i=1}^{r} a_i * M_i * (M_i^{-1} \bmod m_i) \bmod M \quad (b)$
- 其中$M_i = M/m_i$

## 证明中国剩余定理

(1) 先证明$\sum_{i=1}^{r} a_i * M_i * (Mi^{-1} \bmod mi)$(c)是同余方程组(a)的一个解

- 对于任意1≤j≤r，都有$\sum_{i=1}^{r} a_i * M_i * (Mi^{-1} \bmod mi) \bmod m_j = a_j$，所以(c)是(a)的一个解。

## 证明中国剩余定理

(2) 再证明(b)是同余方程组(a)的模M唯一解

- 假定$x_1$及$x_2$是(a)的两个不同解，即
- $x_1 = a_i \bmod m_i$， $i=1, 2, 3\cdots, r$
- $x_2 = a_i \bmod m_i$， $i=1, 2, 3\cdots, r$
- 则$x_1-x_2 = 0 \bmod m_i$， $i=1, 2, 3, \cdots, r$
- 即$m_i \mid (x_1-x_2)$， $i=1, 2, 3, \cdots, r$
- 又因为$m_1, m_2, m_3, m_r$两两互素，所以 $M \mid (x_1-x_2)$
- 即 $x_1 = x_2 \bmod M$
- 因此(b)是(a)模M的唯一解。

## RSA算法的数学基础 – Euler函数的乘法性质

- 若n1,n2互素，则$\varphi(n_1*n_2)=\varphi(n_1)*\varphi(n_2)$

- 例如: $\varphi(3*5)=\varphi(3)*\varphi(5)=2*4=8$

- 与15互素的数包括: 1,2,4,7,8,11,13,14

## RSA算法的数学基础 – Euler函数的乘积公式

- $\varphi(n) = n * \prod_{p|n}(1 - 1/p)$

- 例如: $\varphi(10) = 10 * (1\text{-}1/2) * (1\text{-}1/5) = 4$

## RSA算法证明

- 方法1：

设m是明文，c是密文，$c=m^e \bmod n$。现证明$m=c^d \bmod n$。

因为$\phi(n)= \phi(p*q) = \phi(p)*\phi(q)= (p-1)(q-1)$，

又因为$ed = 1 \bmod (p-1)(q-1)$，

所以一定可以找到一个k使得$ed = 1 + k(p-1)(q-1)$成立，

于是$c^d = m^{ed} = m^{1+k(p-1)(q-1)} = m * m^{k(p-1)(q-1)}$

$= m * (m^{\phi(n)})^k = m * (1)^k = m \bmod n$

## RSA算法证明

➢ 方法1：

为什么 $(m^{\phi(n)})^k = (1)^k \bmod n$ ?

a = a' mod n

b = b' mod n

则一定有a*b = a'* b' mod n，这是因为：

a = kn+a'

b = jn+b'

a*b=(kn+a')(jn+b') = kjnn + a'jn + b'kn + a'b'

## RSA算法证明

➢ 方法1:

上述证明的前提是gcd(m, n)=1。

当gcd(m, n) ≠ 1时，则一定有gcd(m, n)=p或gcd(m, n)=q。现假设gcd(m, n)=p，即m是p的倍数，则m与q一定互素。于是有：

$m^{\phi(q)} = 1 \bmod q \rightarrow m^{(q-1)} = 1 \bmod q \rightarrow m^{(q-1)*k(p-1)} = 1 \bmod q \rightarrow$

$m^{\phi(n)*k} = 1 \bmod q \rightarrow m^{\phi(n)*k} = q*s + 1 \rightarrow m * m^{\phi(n)*k} = m*q*s + m \rightarrow$

$m^{\phi(n)*k+1} = cp * q * s + m \rightarrow m^{\phi(n)*k+1} = m \bmod n \rightarrow m^{ed} = m \bmod n$

## RSA算法证明

- 方法2:

$ed = 1 \bmod \phi(p*q)$ ➔

$ed - 1 = k(p-1)(q-1)$ ➔

①设$m = 0 \bmod p$，则$m^{ed} = 0 = m \pmod p$

②设$m \neq 0 \bmod p$，则

$m^{ed} = m^{(ed-1)}*m = m^{k(p-1)(q-1)}*m$

$= (m^{(p-1)})^{k(q-1)} *m = (1)^{k(q-1)} *m = m \pmod p$ 【费马小定理】

## RSA算法证明

➢ 方法2：

综合①②两种情况可得：

$m^{ed} = m \bmod p$ 　　(a)

同理可证：

$m^{ed} = m \bmod q$ 　　(b)

## RSA算法证明

➢ 方法2：

根据(a)(b)可得：

$m^{ed} - m = 0 \bmod p$

$m^{ed} - m = 0 \bmod q$

即$m^{ed} - m$既可以被p整除，又可以被q整除，

而p、q是互素的，所以$m^{ed} - m$一定同时包含因子p及q，于是有：

$m^{ed} - m = 0 \bmod p*q \rightarrow m^{ed} = m \bmod p*q$ （明文已恢复）

上述结论其实可以由中国剩余定理得出：

$m^{ed} - m = 0*q*(q^{-1} \bmod p)+0*p*(p^{-1} \bmod q) \pmod{p*q} = 0 \bmod p*q$

## RSA算法证明

- 方法2:

上述划波浪线的证明也可以换成以下方法:

$m^{ed} = m \bmod P$ (a)

$m^{ed} = m \bmod Q$ (b)

由(a)(b)可得:

$m^{ed} = k_1*P + m$

$m^{ed} = k_2*Q + m$

$\rightarrow k_1*P + m = k_2*Q + m \rightarrow k_1 * P = k_2 * Q \rightarrow$

$k_1 * P = 0 \bmod Q$

$k_2 * Q = 0 \bmod P$

## RSA算法证明

➢ 方法2：

由于P、Q 互素，所以

$k_1 = a*Q, \quad K_2 = b*P$

➔

$m^{ed} = k_1*P + m$

$m^{ed} = k_2*Q + m$

$m^{ed} = a*Q*P + m = m \bmod (P*Q)$

或者

$m^{ed} = b*P*Q + m = m \bmod (P*Q)$

## RSA算法应用

- 保密p,q,(p-1)*(q-1)的前提下：
- 已知e、n的情况下，无法算出d;
- 同理在d、n已知的情况下，也无法算出e;

- aes+rsa:
- 对文件加密用的是128位密钥的aes算法。
- 用rsa算法加密文件的话，速度太慢，故没有采纳。

## 调用openssl库函数

0. RSA_generate_key()　产生密钥

　如RSA_generate_key(256, 0x10001, NULL, NULL);

　N位数　公钥

1. RSA_public_encrypt()　公钥加密
2. RSA_private_decrypt() 私钥解密
3. RSA_private_encrypt() 私钥加密
4. RSA_public_decrypt()　公钥解密
5. RSA_new() 分配一个RSA结构指针
6. RSA_free() 释放一个RSA结构指针
7. BN_new() 分配一个大数; BN:Big Number

## 调用openssl库函数 – 大数处理

- BN指128位、256位、512位、1024位整数。
- 密钥N为128位即16字节的情况下，明文长度必须是16字节，并且明文的值一定要小于N，密文长度也只能是16字节。
- 设明文char m[]={'A','B','C','D',0,0,···}（共12个0）
- 则实际上该明文是被当成如下这个大数来处理：
- 0x41424344000000000000000000000000

## RSA算法 - 软件应用

(1)软件打开时显示一个机器码

其中机器码m'=rsa(mac, 公钥)

(2)软件作者: mac=rsa(m', 私钥)

注册码sn=(mac, 私钥)

(3)软件验证注册码:

rsa(sn, 公钥)==mac

- 为什么加密时明文m的位数应该与N相同?
  - 假定m很小，如$m^e < N$, 则解密时不需要用到d，只要对m开e次方即可。

## 数字签名

- 假定以下是A的公钥（e、n）及私钥（d、p、q）：
- n=F03E1D9F9BFB6827B4A49AED686F7790868CA58FA7BA7110C29D3241A8E2EF53
- p=F9298817691C971281C3CD6D203C28BF
- q=F6D5EB95C1915957FB00604580B9EA6D
- d=08F1C718922E220A9867287D7E4DE81D9EED52D623E1BB48758146F22C515D41
- e=010001

## 数字签名

- 假定A要发一封信给B:
  - 信的内容L="Hello, I'm A."
- 如何对信进行加密?
  - L' = RSA(L, B的公钥)
- A把L'发给B，B收到后如何解密?
  - L = RSA(L', B的私钥)

## 数字签名

- A如何对信件进行签名?

- 首先对信的内容计算摘要(digest),这里采用MD5算法:
  - M = MD5(L) = 82228599d3e30286a22e7d170ebe2546
- 用A的私钥对M进行签名(实际上是用A的私钥对M加密):
  - M' = RSA(M, A的私钥)
    =6EFADDDBBEC3F995F4F4398F2A6049BF42A41645B36B1A2E1AE6C52BFE11950B
- 此时，M'就是A对信件摘要M的签名。

## 数字签名

- 假定A把L及M'都发送给B。
- B如何对A的签名M'进行验证?

- 首先要用A的公钥对M'进行解密：
  - m = RSA(M', A的公钥) =82228599d3e30286a22e7d170ebe2546
- 最后还要判断m是否正确:
  - 若MD5(L) = m，则证明此信确实是A所发。

## RSA算法 – 安全性

- 攻击RSA的可能方法有：
- 暴力搜索（ brute force key search ）
- 数学攻击（ mathematical attacks，基于计算φ（n）的难度，通过分解模n）
- 计时攻击（ timing attacks ）
- 选择密文攻击（ chosen ciphertext attacks ）

## RSA算法 – 因式分解问题

- 数学方法攻击有三种形式:
  - n=p*q，因此计算φ（n），然后计算d
  - 直接确定φ（n）并计算d
  - 直接找到d

- 目前假设1024-2048位的RSA是安全的
- 确保p、q具有相似的大小并匹配其他约束条件

## RSA算法 － Timing Attacks

- 由Paul Kocher在20世纪90年代中期开发
- 基于对计算加密操作所需时间的观察
  - 用小的或大的数字相乘
  - 或者执行的指令不同
- 根据所用时间推断操作和大小 （page 16）
- RSA利用了指数运算所需的时间
- 对策
  - 使用常数求幂时间
  - 添加随机延迟
  - 计算中使用盲值

## RSA算法 － Chosen Ciphertext Attacks

- RSA易受Chosen Ciphertext Attacks （CCA）的攻击
  - 攻击者选择密文并获得解密后的明文
  - 选择密文来利用RSA的属性获取帮助密码分析的信息
  - 可以用随机的明文来抵抗简单的攻击或者使用Optimal Asymmetric Encryption Padding （OASP）来修改明文

## RSA算法 – Chosen Ciphertext Attacks

- 加密：

$$C = t^e \mod n$$

- 假设攻击者选择2作为明文，计算 $C_a = 2^e \mod n.$
- 对受害者发送 $C_b = C_a * C$

$$C_b = C \cdot 2^e = t^e 2^e \mod n$$

- 受害者解密：

$$(C_b)^d = [t^e 2^e]^d = t^{ed} 2^{ed} = 2t \mod n$$

- 即可得到t的值

# Thank you！