密 码 学

第五章 公钥密码算法

网络空间安全学院
胡 伟　朱丹
weihu/zhudan@nwpu.edu.cn

知识回顾-公钥密码的基本思想

- 将密钥 K 一分为二：K_e 和 K_d。K_e 专门加密，K_d 专门解密， $K_e \neq K_d$
- 由 K_e 不能简单地计算出 K_d，于是可将 K_e 公开，使密钥 K_e 分配简单
- 由于 $K_e \neq K_d$ 且由 K_e 不能计算出 K_d，所以 K_d 便成为用户的指纹，可方便地实现数字签名和身份认证

称上述密码体制为公开密钥密码，简称为公钥密码

知识回顾-公钥密码的工作方式

- 确保数据秘密性（安全性分析）：
  - ① 只有B才有K_{dB}，因此只有B才能解密，所以确保了数据的秘密性
  - ② 任何人都可查PKDB得到B的K_{eB}，所以任何人都可冒充A给B发送数据。

  不能确保数据的真实性

$C = E(M,\ K_{eB})$ $\longrightarrow$ $M = D(C, K_{dB}) = D(E(M,\ K_{eB}), K_{dB})$

知识回顾-公钥密码的工作方式

- 确保数据真实性（安全性分析）：
  - ① 只有A才有K_{dA}，因此只有A才能加密产生C，所以确保了数据的真实性
  - ② 任何人都可查PKDB得到A的K_{eA}，所以任何人都可解密得到明文。不能确保数据的秘密性

A

B

$C = D(M, K_{dA})$ $\longrightarrow$ $M = E(C, K_{eA}) = E(D(M, K_{dA}), K_{eA})$

知识回顾-公钥密码的工作方式

- 同时确保数据机密性和真实性(安全性分析)：
  - ① 只有A才有K_{dA}，因此只有A才能加密产生S，所以确保了数据的真实
  - ② 只有B才有K_{dB}，因此只有B才能解密获得明文，所以确保了数据的机密性

A

B

$$C = E(S, K_{eB}) = E(D(M, K_{dA}),\ K_{eB})$$

$$\longrightarrow$$

$$M = E(S, K_{eA}) = E(D(C, K_{dB}), K_{eA})$$

知识回顾-RSA公钥密码算法

- 加解密算法
  - 随机地选择两个大素数p和q，而且保密
  - 计算$n = p * q$，将n公开
  - 计算$\varphi(n) = (p-1) * (q-1)$，对$\varphi(n)$保密
  - 随机地选取一个正整数e，$1 < e < \varphi(n)$且$(e, \varphi(n)) = 1$，将 e公开
  - 根据 $ed = 1 \bmod \varphi(n)$，求出d，并对d保密
  - 加密运算：$C = M^e \bmod n$
  - 解密运算：$M = C^d \bmod n$
  - 公开密钥$K_e = <e, n>$，保密密钥$K_d = <p, q, d, \varphi(n)>$

知识回顾-RSA公钥密码算法

- 解密算法E和加密算法D的可逆性
  - 要证明：$D(E(M)) = M$，即要证明

$$M = C^d = (M^e)^d = M^{ed} \bmod n$$

  - 由$ed = 1 \bmod \varphi(n)$，有$ed = t\varphi(n) + 1$，其中t为整数。所以，

$$M^{ed} = M^{t\varphi(n)+1} \bmod n$$

  - 因此要证明 $M^{ed} = M \bmod n$，只需证明

$$M^{t\varphi(n)+1} = M \bmod n$$

  - 在$(M, n) = 1$的情况下，根据数论(Euler定理)，$M^{\varphi(n)} = 1 \bmod n$，所以，

$$M^{t\varphi(n)} = 1 \bmod n$$

  - 于是，$M^{t\varphi(n)+1} = M \bmod n$

$$M^{ed} \longrightarrow M^{t\varphi(n)+1}$$

知识回顾-RSA公钥密码算法

- 解密算法E和加密算法D的可逆性
  - $(M，n) \neq 1$的情况下，分两种情况：
  - 第一种情况（$M \geq n$时无法解密）：$M \in \{1, 2, 3, \ldots, n-1\}$
  - 因为$n = pq$, p和q为素数，$M \in \{1, 2, 3, \ldots, n-1\}$，且$(M, n) \neq 1$
  - 这说明M必含p或q之一为其因子，而且不能同时包两者，否则将有$M \geq n$，与$M \in \{1, 2, 3, \ldots, n-1\}$矛盾
  - 不妨设$M = ap$。又因q为素数，且M不包含q，故有$(M, q) = 1$，于是有，

$$M^{\varphi(q)} = 1 \bmod q$$

Euler定理

知识回顾-RSA公钥密码算法

- 解密算法*E*和加密算法*D*的可逆性
  - 进一步有$M^{t(p-1)\varphi(q)} = 1 \bmod q$。因为$q$是素数，$\varphi(q) = (q-1)$，所以有$t(p-1)\varphi(q) = t\varphi(n)$，

$$M^{t\varphi(n)} = 1 \bmod q$$

  - 于是，$M^{t\varphi(n)} = bq + 1$，其中b为整数。两边同乘M，$M^{t\varphi(n)+1} = bqM + M$。因为$M = ap$（根据上一页假设），故

$$M^{t\varphi(n)+1} = bqap + M = abn + M$$

  - 取模n得，$M^{\varphi(n)+1} = M \bmod n$
  - 第二种情况：$M = 0$
  - 当$M = 0$时，直接验证，可知命题成立

知识回顾-RSA公钥密码算法

- 在计算上由公开的加密钥不能求出解密钥
  - 公开密钥$K_e = \langle e, n \rangle$
  - 保密密钥$K_d = \langle p, q, d, \varphi(n) \rangle$
  - 假设攻击者截获密文C，*需恢复明文*

$$n = pq$$

$$\varphi(n) = (p - 1)(q - 1)$$

$$ed \equiv 1 \bmod \varphi(n)$$

$$M \equiv C^d \bmod n$$

$$C$$

1. RSA密码算法中，假设素数$p = 29$，$q = 23$，选择公钥$e = 19$，请按RSA密钥产生算法和参数选取原则为用户产生公钥和私钥对。
2. 用所产生的密钥对消息123进行解密。
3. 某用户收到消息m = 13 | sig = 208，试对该消息来源的真实性进行验证。
4. 验证：使用RSA Calculator，https://www.cs.drexel.edu/~jpopyack/IntroCS/HW/RSAWorksheet.html

1. 密钥生成

$$n = p * q = 29 * 23 = 667$$

$$\varphi(n) = (p-1)(q-1) = 28 * 22 = 616$$

由$ed = 1 \bmod 616$可求解出私钥$d = 227$

2. 解密

$$M = C^{d} \bmod n = 123^{227} \bmod 667$$

将227转化为二进制数11100011，逐步计算

$123^{1} \bmod 667 = 123$　　$123^{2} \bmod 667 = 455$

$123^{4} \bmod 667 = 255$　　$123^{8} \bmod 667 = 326$

$123^{16} \bmod 667 = 223$　　$123^{32} \bmod 667 = 371$

$123^{64} \bmod 667 = 239$　　$123^{128} \bmod 667 = 426$

因此，$M = (123^{1} * 123^{2} * 123^{32} * 123^{64} * 123^{128}) \bmod 667$

$= (123 * 455 * 371 * 239 * 426) \bmod 667$

$= 633$

3. 对签名进行验证

$sig = 208, e = 19$

$ver(sig, e) = 208^{19} \bmod 667$

类似的$19 = (10011)_B$

$208^1 \bmod 667 = 208$ $\quad$ $208^2 \bmod 667 = 576$

$208^4 \bmod 667 = 277$ $\quad$ $208^8 \bmod 667 = 24$

$208^{16} \bmod 667 = 576$

因此，

$208^{19} \bmod 667 = (208 * 576 * 576) \bmod 667 = 254 \neq 13$，故消息来源不真实

章节安排

Outline

1. RSA算法参数的选择
2. RSA算法实现

章节安排

Outline

5.4(1) RSA算法参数的选取

- (1) p和q要足够大
  - RSA分解当前的记录是768位2进制数（232位10进制数)
  - 目前主流的密钥长度为1024位 – 4096位
  - 一般应用：p和q应至少为512位，使n达1024位
  - 重要应用：p和q应至少为1024位，使n达2048位以上
- (2) p和q要是强素数
  - 只要(p-1)、(p+1)、(q-1)、(q+1)之一有小的素因子，n就容易分解

定义: p为素数，若p满足以下两个条件，则称p为强素数或一级素数
(1) 存在两个大素数p_1和p_2，使得$p_1 \mid p - 1$，$p_2 \mid p + 1$
(2) 存在4个大素数r_1，r_2，r_3和r_4，使得$r_1 \mid p_1 - 1$，$r_2 \mid p_1 + 1$，$r_3 \mid p_2 - 1$，$r_4 \mid p_2 + 1$

https://en.wikipedia.org/wiki/RSA_Factoring_Challenge https://github.com/DarkenCode/yafu

5.4(1) RSA算法参数的选取

- (3) p和q位数相差不能太小，也不能太大
- 若p和q相差很小，可以估算$(p+q)/2$约为$\sqrt{n}$，可以用$\sqrt{n}$来估算$(p+q)/2$，联合$p*q$和$p+q$即可分解n
  - 例，假设$p=2$，$q=3$，$n=6$，$(p+q)/2=2.5$, $\sqrt{6}\approx 2.45$，$p+q$应该在4.5附近取值
  - 例，假设p和q相差很小，$n=164009$，$\sqrt{n}\approx 405$，可以估计$(p+q)/2=405$，又$p*q=164009$，可得$p=409$，$q=401$
- 若p和q相差太大，则从可从小的素数开始尝试分解n

5.4(1) RSA算法参数的选取

- (4) 在给定的条件下，找到的$d = e$，这样的密钥必须舍弃

- (5) $p - 1$和$q - 1$的最大公因子要小，最好是2，可选择p和q为理想的强素数
  - 在唯密文攻击中，假设攻击者截获了某个密文

$$C_1 = M^e \bmod n$$

  - 攻击者进行迭代攻击

$$C_i = C_{i-1}^e = M^{e^i} \bmod n$$

  - 如果有$e^i = 1 \bmod \varphi(n)$，则有$C_i = M \bmod n$
  - 如果使得$e^i = 1 \bmod \varphi(n)$成立的i值很小，则很容易进行密文迭代攻击

5.4(1) RSA算法参数的选取

- 例，设p = 17，q = 11，n = 17*11 = 187，e = 7，M = 123，则有
  - $C_0 = M = 123$
  - $C_1 = {C_0}^e = 123^7 \, mod \, 187 = 183$
  - $C_2 = {C_1}^e = 183^7 \, mod \, 187 = 72$
  - $C_3 = {C_2}^e = 72^7 \, mod \, 187 = 30$
  - $C_4 = {C_3}^e = 30^7 \, mod \, 187 = 123 = M$
  - $C_5 = {C_4}^e = 123^7 \, mod \, 187 = 183$
- 可见，迭代加密出现$C_5 = C_1$， $C_4 = M$，周期t = 4，$\varphi(n) = (p-1)(q-1) = 160$，周期$t$是$\varphi(n)$的因子

5.4(1) RSA算法参数的选取

- (6) e的选择
  - 随机且二进制表示中含1多就安全，但加密速度慢
  - 为了使加密速度快，二进制表示中含1应尽量少
  - 有学者建议取$e = 2^{16} + 1 = 65537$，它是素数，且二进制表示中只含两个1
  - 若e太小，对于小的明文M，$C = M^e$未超过n，无需对n取模，此时直接对C开e次放即可求出明文M，也不安全

- (7) d的选择
  - 为了使加密速度快，希望选用尽量小的d
  - d不能太小，要足够大，否则不安全
  - 当d小于n的1/4时，已有求出d的攻击方法

5.4(1) RSA算法参数的选取

- (8) 模数n的使用限制
  - 不要许多用户共用一个模n，否则易受共模攻击
  - 设用户A的加密密钥为e_A，用户B的加密密钥为e_B，他们使用同一个模数n，对于同一条明文有

$$C_A = M^{e_A} \bmod n$$

$$C_B = M^{e_B} \bmod n$$

  - 当e_A和e_B互素时，可利用欧几里德算法求出两个整数r和s，使得

$$re_A + se_B = 1$$

  - 于是，${C_A}^r {C_B}^s = M^{re_A + se_B} = M \bmod n$

5.4(2) 素数的产生

- 根据素数的定义，因子只有1和p本身
  - 1 ～ $p-1$，检查能够整除p的整数的个数
- 测试算法
  - 1 ～ $\sqrt{p}$即可
  - 如何快速筛选出1到整数n之间的所有素数?

2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	…
✓		✕		✕		✕		✕		✕		✕		✕		✕		✕		…

2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	…
✓	✓	✕		✕		✕	✕	✕		✕		✕	✕	✕		✕		✕	✕	…

2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	…
✓	✓	✕	✓	✕	✓	✕	✕	✕	✓	✕	✓	✕	✕	✕	✓	✕	✓	✕	✕	…

5.4(3) 大素数的产生

- 概率产生法
  - 目前最常用的概率性算法是Miller检验算法
  - Miller检验算法已经成为美国的国家标准

- Miller检验算法
  - 欧拉定理：若n, a为正整数，且n, a互素，则有$a^{\varphi(n)} \equiv 1 \, mod \, n$
  - 费马小定理（欧拉定理的特殊情况）：如果p是一个素数，而整数a不是p的倍数（a和p互素），则有$a^{p-1} \equiv 1 \bmod p$
  - 不断取$a \in [1, p-1]$，且$a \in Z$，验证$a^{p-1} \equiv 1 \bmod p$是否成立，不成立则p肯定不是素数，共取s次
  - 若s次均通过测试，则p不是素数的概率不超过2^{-s}

5.4(4) RSA密钥产生及加解密

- https://www.openssl.org
- http://gmssl.org

- 产生私钥
  - openssl genrsa -out rsa_2048_priv.pem 2048
- 产生公钥
  - openssl rsa -pubout -in rsa_2048_priv.pem -out rsa_2048_pub.pem
- RSA加密
  - openssl rsautl -encrypt -inkey rsa_2048_pub.pem -pubin -in plaintext.txt -out ciphertext.txt
- RSA解密
  - openssl rsautl -decrypt -inkey rsa_2048_priv.pem -in ciphertext.txt -out plaintext2.txt

章 节 安 排

Outline

RSA算法参数的选择

RSA算法实现

5.5(1) RSA密钥产生及加解密

- 加解密运算
  - 加密运算：$C = M^e \bmod n$
  - 解密运算：$M = C^d \bmod n$
- 模幂运算算法
  - 模幂运算基本定义
  - 重复平方算法
  - 滑动窗口算法
  - CRT算法
  - 蒙哥马利算法（了解）

5.5(1) RSA密钥产生及加解密

- 模幂运算的基本定义
  - $C = ((M * M \bmod n) * M \bmod n) * M \bmod n\ldots$
  - 简单直观
  - 只适用于e很小的情况
  - 当e较大（如e = 65537）时效率低

```
unsigned long mod_exp(unsigned long m, unsigned long e, unsigned long n)
{
    unsigned long result = 1;
    while (e > 0){
        result = (result * m) % n;

        e = e - 1;
    }

    return result;
}
```

5.5(2) 重复平方算法

- 从右至左：$M = C^d \mod n$

$$d = (d_{w-1}, d_{w-2}, \ldots, d_1, d_0)$$

$$d = d_{w-1}*2^{w-1} + d_{w-2}*2^{w-2} + \ldots + 2^1*d_1 + 2^0*d_0$$

$$M = C^d \mod n = C^{d_{w-1}*2^{w-1}+d_{w-2}*2^{w-2}+\cdots+d_1*2^1+d_0}$$

$$= (C^{d_{w-1}})^{2^{w-1}} * (C^{d_{w-2}})^{2^{w-2}} * \cdots (C^{d_1})^2 * C^{d_0}$$

$C^{2^{(w-1)}}$ / 1　　$C^{2^{(w-2)}}$ / 1　　…　　C^2 / 1　　C / 1

- 通过逐次平方，依次计算C^2，C^4，…，$C^{2^{w-2}}$，$C^{2^{w-1}}$
- 当$d_i = 1\ (0 \le i \le w-1)$时，结果乘以$C^{2^i}$　　$C^{2^i} = C^{2^{i-1}*2} = (C^{2^{i-1}})^2$
- 当$d_i = 0$时，$C^{d_i} = 1$，乘以 1 (即1^{2^i})

5.5(2) 重复平方算法

- 从右至左：$M = C^d \bmod n$

```
// right to left
unsigned long Rep_Sqr(unsigned long c, unsigned long d, unsigned long n)
{
    unsigned long result = 1;

    while (d > 0){
        if ((d & 0x01) == 1)
            result = (result * c) % n;

        c = (c * c) % n;
        d >>= 1;
    }

    return result;
}
```

5.5(2) 重复平方算法

- 从左至右：$M = C^d \bmod n$

$$d = (d_{w-1}, d_{w-2}, \dots, d_1, d_0)$$

$$d = d_{w-1} * 2^{w-1} + d_{w-2} * 2^{w-2} + \dots + 2^1 * d_1 + 2^0 * d_0$$

$$M = C^d \bmod n = C^{d_{w-1}*2^{w-1}+d_{w-2}*2^{w-2}+\cdots+d_1*2^1+d_0}$$

- 从最高的非0密钥位($d_{w-1} = 1$)开始
- 以$C^{d_i*2^i}$的计算为例
  - $d_i * 2^i = \left(d_i * 2^{i-1}\right) * 2 = \dots = (d_i * 2^0) * 2 \overbrace{* 2 * \dots * 2}^{i\text{个}2\text{，即}2^i}$
  - 当$d_i = 1$时，$C^{d_i*2^i} = C^{2^i}$，逐次平方　　$C^{2^i} = C^{2^{i-1}*2} = (C^{2^{i-1}})^2$
  - 当$d_i = 0$时，$C^{d_i*2^i} = 1$，逐次平方仍然为1

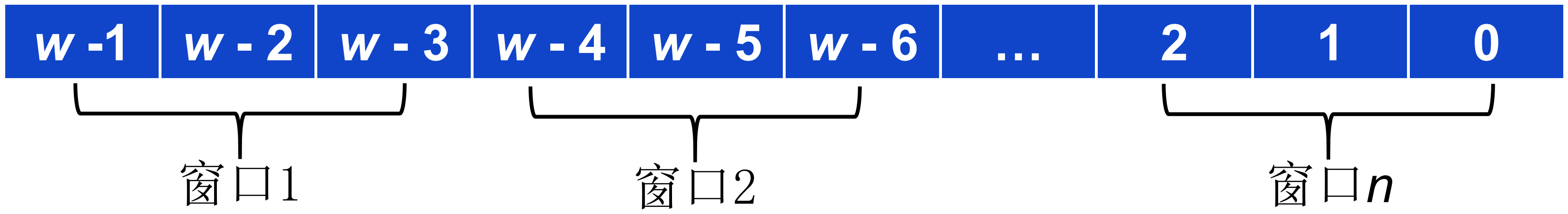
5.5(2) 重复平方算法

- 从左至右：$M = C^d \bmod n$

```
// left to right
unsigned long Rep_Sqr(unsigned long c, unsigned long d, unsigned long n)
{
    unsigned long d_val, d_width = 1;
    unsigned long s = 1, result;

    d_val = d; // copy d to d_val for processing
    while(d_val > 1){ // determine the position of the MSB of d
        d_width <<= 1;
        d_val >>= 1;
    }
    while (d_width > 0){
        if ((d & d_width) != 0) result = (s * c) % n;
        else result = s;

        s = (result * result) % n;
        d_width >>= 1;
    }

    return result;
}
```

5.5(3) 滑动窗口算法

- 每次处理一个窗口大小(k比特密钥)
- 分为固定窗口和可变窗口
- 分为从左至右和从右至左
- 重复平方法是窗口大小为1的特例
- 滑动窗口算法是从二进制数到多进制数情况的扩展
- 例如，窗口大小为3时，即为8进制数的情况



5.5(3) 滑动窗口算法

- 滑动窗口算法：$M = C^d \bmod n$

输入c，$d = (d_t, d_{t-1}, \ldots, d_1, d_0)$，其中$d_t = 1$，整数$k \geq 1$
输出c^d
1: 预计算 // $c, c^2, c^3, \ldots, c^{2^k-1}$
$g_1 = c$, $g_2 = c^2$
for $i = 1$ to $2^{k-1} - 1$ do $g_{2i+1} = g_{2i-1} * g_2$ // 初始化表
2: $A = 1$, $i = t$
3: while $i \geq 0$ do {
if($d_i == 0$) then $A = A^2$, $i = i - 1$
else { 寻找$i - L + 1 \leq k$，且$d_L = 1$的最长密钥串$p = (d_i, d_{i-1}, \ldots, d_L)$
$A = A^{2^{i-L+1}} * g_p$, $i = L - 1$
}
}

5.5(4) CRT算法

- 中国剩余定理(Chinese Remainder Theorem, CRT)表明，如果，

$$a = x \bmod n$$

$$b = x \bmod m$$

- 那么x存在唯一mod mn的解，当且仅当$gcd(m, n) = 1$

- 对于RSA，m和n为素数p和q，计算$M \bmod p$ 和$M \bmod q$ ：

$$M_p = C^d \bmod p = C^{d \bmod p-1} \bmod p \qquad C^{\varphi(p)} = 1 \bmod p$$

$$M_q = C^d \bmod q = C^{d \bmod q-1} \bmod q \qquad C^{\varphi(q)} = 1 \bmod q$$

- 那么C^d存在唯一mod pq的解，当且仅当$gcd(p, q) = 1$

5.5(4) CRT算法

- 计算常数$T = p^{-1} \bmod q$，

$$u = (M_q - M_p) * T \bmod q$$

$$M = M_p + u * p$$

- 其中，$M_p = C^d \bmod p$ 和 $M_q = C^d \bmod q$：

$$M_p = C^d \bmod p = C^{d \bmod p-1} \bmod p$$

$$M_q = C^d \bmod q = C^{d \bmod q-1} \bmod q$$

基本思想：将mod n级别的运算转化为mod p和mod q级别的运算

5.5(5) 蒙哥马利算法

蒙哥马利乘法

- 优化模乘$x \cdot y \bmod q$运算中取模环节
- 原始取模操作：除法取余
- 蒙哥马利算法：转化为移位
  - 以R表示约简操作（R 为 2 的幂）
  - 首先将变量转化为Montgomery形式，如x转化为$xR \bmod q$
  - $xR \cdot yR = zR^2$, 通过Montgomery约简得$zR^2 \cdot R^{-1} = zR$
  - 约简后的zR仍为Montgomery形式，可继续参与后续运算
  - 最终结果乘以$R^{-1} \bmod q$转回标准形式（非Montgomery）

课后阅读与实践

- RSA Calculator，
  https://www.cs.drexel.edu/~jpopyack/IntroCS/HW/RSAWorksheet.html
- OpenSSL和GmSSL实践
- 阅读和理解Miller-Rabbin算法
- CRT-RSA算法，https://www.di-mgt.com.au/crt_rsa.html
- CTF密码学中RSA学习以及总结
  https://blog.csdn.net/huanghelouzi/article/details/82943615

感谢聆听!

THANK YOU FOR YOUR ATTENTION!