第三讲 幂模

引言

上一讲，咱们介绍了公钥密码中知名度最广的一种，RSA。你应该已经知道RSA的安全性建立在这样一个正向计算容易，而反向计算难的一个函数之上。

方向	函数
正向	已知两个大素数p和q，求p和q的乘积n。
反向	已知两个大素数的乘积n，求它的两个素因子p和q。

这种函数，我们称为单向陷门函数，只要有一种单向陷门函数，就能设计出一种公钥密码。

在接下来的两讲中，我们将介绍一种新的单向陷门函数和一种与RSA不一样的数字签名算法。

幂模

这种新的单向陷门函数就是幂模。

其实幂模运算我们并不陌生，在RSA中已经多次用到幂模。RSA无论是加密还是解密都是通过幂模运算完成的。

RSA加解密	运算
加密	$c = p^e \bmod n$
解密	$p = c^d \bmod n$

所谓幂模，就是先做一次幂运算，再做一次模运算。

$$g^x \bmod n = y \tag{1}$$

当然，只要涉及模运算，那么所有参与运算的对象g, x, n, y都是整数。

很容易证明模运算有以下特别好的性质。

$$(a \bmod n) \times (b \bmod n) \bmod n = a \times b \bmod n \tag{2}$$

也就是说，先模再乘和先乘再模，只要最后都模了同一个模数，结果都是一样的。

交换律

有了这个性质，我们首先得到幂模运算的这种两次运算交换律。

$$< Empty\ Math\ Block > \tag{3}$$

$$(g^{a} \bmod n)^{b} \bmod n = (g^{b} \bmod n)^{a} \bmod n = g^{a\times b} \bmod n \tag{4}$$

例如：

$$(2^{4} \bmod 13)^{5} \bmod 13 = (2^{5} \bmod 13)^{4} \bmod 13 = 2^{4\times 5} \bmod 13 \tag{5}$$

单向性

此外幂模运算还满足单向性，也就是说已知g和x容易计算得到y，而已知g和y很难求得x。

$$g^{x} \bmod n = y \tag{6}$$

这种反向运算，我们称为离散对数运算，记作$x \equiv \mathrm{ind}_g y \pmod n$。

我们举个例子来说明幂模运算的单向性。

当取底数g为2，模数n为13时，无论指数x取几，都可以快速的求得对应的y。

x	y
0	1
1	2
2	4
3	8
4	3
5	6
6	12
7	11
8	9
9	5
10	10
11	7
12	1

一方面，这个表显示结果的最后一行，也可以再次印证欧拉定理，即当a与n互素时，$a^{\varphi(n)} = 1$。在这个例子中，2与13显然是互素的，而由于13是素数，所以$\varphi(13) = 13 - 1 = 12$，所以$2^{12} \bmod 13 = 1$。

另一方面，我们还可已看出，当x从0到$n-1$遍历一遍，就相当于把y的所有可能取值做了一次随机排列。

如果我们事先没有做正向运算，例如取$g=6, y=4$，那么6的多少次幂模13等于4呢？说白了，就是要求离散对数$x\equiv \mathrm{ind}_6 4 \pmod{13}$。

由于到目前位置都没有一个很好的求解离散对数的算法，只有正向一个一个的试才能找到那个满足条件的x。

例如$6^0 \bmod 13=1$，$6^1 \bmod 13=6$，$6^2 \bmod 13=10$，$6^3 \bmod 13=8$，$6^4 \bmod 13=9$，$6^5 \bmod 13=2$，$6^6 \bmod 13=12$，$6^7 \bmod 13=7$，$6^8 \bmod 13=3$，$6^9 \bmod 13=5$，$6^{10} \bmod 13=4$。

在这个例子中，我们共尝试了11次幂模运算才得到离散对数的结果。

如果是一个一般性的离散对数$x\equiv \mathrm{ind}_g y \pmod{n}$，如果运气好的话，做1次幂模运算就能得到结果；而运气不好的话要做$n-2$次幂模运算，平均情况下，大约需要做$\frac{n-1}{2}$次幂模运算。也就说一般情况幂模运算的正向和反向运算的时间复杂度之比是$1:\frac{n-1}{2}$。

就像RSA中需要取两个大素数p和q来确保$n=pq$的素因子分解是难题一样，在这里，我们也可以取一个大模数n，来确保离散对数求解是个难题。

基于幂模的公钥密码

由于幂模运算的单向性，离散对数和大整数素因子分解一样都是一种陷门函数，所以同样可以基于幂模运算设计一套公钥密码。由于在幂模运算$g^x \bmod n=y$中已知x容易求得y，而已知y难求得x，所以可以用g和n作为公共参数，x作为私钥，用y作为公钥。

DH密钥交换

例如公钥密码的提出者Diffie和Hellman，就利用幂模运算的交换律和单向性设计一种密钥协商机制——DH密钥交换。

假设通信双方Alice和Bob需要使用对称密码进行加密通信，对称密码所使用的密钥我们通常称为会话秘钥，那么可以用一下的DH密钥交换过程在**不安全**的信道上实现会话密钥的**安全**协商。

- **第一步，**双方协商公共参数g和n；
- **第二步，**Alice和Bob分别在0到$n-1$之间随机生成x_a, x_b，作为各自的私钥，这个私钥是要保密的；
- **第三步，**Alice和Bob分别计算$y_a=g^{x_a} \bmod n, y_b=g^{x_b} \bmod n$，作为各自的公钥；
- **第四步，**Alice和Bob将各自的公钥发送给对方；
- **第五步，**此时Alice收到了Bob发来的y_b，Bob收到了Alice发来的y_a，然后分别计算$k_a={y_b}^{x_a} \bmod n$和$k_b={y_a}^{x_b} \bmod n$，并分别使用k_a和k_b作为对称密码的密钥

正确性

这个过程看起来好复杂，但你一定会想，不是说好了要协商会话密钥的吗？这个k_a和k_b能确保一样吗？咱们可以来证明一下，很简单。

由于幂模运算满足交换律，所以

$$\begin{aligned} k_a &= {y_b}^{x_a} \bmod n \\ &= (g^{x_b} \bmod n)^{x_a} \bmod n \\ &= g^{x_a x_b} \bmod n \\ &= (g^{x_a} \bmod n)^{x_b} \bmod n \\ &= {y_a}^{x_b} \bmod n \\ &= k_b \end{aligned}$$

所以，他们协商出的会话密钥一定是相同的！

并且，由于x_a和x_b都是随机生成的，所以还确保了会话密钥的随机性。

安全性

你一定还会想到第二个问题，那就是为什么说这个密钥交换过程可以在**不安全**的信道**安全**密钥呢？

分析整个密钥交换过程，你会发现所有通过不安全信道明文传输的信息有

$$g, n, y_a, y_b \tag{7}$$

那么从这些公开的，或者说可能被攻击者窃听的信息有没有可能破解出最终的会话密钥k_a或k_b呢？

要想得到k_a，根据公式$k_a = {y_b}^{x_a} \bmod n$，已知$y_b$和$n$，还必须知道$x_a$，而$x_a$是Alice的私钥，是Alice保密的，直接获取是不可能的，公开信息跟x_a有关的只有$y_a = g^{x_a} \bmod n$，在这个等式中，y_a, g, n都是公开的，要求x_a，其实就是求解离散对数$x_a \equiv \mathrm{ind}_g y_a \pmod{n}$，而只要参数$n$和$g$选取恰当，就可以确保这个离散对数是极难求解的问题。

也就是说，只要参数n和g选取恰当，就可以确保交换获得密钥是安全的，不可能被破解的。

小结

本节课我们学习了一种新的单向陷门函数——幂模，以及基于幂模单向性的DH密钥交换。

下节课，我们将为大家介绍一种基于幂模单向性的数字签名方案——DSA。