第二讲 公钥密码——RSA

引言

通过上节课的学习，我们知道了货币数字化的切入点在与交易数字化，交易数字化需要解决的首要问题是账户和签名的数字化。账户和签名数字化问题的本质是唯一性和身份认证问题。

接下来，我们先解决身份认证问题，捎带着就把账户的唯一性问题解决了。

在上节课中，我们还提到解决身份认证问题的强有力工具是密码学。

没错，那就是密码学中的公钥密码。

公钥密码

1976年Diffie和Hellman发表的学术论文“New Directions in Cryptography”，第一次提出了公钥密码的概念，用于解决密钥分发和身份认证问题。

密钥分发问题

什么是密钥分发问题呢?

我们在日常生活中或多或少都使用过加密，比如你在发送一份隐私或秘密的文件时，为了防止泄密，你会给压缩包设置密码，然后把密码告诉接收文件的人，接收方在解压缩时再填入同样的密码就可以获得文件。这就是一个典型的对称加密场景。你在告诉对方密码的时候，有没有想过无论你是用电话、短信，或是微信，这些渠道安全吗？如果安全的话，干吗还要加密呢，直接把文件发过去不就行了吗？如果不安全，密码就不安全，密钥不安全，加密不就形同虚设了吗？这就是密钥分发问题。

公钥密码是怎么解决密钥分发问题的呢?

所谓公钥密码，又称非对称密码，顾名思义就是加密和解密使用的密钥不同，其中一个称为公钥，即可以公开的密钥；另一个称为私钥，即必须持有者保密的密钥；而且由公钥不可能计算出私钥。

任何人都可以生成自己的私钥和公钥，把公钥公开，把私钥自己保管。当别人想把加密文件的密码发送给我的时候，就可以用我的公钥对密码进行加密，然后发给我，然后我用我的私钥解密就可以拿到加密文件的密码。因为只有我拥有我的私钥，所以除了我之外的其他人，是不可能获得加密文件密码的。

如此一来，公钥密码就解决了密钥分发问题。

身份认证问题

咱们再来看身份认证，什么是身份认证呢?

现代网络通信，通信双方很可能远隔万里，那么你很难确认给你发消息的人就是他声称的那个人，所以你需要通过某种方式确认他的身份，这就是身份认证。

公钥密码是怎么解决身份认证问题的呢?

当我给别人发信息时，我可以用我的私钥对消息加密，将得到的密文和消息本身附在一起发送给对方，对方可以用我的公钥从密文中解密出明文，然后将明文与消息对比，如果一致就说明发送消息的人是我，不一致就说明发送消息的人不是我。为什么呢？因为只有拥有与我的公钥对应的私钥的人才可以做到这一点，而除了我之外，没有任何人拥有我的私钥。

总体来说，公钥加密，私钥解密可以保护消息机密性；私钥加密，公钥解密可以进行身份认证。

RSA

什么公钥、私钥，讲了这么多，它到底长什么样啊?

接下来我们介绍一种最为经典的公钥密码——RSA。

RSA数学基础

要理解RSA，我们要先从一些基本的数学开始。

整除

这是我们小学就学过的概念。

- 如果a整除b，记为$a|b$。

关于整除有一个很简单的结论：

- 若$c = k_1 a + k_2 b$，$e|a$，且$e|b$，则$e|c$。

最大公因子

所有同时整除a和b的整数中，最大的那个，称为a和b的最大公因子，记为(a, b)

根据这个结论，对任意正整数a和b，首先a一定可以表示成$a = kb + c, 0 \leq c < b$，也就是说$k = \frac{a}{b}$，$a \mod b = c$。

其次$(a, b) = (b, c)$，也就是说，a和b的最大公因子，b和c的最大公因子，是相同的。

为什么呢，由于$a = kb + c$，所以所有b和c的公因子同时整除b和c，所以也能整除a，所以也是a和b的公因子。

由于$c = a - kb$，所以所有a和b的公因子同时整除a和b，所以也能整除c，所以也是b和c的公因子。

也就是说，a和b的公因子，b和c的公因子是同一拨，那么a和b的公因子中最大的那个，b和c的公因子中最大的那个当然是同一个了。

所以$(a, b) = (b, c)$。

欧几里得算法

有了这个结论，当我们要求两个整数的最大公因子时，就可以根据这个结论迭代的求取，这就是著名的欧几里得算法。

```python
def gcd(a, b):
    if b == 0:
        return a
    return gcd(b, a % b)

```

扩展欧几里得算法

欧几里得算法还有一种高级用法，即在求得a和b的最大公因子的公式，求出两个系数k_1和k_2，使得$k_1a+k_2b=(a,b)$，只需在欧几里得算法的基础上扩展一下，这就是扩展欧几里得算法

```python
def egcd(a, b):
    if b == 0:
        return a, 1, 0
    gcd, k1, k2 = egcd(b, a % b)
    return gcd, k2, k1 - a / b * k2

```

互素

最大公因子的最小可能取值是1，当$(a,b)=1$，即a和b的最大公因子为1时，我们称a和b互素。

乘法逆元

当$(a,b)=1$时，有时候我们很希望求得一个数$k, 0 \le k < b$，使$ka \bmod b = 1$，这样的数我们称为a的乘法逆元，起始这看起来就像是在0到$b-1$这些整数中找到a的倒数一样。那怎么找到这样的数呢？

扩展欧几里得算法可以帮我们解决这个问题。

由于$(a,b)=1$，根据扩展欧几里得算法，可求得两个系数k_1和k_2，使得$k_1a+k_2b=(a,b)=1$，所以有$k_1a=-k_2b+1$，所以$k_1a \bmod b = 1$，所以$(k_1 \bmod b)a \mod b = 1$，而$0 \le (k_1 \bmod b) < b$，所以$(k_1 \bmod b)$就是我们想要的那个乘法逆元。

欧拉函数

对任意一个正整数n，在1到n的这n个整数里，显然有些和n是互素的，而有些和n是不互素的，那些和n互素的整数的数量就是n的欧拉函数，记作$\varphi(n)$。那么$\varphi(n)$该怎么计算呢？

我们都知道任意整数n都可以表示成它的所有素因子的乘积：

$$n = p_1^{l_1} p_2^{l_2} \dots p_s^{l_s} \tag{1}$$

所以所有那些和n不互素的数，一定和n有其中某个素因子作为公共因子。

所以我们只要从1到n中的所有整数中，是$p_1, p_2, \dots, p_s$的倍数的依次剔除，剩下的就是与n互素的数。

例如，p_1的倍数一共有多少个呢，由于p_1的倍数在1到n中是均匀分布的，所以占据的比例是$\frac{1}{p_1}$，剔除p_1的倍数后，还剩下$n(1-\frac{1}{p_1})$个；在剩下的数中，由于p_2的倍数在1到n中也是均匀分布的，所以占据的比例是$\frac{1}{p_2}$，所以再剔除p_2的倍数后，剩下$n(1-\frac{1}{p_1})(1-\frac{1}{p_2})$个。以此类推，当把所有素因子的整数倍都剔除后，剩下的数共有$n(1-\frac{1}{p_1})(1-\frac{1}{p_2})\dots(1-\frac{1}{p_s})$个。即

$$\varphi(n) = n\prod_{i=1}^{s}(1-\frac{1}{p_s}) \tag{2}$$

由此可见，求欧拉函数的关键在于求出n的的所有素因子，即对n做素因子分解。

有一种特殊情况，n为素数，那么n仅有一个素因子，即它自己。此时$\varphi(n) = n(1-\frac{1}{n}) = n-1$。

还有一种特殊情况，n仅有两个素因子，即$n = pq$，那么$\varphi(n) = pq(1-\frac{1}{p})(1-\frac{1}{q}) = (p-1)(q-1)$。如果已知$p$和$q$，显然$\varphi(n)$是好求的；而如果仅知道$n$，而不知道$p$和$q$，那么必须要先对$n$做素因子分解，得到$p$和$q$，才能求得$\varphi(n)$。

如果这两个素因子p, q都极大，那么当然n也就极大。要从1到n这n个数中找出这两个素因子，就如同大海捞针，复杂度极高。

欧拉定理

现在要进入最精彩的一个部分——欧拉定理了。欧拉定理是RSA所依赖的直接数学基础。

我们先给出欧拉定理的结论，再予以证明。

当$(a, n) = 1$时，

$$a^{\varphi(n)} \equiv 1 \pmod n \tag{3}$$

也就是说，当a和n互素时，$\varphi(n)$个a相乘再模n等于1。

我们先把在1到n之间，与n互素的那$\varphi(n)$个数都列出来，将它们相乘，然后模n，用符号表示如下：

$$P = x_1, x_2, \ldots, x_{\varphi(n)} \bmod n \tag{4}$$

考虑其中人一个x_i，由于它和n互素，而a也和n互素，所以ax_i也和n互素，所以$ax_i \bmod n$也和n互素。

考虑任何两个x_i, x_j，一定有$ax_i \bmod n \neq ax_j \bmod n$。因为若$ax_i \bmod n = ax_j \bmod n$，则$n|a(x_i - x_j)$，由于$a$和$n$互素，所以$n|x_i - x_j$，所以$x_i \equiv x_j \pmod n$，由于$x_i$和$x_j$各不相同，所以$x_i \equiv x_j \pmod n$不可能，所以$ax_i \bmod n \neq ax_j \bmod n$。

也就是说，下列这些元素都与n互素，且两两各不相等。

$$ax_1 \bmod n, ax_2 \bmod n, \ldots, ax_{\varphi(n)} \bmod n \tag{5}$$

所以这些元素和$x_1, x_2, \ldots, x_{\varphi(n)}$是同一拨。

所以如果将这些元素相乘，应当和P，即

$$\begin{aligned} x_1x_2\cdots x_{\varphi(n)} \bmod n &= (ax_1 \bmod n)(ax_2 \bmod n)\cdots(ax_{\varphi(n)} \bmod n) \\ &= ax_1\dot{a}x_2\cdots ax_{\varphi(n)} \bmod n \\ &= a^{\varphi(n)}x_1x_2\cdots x_{\varphi(n)} \bmod n \end{aligned} \tag{6}$$

由于$x_1, x_2, \ldots, x_{\varphi(n)}$通通与$n$互素，等式两边可以约去，从而得到

$$a^{\varphi(n)} \equiv 1 \pmod n \tag{7}$$

欧拉定理推论

由欧拉定理，显然有，对任意整数k，

$$a^{k\varphi(n)} \equiv 1 \pmod n \tag{8}$$

显然也有

$$a^{k\varphi(n)+1} \equiv a \pmod n \tag{9}$$

这个结论就有意思了，a经过若干次幂再模n后又等于n，如果我们能把这个操作拆成两步，第一步不就是相当于加密，第二步不就相当于是解密了吗？

那怎么拆呢？如果我们能找到两个数e和d，是的$ed = k\varphi(n) + 1$，不就可以得到

$$a^{ed} \bmod n = (a^e \bmod n)^d \bmod n \tag{10}$$

这就相当于做e次幂模n是加密，做d次幂模n是解密。

哇哦！如果e和d不相等，这不就是一种非对称密码吗，加密和解密使用不同的密钥。

可是，怎样找到这样的e和d呢？

我们希望找到的e和d满足$ed = k\varphi(n) + 1$，其实就是满足

$$ed \bmod \varphi(n) = 1 \tag{11}$$

等等，这个式子好像似曾相识哦！对，这不就是说e和d互为乘法逆元吗！

因此，我们只需要先找一个e，使得e和$\varphi(n)$互素，即$(e, \varphi(n)) = 1$，然后再用扩展欧几里得算法求出d不就行啦！

等等，还差一点，还记得我们一开始介绍公钥密码时讲过，公钥密码除了机密和解密的密钥不同之外，还要求除了私钥的拥有者之外的其他人，无法由公钥求出私钥。那怎样才能让已知公钥不能求出私钥呢？

无论是由e求d或者由d求e，其关键都在于等式$ed \bmod \varphi(n) = 1$，如果私钥的拥有者能计算出$\varphi(n)$，而其他人不能求出$\varphi(n)$，不就行了吗？

那怎么才能做到这一点呢？还记得我们在介绍欧拉函数时所讲的那个特殊情况了吗？n等于两个大素数的乘积，那么知道这两个大素数p, q的人很容易求得欧拉函数，而仅知道n，而不知道那两个素因子p, q的人就很难求得欧拉函数！

RSA

咱们介绍了这么多数学基础，现在要正式进入RSA了。

RSA密钥生成

首先，我们要讲RSA密钥n, e, d的生成。

第一步，随机生成两个大的素数p和q；

第二步，计算$n = pq$，$\varphi(n) = (p-1)(q-1)$，销毁p和q；

第三步，随机生成e，满足$(e, \varphi(n)) = 1$；

第四步，计算d，满足$ed \bmod \varphi(n) = 1$。

常称n, e为公钥，n, d为私钥。

显然，由于对n做素因子分解很难，所以由公钥计算出私钥也很难。

RSA加解密

用RSA怎样进行加解密呢，其实在前面的数学基础中我们已经介绍的很清楚了。我们可以任选公钥或私钥进行加密，然后用另一个解密。例如，如果明文是p，可以用下面的公式进行加密得到密文c。

$$c = p^e \bmod n \tag{12}$$

用下面的公式进行解密。

$$p = c^d \bmod n \tag{13}$$

有了前面的数学基础，加解密的正确性应该是不言而喻的了。

密钥分发

若要保护对称加密算法密钥的机密性，那么发送方可用接收方的公钥对消息加密得到密文，将密文发送给接收方，接收方用自己的私钥解密获得消息明文。除了接收方之外的其它人由于不知道私钥，所以不可能破解出消息。

身份认证

若要进行身份认证，则发送方用自己的私钥对消息加密，并将得到的密文附于明文之后一同发送给接收方，接收方用发送方的公钥解密，将解密得到的消息与明文消息对比，若一致则认为消息来自于真实的发送方，否则认为消息并非来自真实发送方。这里所说的密文就是所谓的**数字签名**。其实这么说并不太准确，由于消息可能比较长，所以发送方加密的不是消息本身，而是消息的哈希值。哈希值又是什么呢？这等到课程的第三单元我们再详细介绍。

交易数字化

签名数字化——身份认证

在身份认证中，我们可以看出，公钥的作用是公开向外界别标榜自己的身份，而私钥的作用是对消息生成数字签名，用数字签名向外界证明自己的身份与公钥所标榜的身份相符。如此一来，我们只要用公钥作为账户，用私钥对交易信息生成数字签名，不就解决了交易中账户和签名的问题了吗。

没错，任何一个用户都可以按照下面的步骤生成账户和交易。

- **第一步，**用RSA生成公钥$< n, e >$和私钥$< n, d >$，用公钥$< n, e >$作为自己的账号；
- **第二步，**填写交易信息T

字段	值
转出账户	自己的公钥$< n, e >$
转入账户	收款人的公钥$< n', e' >$
金额	1000.00
其它（时间、说明等）	略
数字签名	暂时不填

- **第三步，**用自己的私钥对交易信息T生成数字签名$s = (H(T))^d \bmod n$，其中H表示一种哈希函数，将s加入交易信息，得到最终的交易信息。

字段	值
转出账户	自己的公钥$< n, e >$
转入账户	收款人的公钥$< n', e' >$
金额	1000.00
其它（时间、说明等）	略
数字签名	s

用户创建完交易信息T，就可以把它公告给其它用户，试图得到其它用户的认可。收到该消息的任何用户都可以用转出账户（创建交易人的公钥）和数字签名（创建交易人用自己的私钥生成的）来验证该用户是不是该账户的拥有者，有没有权力从该账户转账。

账户数字化——唯一性

等等，交易数字化问题就这么解决了？

虽说我们用公钥密码解决签名数字化问题，可是这个过程中压根没提账户唯一性问题啊！

区块链确实没有通过任何机制检验新账户的唯一性。

可是，别忘了，我们使用什么作为账户标识的？

公钥！

公钥是怎么来的？

以RSA的密钥生成过程为例。

如果素数p和q都取1024位，那么n和$\varphi(n)$就都大约是2048位；

若e取随机值，由于$ed \bmod \varphi(n) = 1$，那么d也是随机值，大约也是2048位。

所以公钥$< e, d >$大约是4096位的随机值；

也就是说，我们使用4096位的随机值作为账户标识的；

所以，出现两个账户标识相同的概率是$\frac{1}{2^{4096}}$；

哇，这是什么概念？

我们死于小行星撞击地球的概率是7481万分之一。

所以，尽管尽管我们没有强制要求账户标识唯一，但公钥的随机性使得账户标识几乎不可能出现重合。

小结

本节课，为了解决账户唯一性和身份认证为问题，学习了公钥密码。

知道了公钥密码可以用于密码分发和身份认证。

当用公钥作为账号时，捎带着就解决了账户唯一性问题。

我们还着重学习了公钥密码的最典型代表RSA的原理，和怎样用RSA实现交易数字化。

在接下来的几节课中，我们再沿着公钥密码进一步拓展，为大家介绍基于离散对数的公钥密码、DH密钥交换、DSA和椭圆曲线密码知识。