第四讲 DSA

引言

通过前面两讲，我们了解了公钥密码在保护机密性和身份认证方面的一般，还了解了除了大整数素因子分解之外另一个单向陷门函数——离散对数。这一讲中，我们深入学习一种标准数字签名算法——DSA。

以比特币为代表的数字货币都是建立在一个去中心化的P2P网络之上的，在这个网络中，任何用户都可以创建和传播交易信息，交易信息得到大部分用户确认验证后，最终会由矿工记录在一个分布式的总账本，也就是区块链中。

因此，比特币的信任体系完全建立在一个个用户个体对交易信息可靠确认的基础之上。

问题来了，假设你是一个比特币用户，你收到了一笔交易信息，这笔交易信息描述了某人要从账户A转账5比特币到B账户。

你会信任这笔交易并把它传播给其他用户吗？

当然不会！

因为你会怀疑这个人有没有权利从A账户中转账？这个人将来会不会抵赖？

反过来，如果你需要创建一笔交易，那你该怎样做才能其他用户的信任呢？

与数字签名有关的信息

在现实中，当你用支票支付时，你得在支票上签署你的名字，才能让这张支票具备效力。

同样的道理，你必须在数字化的交易信息上附加数字化的签名，才能让这笔交易信息具备效力，才能得到其他用户的信任。

那什么样的数字才具备签名的效力呢。

第一，支票签名具有签署者独一无二的风格，因此数字签名必须依赖于签署者独有的秘密数字x。

第二，不同的支票上签名都是不同的，因此数字签名必须依赖于签署的消息m，为了方便处理，我们用消息的哈希值$H(m)$代替它。

第三，即便是相同的支票，每一次签名也都具有一定的随机性，因此数字签名也必须依赖于签署者独有的随机数k。

支票签名	数字签名
签名具有签署者独一无二的风格	必须依赖于签署着独有的秘密数字x
支票不同，签名不同	必须依赖于消息m的哈希值$H(m)$
签名具有随机性	必须依赖于签署者独有的随机数字k

数字签名方案初探

私钥泄露问题

用数学公式表示的话，签名s应当是秘密数字x，消息的哈希值$H(m)$和随机数字的一个k函数。

$$s = f(x, H(m), k) \tag{1}$$

现在，如果你收到了一条消息m和对应的签名s，你可以通过验证签名真伪来决定是否信任这条消息。

显然，你只需要验证这个等式$s = f(x, H(m), k)$是否成立就可以啦！

成立就是真的，不成立就是假的。

真的这么简单吗？

等等！秘密数字x和随机数字k都是签署者独有的哦！

你作为接收方并不知道x和k，所以你无法验证等式$s = f(x, H(m), k)$是否成立。

我们好像走进了一个死胡同。

签署者如果不发送x和k，接收方就无法验证签名；

签署者如果发送x和k，那么就泄露了秘密数字，而任何知晓x和k的人都可以随意伪造他的签名。

那怎么办呢？

解决技巧

上一讲中的离散对数问题为了我们提供了思路。

若整数x, y, g和素数p满足$g^x \mod p = y$，那么当p是一个很大的素数时，已知x很容易求得y。但反过来，已知y时却极难求得x。

有了离散对数问题的这一优良特性，签署者就可以放心大胆的将$y = g^x \mod p$和$r = g^k \mod p$随消息m和签名s一起发送给对方，而不必担心其独有的x和k泄露给接收方。

在这些数字里，我们常称x为私钥，称y为公钥，称$< r, s >$为数字签名。

接收方没有x和k，那么该怎样验证签名有效性呢？

鉴于x, k, m, s之间有关系$F: s = f(x, H(m), k)$，而且，x和y，r和k之间都有指数和幂的关系：

$$\begin{aligned} y &= g^x \mod p \\ r &= g^k \mod p \end{aligned} \tag{2}$$

所以r, s, y, m这四个公开的数字之间必然存在某种关系G，那么接收方只要验证关系G是否成立就可以验证签名是否有效了。

显然签名关系F和签名函数f是等价的，而验签关系G是由F决定的。因此关系F是决定签名方案的关键。

中学阶段我们就学习过指数和幂的性质。如果指数间做加法运算，那么对应的幂之间就有乘法运算。

所以，我们最容易想到的一种方案是让F中各数字之间有加法运算，例如

$$s = x + k + H(m) \mod (p - 1) \tag{1}$$

对应的，G应当为

$$\begin{aligned} g^s &\equiv g^{x+k+H(m)} \pmod{p} \\ &\equiv g^x \cdot g^k \cdot g^{H(m)} \pmod{p} \\ &\equiv y \cdot r \cdot g^{H(m)} \pmod{p} \end{aligned} \tag{2}$$

细心的同学可能会发现，公式(1)和公式(2)里有些微妙的变化。

- **第一**，=变成了≡。解释一下，=仅表示等于关系，而≡表示同余关系，例如$g^s \equiv y \cdot r \cdot g^{H(m)} \pmod{p}$表示$g^s \mod p = y \cdot r \cdot g^{H(m)} \mod p$。
- **第二**，模数从$p - 1$变成p了。根据欧拉定理，由于g和p必互素，所以$g^{p-1} \mod p = 1$，所以当指数模$p - 1$同余时，幂模p也应当同余。

也就是说，接收方只需要验证关系

$$G: g^s \equiv y \cdot r \cdot g^{H(m)} \pmod{p} \tag{3}$$

是否成立就可以验证签名了。

数字签名方案再探

签名伪造问题

这样就可以了吗？对不起，这又是一个坑，这个方案还是不行。根据我们的设计方案，正常的签名一定能通过验签方案。

而通过验签方案的签名一定是正常的吗？不一定。

攻击者很容易在不知晓x和k的情况下，伪造签名r和s。

怎么伪造呢？

攻击者只要先对s任取一值s'，然后通过解方程

$$g^{s} \equiv y \cdot r \cdot g^{H(m)} \pmod{p} \tag{4}$$

得到

$$r' \equiv \frac{g^{s'}}{y \cdot g^{H(m)}} \pmod{p} \tag{3}$$

显然，满足公式(3)的r', s'也一定能通过验签方案。也就是说攻击者可以在不知道私钥的情况下对消息m伪造签名。

解决技巧

这个漏洞显然是由验签方案不安全导致的。怎样才能让验签方案更安全呢？

回想离散对数问题的特性，如果我们让验签方案中的s和r都出现在指数上，攻击者如果要先取其中一个，再求解另一个，就必须解决离散对数问题，而这是极难的。

验签关系G是由签名关系F确定的，要调整G，就要要调整F。

这里需要用到一个小技巧，我们只要让F中的r和s分别与x和k相乘再取和，那么G中的r和s就会自动跑到指数的位置上。

比如，我们可以将签名关系F修改为

$$F: sk \equiv H(m) + xr \pmod{(p-1)} \tag{5}$$

对应的，签名函数修改为

$$s = \frac{H(m) + xr}{k} \bmod (p-1) \tag{6}$$

类似的，我们也可以得到

$$k = \frac{H(m) + xr}{s} \bmod (p-1) \tag{7}$$

因此有

$$
\begin{aligned}
r &= g^k \bmod p && (8)\\
&= g^{\frac{H(m)+xr}{s}} \bmod p && (9)\\
&= g^{\frac{H(m)}{s}} \cdot g^{\frac{xr}{s}} \bmod p && (10)\\
&= g^{\frac{H(m)}{s}} \cdot y^{\frac{r}{s}} \bmod p && (11)
\end{aligned}
$$

在等式$r = g^{\frac{H(m)}{s}} \cdot y^{\frac{r}{s}} \bmod p$中仅包含签名$< r, s >$、公钥$y$和消息$m$，不会泄露私钥，因此我们可用该等式作为验签关系：

$$
G : r = g^{\frac{H(m)}{s}} \cdot y^{\frac{r}{s}} \bmod p \quad (12)
$$

显然G中r和s都跑到了指数上，攻击者在不知道私钥x和m的情况下就不可能伪造签名了，除非他能解决离散对数问题。

至此，我们已经得到了著名的数字签名方案DSA的雏形了。即以

$$
\begin{aligned}
r &= g^k \bmod p && (13)\\
s &= \frac{H(m) + xr}{k} \bmod (p-1) && (14)
\end{aligned}
$$

为数字签名；

以关系

$$
G : r = g^{\frac{H(m)}{s}} \cdot y^{\frac{r}{s}} \bmod p \quad (15)
$$

为验签方案。

数字签名方案再再探

签名规模问题

为了确保离散对数问题不被暴力破解，通常素数p需要取到1024比特以上。

由于r和s分别是由模p和p得来的，所以r和s的规模大致与p相同。

所以签名r和s大约需要2048比特。

解决技巧

为了减小数字签名的规模，DSA采取这样的策略：

选择一个相对较小的整数q，让q满足

$$g^q \bmod p = 1 \tag{16}$$

如果p取1024比特，那么q就取160比特。

有了这样的q，便可将签名方案中的

$$s = \frac{H(m) + xr}{k} \bmod (p-1) \tag{17}$$

模p改为模q:

$$s = \frac{H(m) + xr}{k} \bmod q \tag{18}$$

由于$g^q \bmod p = 1$，所以并不影响眼前方案中的

$$r = g^{\frac{H(m)}{s}} \cdot y^{\frac{r}{s}} \bmod p \tag{19}$$

如此一来，s缩小到与q同等规模，而r大小却没变。

为了将r也缩小至同等规模，DSA将签名方案中的$r = g^k \bmod p$在模p的基础上再模q:

$$r = \left(g^k \bmod p\right) \bmod q \tag{20}$$

此时，验签方案就要同步修改成

$$r = \left(g^{\frac{H(m)}{s}} \cdot y^{\frac{r}{s}} \bmod p\right) \bmod q \tag{21}$$

如此一来，签名r和s就都缩小到了与q同等规模。

至此，我们得到了真正的DSA数字签名方案。

DSA数字签名方案

签名过程

- 第一步，生成参数素数p，素数q，底数g，满足$g^q \bmod p = 1$;
- 第二步，对任意消息m，生成随机数$k, 1 < k < q$;
- 第三步，计算$r = \left(g^k \bmod p\right) \bmod q$，若$r = 0$，重复该步骤;
- 第四步，计算$s = \frac{H(m)+xr}{k} \mod q$，若$s = 0$，重复该步骤;
- 第五步，令$< r, s >$为数字签名

验签过程

检查下面两个条件是否同时成立，若同时成立则认为签名有效，否则认为签名无效。

- $0 < r < q$；
- $0 < s < q$；
- $r = \left(g^{\frac{H(m)}{s}} \cdot y^{\frac{r}{s}} \bmod p\right) \bmod q$。

小结

回顾我们的DSA探秘之路，有以下三个要点：

第一，为了防止泄露私钥x，只能公开公钥y；

第二，为了防止伪造签名，必须在验签方案中让签名出现在指数上；

第三，为了减小数字签名规模，为指数选择一个相对较小的模数。

而这一切都建立在素数域$GF(p)$中的整数加法、乘法、幂和离散对数的运算性质上。

如果将素数域更换为有限域上的椭圆曲线，运算的对象变为椭圆曲线上的点，那么仍然可以定义出具有类似性质的加法、乘法。参照我们的DSA探秘之路，你自己就可以设计出具有更高安全性的椭圆曲线数字签名方案ECDSA，而ECDSA正是比特币体系使用的数字签名方案。

等等，有限域上的椭圆曲线是什么？下节课继续为大家讲述。