Segwit地址

Segwit地址又称隔离见证地址。在比特币区块链上，经常可以看到类似`bc1qmy63mjadtw8nhzl69ukdepwzsyvv4yex5qlmkd`这样的以`bc`开头的地址，这种地址就是隔离见证地址。

Segwit地址有好几种，一种是以`3`开头的隔离见证兼容地址（Nested Segwit Address），从该地址上无法区分到底是多签地址还是隔离见证兼容地址，好处是钱包程序不用修改，可直接付款到该地址。

另一种是原生隔离见证地址（Native Segwit Address），即以`bc`开头的地址，它本质上就是一种新的编码方式。

我们回顾一下`1`开头的比特币地址是如何创建的：

1. 根据公钥计算hash160；
2. 添加固定头并计算带校验的Base58编码。

简单地概括就是使用Base58编码的公钥哈希。

而`bc`地址使用的不是Base58编码，而是Bech32编码，它的算法是：

1. 根据公钥计算hash160；
2. 使用Base32编码得到更长的编码；
3. 以`bc`作为识别码进行编码并带校验。

Bech32编码实际上由两部分组成：一部分是`bc`这样的前缀，被称为HRP（Human Readable Part，用户可读部分），另一部分是特殊的Base32编码，使用字母表`qpzry9x8gf2tvdw0s3jn54khce6mua7l`，中间用`1`连接。对一个公钥进行Bech32编码的代码如下：

```
const
    bitcoin = require('bitcoinjs-lib'),
    bech32 = require('bech32'),
    createHash = require('create-hash');
```

```
// 压缩的公钥:
let publicKey =
'02d0de0aaeaefad02b8bdc8a01a1b8b11c696bd3d66a2c5f10780d95b7df42645c';

// 计算hash160:
let
    sha256 = createHash('sha256'),
    ripemd160 = createHash('ripemd160'),
    hash256 = sha256.update(Buffer.from(publicKey, 'hex')).digest(),
    hash160 = ripemd160.update(hash256).digest();

// 计算bech32编码:
let words = bech32.toWords(hash160);
// 头部添加版本号0x00:
words.unshift(0);

// 对地址编码:
let address = bech32.encode('bc', words);
console.log(address); // bc1qmy63mjadtw8nhzl69ukdepwzsyvv4yex5qlmkd
```

```
bc1qmy63mjadtw8nhzl69ukdepwzsyvv4yex5qlmkd
```

和Base58地址相比，Bech32地址的优点有：

1. 不用区分大小写，因为编码用的字符表没有大写字母；
2. 有个固定前缀，可任意设置，便于识别；
3. 生成的二维码更小。

它的缺点是：

1. 和现有地址不兼容，钱包程序必须升级；
2. 使用`1`作为分隔符，却使用了字母`l`，容易混淆；
3. 地址更长，有42个字符。

那为什么要引入Segwit地址呢？按照官方说法，它的目的是为了解决比特币交易的延展性（Transaction Malleability）攻击。

延展性攻击

什么是延展性攻击呢？我们先回顾一下比特币的区块链如何保证一个交易有效并且不被修改：

1. 每个交易都必须签名才能花费输入（UTXO）；
2. 所有交易的哈希以Merkle Tree计算并存储到区块头。

我们再看每个交易的细节，假设有一个输入和一个输出，它类似：

```
tx = ... input#index ... signature ... output-script ...
```

而整个交易的哈希可直接根据交易本身计算：

```
tx-hash = dhash(tx)
```

因为只有私钥持有人才能正确地签名，所以，只要签名是有效的，tx本身就应该固定下来。

但问题出在ECDSA签名算法上。ECDSA签名算法基于私钥计算的签名实际上是两个整数，记作`(r, s)`，但由于椭圆曲线的对称性，`(r, -s mod N)`实际上也是一个有效的签名（N是椭圆曲线的固定参数之一）。换句话说，对某个交易进行签名，总是可以计算出两个有效的签名，并且这两个有效的签名还可以互相计算出来。

黑客可以在某一笔交易发出但并未落块的时间内，对签名进行修改，使之仍是一个有效的交易。注意黑客并无法修改任何输入输出的地址和金额，仅能修改签名。但由于签名的修改，使得整个交易的哈希被改变了。如果修改后的交易先被打包，虽然原始交易会被丢弃，且并不影响交易安全，但这个延展性攻击可用于攻击交易所。

要解决延展性攻击的问题，有两个办法，一是对交易签名进行归一化（Normalize）。因为ECDSA签名后总有两个有效的签名`(r, s)`和`(r, -s mod N)`，那只接受数值较小的那个签名，为此比特币引入了一个`SCRIPT_VERIFY_LOW_S`标志仅接受较小值的签名。

另一个办法是把签名数据移到交易之外，这样交易本身的哈希就不会变化。不含签名的交易计算出的哈希称为wtxid，为此引入了一种新的隔离见证地址。

小结

以`bc`开头的隔离见证地址使用了Bech32编码；

比特币延展性攻击的原因是ECDSA签名总是有两个有效签名，且可以相互计算；

规范ECDSA签名格式可强制使用固定签名（例如总是使用较小值的签名）。