09 比特币脚本

比特币脚本是Stack Based language，每个脚本的内存空间就是一个堆栈。

交易id

交易的哈希值

交易使用的协议版本号

交易的大小

交易的生效等待的时间，0就是立刻生效

输入

输出

交易所在的区块的哈希值

当前交易已经经过了多少个确认

交易产生的时间

区块产生的时间

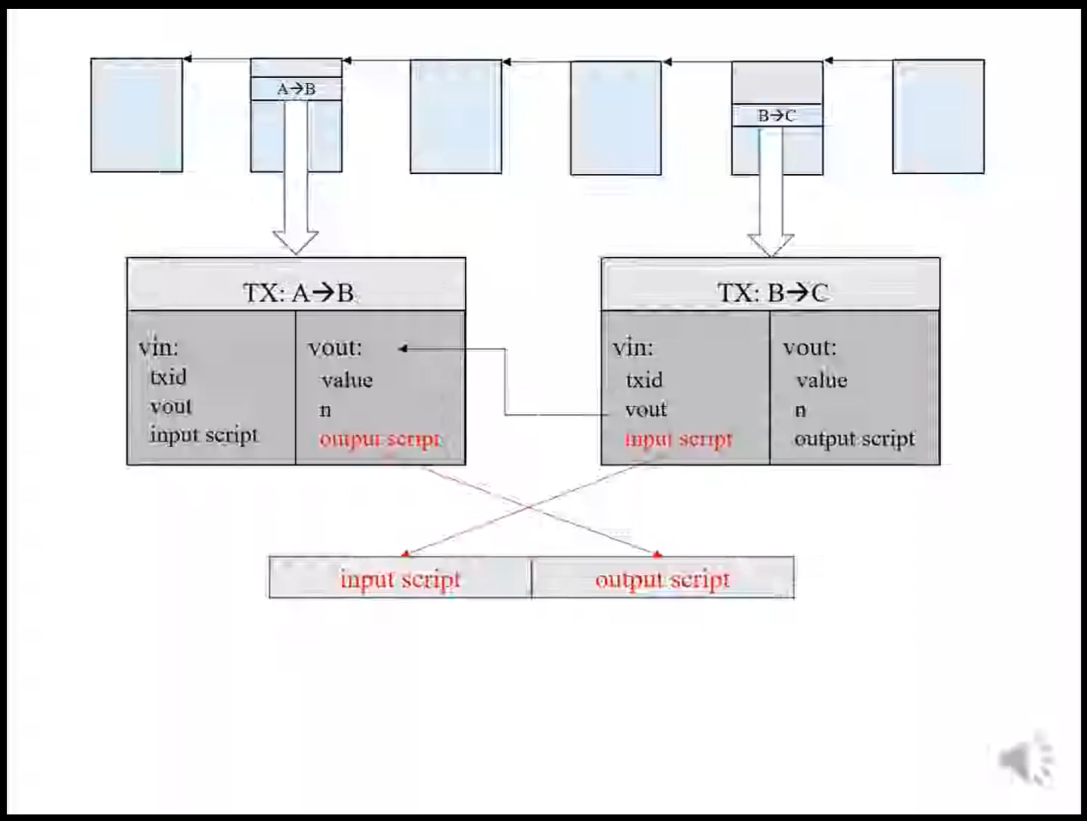
vin里面需要说明当前交易花的比特币的来源，txid指定的是币是来源于哪个交易，vout说明的是用的txid交易中的第几个output

交易的输出


```
"vout": [{
    "value": 0.22684000,
    "n": 0,
    "scriptPubKey": {
        "asm": "DUP HASH160 628e…d743 EQUALVERIFY CHECKSIG",
        "hex": "76a9…88ac",
        "reqSigs": 1,
        "type": "pubkeyhash",
        "addresses": [ "19z8LJkNXLrTv2QK5jgTncJCGUEEfpQvSr"]
    }
  },{
    "value": 0.53756644,
    "n": 1,
    "scriptPubKey": {
        "asm": "DUP HASH160 da7d…2cd2 EQUALVERIFY CHECKSIG",
        "hex": "76a9…88ac",
        "reqSigs": 1,
        "type": "pubkeyhash",
        "addresses": ["1LvGTpdyeVLcLCDK2m9f7Pbh7zwhs7NYhX"]
    }
 }],
```


这里的"scriptPubKey"之后将以output script指代

vout中需要说明输出的金额value，n是第几个输出，scriptPubKey就是输出脚本，就是一个public key。reqsigs是需要多少个签名。type是类型，这里是公钥哈希。然后address是输出的地址。



第二个交易需要验证币的来源，也就是第一个交易的输出。在验证的时候需要讲第一个交易的输出，和第二个交易的输入脚本交叉之后拼接在一起。如果能从头到尾的正确运行，那么交易就是合法的。现在的比特币中出于安全的角度考虑，改成分别运行了。如果没有任何的问题就证明交易是合法的。

最简单的输入输出脚本就是P2PK pay to public key

input script: signature 用当前交易付款人（上个交易收款人）的私钥对整个交易进行加密

output script：是收款人的公钥。（⚠️币的来源的那个交易的output，也就是当前交易的付款人的公钥）

这样的话一对公私钥对就全都有了，就可以进行验证了。

第二种形式就是P2PKH pay to public Key Hash 这是现实中最常用的

input script: 当前交易付款人的签名和当前交易付款人的公钥

Output script：上一个交易收款人（当前交易的付款人）的公钥哈希

最复杂的形式是P2SH (Pay to Script Hash)

input中需要给出签名和赎回脚本的具体内容。

output script中给出的是一个脚本的哈希，redeemScriptHash，赎回脚本哈希。花钱的时候需要给出赎回脚本的内容以及需要给出正确运行赎回脚本的签名。

验证的时候需要两步：

1. 验证序列化的赎回脚本的具体内容是否与赎回脚本的哈希一致
2. 反序列化并执行赎回脚本，验证输入脚本中给出的签名是否正确

赎回脚本的形式可以是P2PK形式，P2PKH形式，或者多重签名形式

这个东西虽然非常复杂，但是可以支持多重签名的形式。

最早的多重签名，现在已经不推荐使用

input script: 输入脚本中只需要提供N个公钥中的M个签名就可以把钱转走

output script：给出N个公钥，以及一个阈值M

CheckMultiSig的代码实现有bug，pop的时候会多pop一个元素，所以为了迁就他，就多push进去一个没用的元素。

这样的多重签名没有用到P2SH，虽然没有安全问题但是不是很user friendly。原因在于比如说你给一个公司转钱，在output中需要说明收款人的信息，如果对方使用多重签名（比如说有五个合伙人，需要N=5 M=3），这些信息都需要付款的客户写在交易的output里。这对于用户来说不是很方便。解决这个问题的办法就是用P2SH。这样付款人只需要填收款人赎回脚本的哈希值。收款人再花这笔钱的时候才需要填多人的签名。

现在的多人签名都是采用P2SH。

在output中使用return函数可以销毁比特币，因为RETURN函数的返回内容是return zero or more ops\text就是永远都返回错误，这样这个币就再也花不出去了。

销毁比特币：

1. 有一些小的币种，称之为AltCoin Alternative Coin，要求必须要销毁一定数量的比特币才能获得。
2. 因为区块链是一个无法修改的账本，所有有些人通过这样的方式就是为了往区块链里写入一些永久保存的内容。比如说我们第一节课中讲过的digital commitment。要证明在某个时间知道某个事情。可以把需要记录的内容取哈希之后放到RETURN函数返回的zero or的后面。将来需要查验的时候就可以证明。

问题是为什么不往之前的那种coinbase transaction中的coinbase域里写入需要证明的内容呢？ 因为往coinbase里写需要获得记账权，而销毁比特币是不需要记账权的。事实上很多这样的交易都没有真正销毁比特币，而只花费了交易费。而这样做的好处就是写了RETURN之后，UTXO就知道不需要保存这个交易信息，对全节点的维护来说是十分友好的。