深入理解比特币脚本

清源
区块链工程师
关注他

2 人赞同了该文章

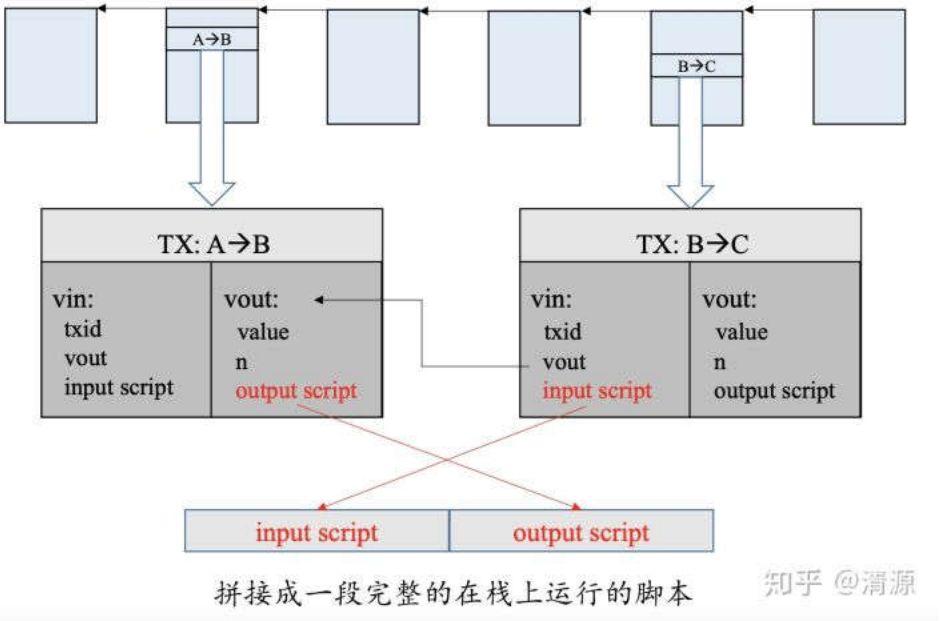
每一笔交易除了铸币交易（coinbase）外，每一笔交易都拥有至少一个输入（TxIn）和至少一个输出（TxOut），和我们直觉上理解的交易的TxIn和TxOut应该是数字不太一样，在比特币中是以脚本的形式存在。

比特币脚本

比特币脚本是一种基于栈的脚本语言，不是图灵完备的，在比特币没有账户的概念，谁拥有这笔交易的输出谁就可以花费这笔交易中的比特币，为了证明拥有这笔交易的输出就需要提供密钥接受验证，验证通过就可以花费这笔交易的输出。

> 其基本的设计思路是，谁能提供一个签名和一个公钥，让这个脚本运行通过，谁就能花费交易中包含的BTC。

执行的脚本由输入和输出拼接而成，如下图所示；



比特币提供了三种输入输入脚本的形式，分别是 `Pay to Publish Key` ， `Pay to Publish Key Hash` 和 `Pay to Script Hash` 。

公钥支付（Pay to Publish Key）

输出脚本直接给出了收款人的公钥，输入脚本提供了用私钥对整个交易的签名，最后通过 `OP_CHECKSIG` 验证。我们知道签名算法是私钥签名公钥验证，如果通过验证，则证明这个行为确实是私钥拥有者所为，在这个例子中，花费交易用私钥对这笔交易进行签名，而上一笔输出交易用公钥对这笔花费交易的私钥进行验证，验证通过后，就可以证明这个交易确实是私钥拥有者做出的，并非冒牌。

在比特币脚本中是这样表示的；

```
input script:
    OP_PUSHDATA(Sig)
output script:
    OP_PUSHDATA(PubKey)
    OP_CHECKSIG
```

首先 `OP_PUSHDATA(Sig)` 和 `OP_PUSHDATA(PubKey)` 将Sig和PubKey压入栈中 Sig |

接着 `OP_CHECKSIG` 弹出栈顶两个元素验证签名 True |

栈中结果是True，证明私钥拥有者同时也拥有花费这笔交易Out的权利。

公钥哈希支付（Pay to Public Key Hash）

在 Pay to Publish Key 中，输出脚本中直接暴露了下一笔交易花费者的公钥，显然是不太合理的，于是又有了第二种输出脚本的类型 Pay to Public Key Hash 。

Pay to Public Key Hash 类型的输出脚本直接给出了收款人公钥的哈希，输入脚本提供了用私钥对整个交易的签名，同时也提供了自己的公钥用作验证，整个过程大同小异。

```
input script:
    OP_PUSHDATA(Sig)  //压入签名
    OP_PUSHDATA(PublicKey)  //压入公钥
output script:
    OP_DUP  //复制栈顶元素，再压入栈
    OP_HASH160  //弹出栈顶元素，取哈希在压入栈
    OP_PUSHDATA(PubKeyHash)  //压入输出脚本提供的公钥哈希
    OP_EQUALVERIFY   //弹出栈顶元素，比较是否相等
    OP_CHECKSIG   //公钥检查签名是否正确
```

脚本哈希支付（Pay to Script Hash）

这种形式的输出脚本是收款人提供脚本（redeemScript）的哈希，到时候收款人要花费这笔交易的时候需要输入脚本的内容和签名，验证的时候分两步；

- 验证输入脚本的哈希是否与输出脚本中的哈希值匹配
- 反序列化并执行redeemScript，验证input script给出的签名是否正确

> 采用BIP16的方案

```
input script:
    ...
    OP_PUSHDATA(Sig)
    ...
    OP_PUSHDATA(serialized redeemScript)
output script:
    OP_HASH160
    OP_PUSHDATA(redeemScriptHash)
    OP_EQUAL
```

其实可以用 Pay to Script Hash 实现 Pay to Public Key

```
redeemScript:
    PUSHDATA(PubKey)
    CHECKSIG
input script
    PUSHDATA(Sig)
    PUSHDATA(serialized redeemScript)
output script:
    HASH160
    PUSDHDATA(redeemScriptHash)
    EQUAL
```

Pay to Script Hash 在比特币最初版本是没有的，后期软分叉加入，最重要的一点是对多重签名的支持。

> 多重签名中，只要提超过供指定数量即可，容忍了一定程度的私钥丢失

原来的多重签名需要外部用户提供——对应的公钥，一共有多少公钥，几个公钥通过验证就可以完成交易，对用户比较繁琐，现在使用 Pay to Script Hash 将整个过程打包到脚本中，对外部用户来说降低了多重签名的复杂性，将复杂性转移到了系统中。

Proof of Burn

```
output script
    RETURN
    ...
```

包含了这样的output script的output被称为Provably Unspendable/Prunable Outputs。 假如有一个交易的input指向这个output，不论input里的input script如何设计，执行到RETURN这个命令之后都会直接返回false，RETURN后面的其他指令也就不会执行了，所以这个output无法再被花出去，对应的UTXO也就可以被剪枝了，无需全节点继续保存。

应用场景；

- 永久存储一些信息，比如在某个时间证明存在某些事情，比如在2019年1月1日把知识产权的哈希放到链上，当以后产生纠纷的时候，你把知识产权公布出来，知识产权的哈希在特定时间已经上链，就可以证明你在特定时间已经知道了这个知识产权。
- 代币转换，比如你把一些比特币转换成其他数字货币，你需要通过这种方式来证明你付出了一些代价。
- 销毁比特币。

其中在比特币脚本中已经出现了智能合约的雏形。

欢迎关注我的博客（qyuan.top），不定期分享一些区块链底层技术文章，博客排版要比知乎好一点（ヘ(̄､ ̄)厂）。

编辑于 2019-11-26

货币　区块链(Blockchain)　比特币 (Bitcoin)

文章被以下专栏收录

清源的区块链实验室　关注专栏

推荐阅读

比特币价格在 2011 年 6 月瞬间崩盘至 0.01 美元

BitMEX 研究

bitcoin 源码解析 - 交易 Transaction(四) - Script2

bitcoin 源码解析 - 交易 Transaction(四) - Script2现在发现写文章真是好没有什么动力… 所以就写的简洁些吧·· 随心说一些最关键的点，细节就不强调了。接上一文的《bitcoin 源码解…

金晓　发表于链块与分散…

深入理解

廖雪峰

还没有评论

写下你的评论...

▲ 赞同 2　▼　添加评论　分享　收藏　…