比特币脚本及交易分析 - 智能合约雏形

大家都有转过账，每笔交易是这样的：张三账上减￥200，李四账上加￥200。
在比特币区块链中，交易不是这么简单，交易实际是通过脚本来完成，以承载更多的功能个，这也是为什么比特币被称为是一种“可编程的货币”。
本文就来分析一下交易是如何实现可编程的。

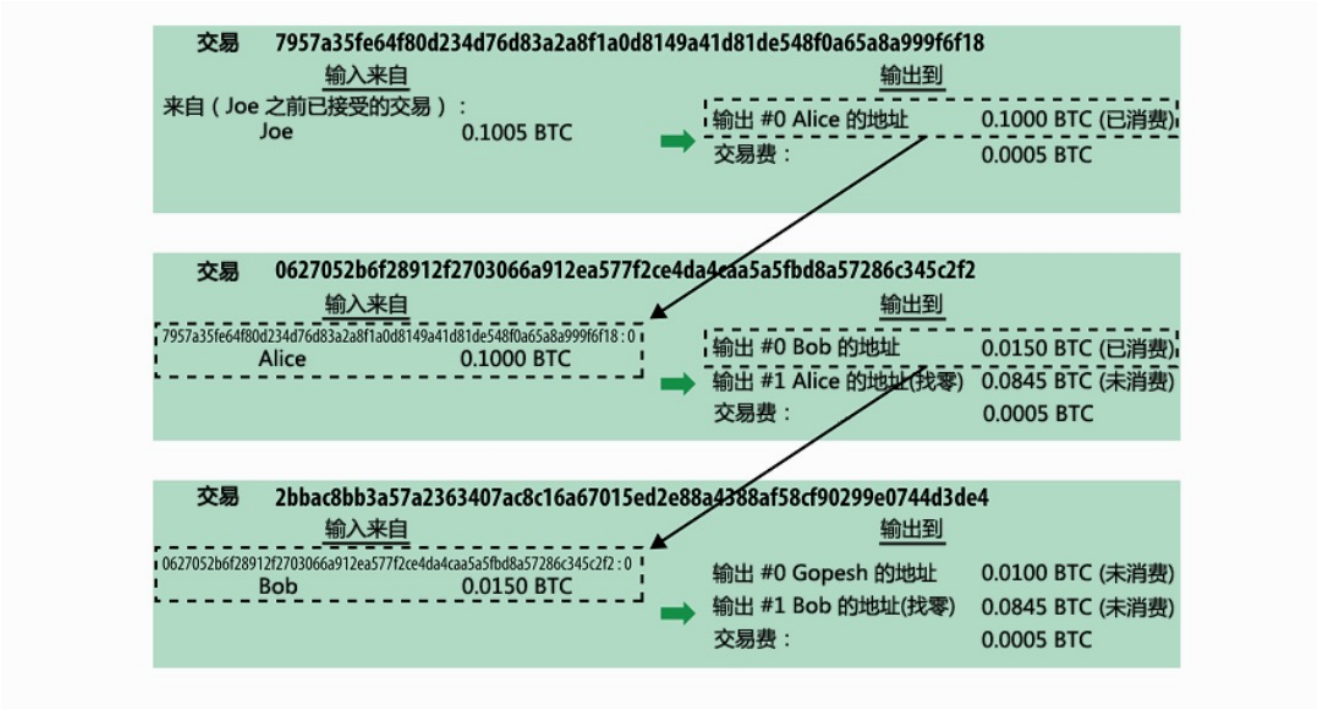
未花费的交易输出(UTXO)

先引入一个概念：未花费的交易输出——UTXO（Unspent Transaction Output）

其实比特币的交易都是基于UTXO上的，即交易的输入是之前交易未花费的输出，这笔交易的输出可以被当做下一笔新交易的输入。

> 挖矿奖励属于一个特殊的交易（称为coinbase交易），可以没有输入。
> UTXO是交易的基本单元，不能再分割。
> 在比特币没有余额概念，只有分散到区块链里的UTXO

随着钱从一个地址被移动到另一个地址的同时形成了一条所有权链，像这样：



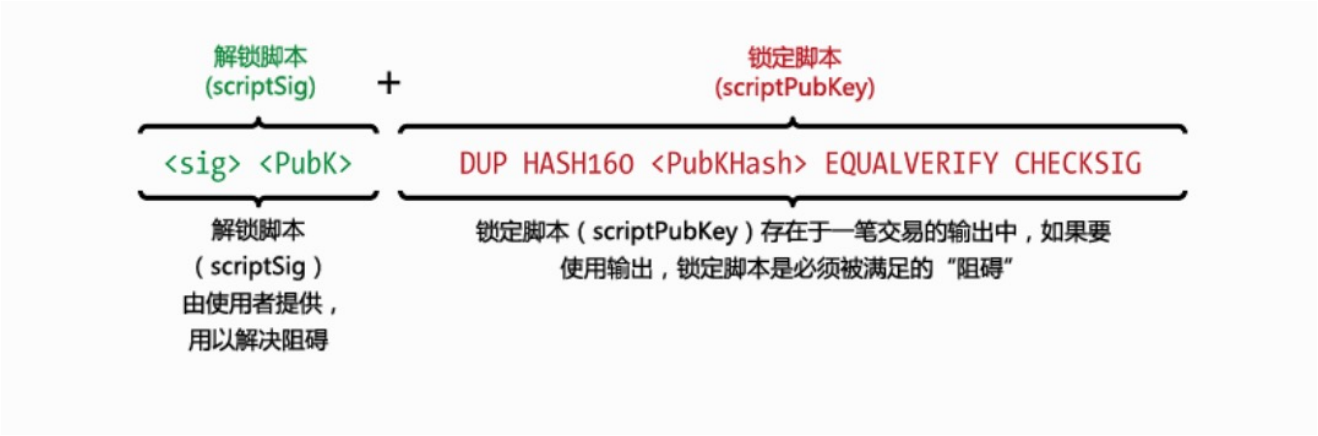
比特币脚本

比特币交易是首先要提供一个用于解锁UTXO（用私钥去匹配锁定脚本）的脚本（常称为解锁脚本：Signature script)，这也叫交易输入，交易的输出则是指向一个脚本（称为锁定脚本：PubKey script），这个脚本表达了：谁的签名（签名是常见形式，并不一定必须是签名）能匹配这个输出地址，钱就支付给谁。

每一个比特币节点会通过同时执行这解锁和锁定脚本（不是当前的锁定脚本，是指上一个交易的锁定脚本）来验证一笔交易，脚本组合结果为真，则为有效交易。

> 当解锁版脚本与锁定版脚本的设定条件相匹配时，执行组合有效脚本时才会显示结果为真

如最为常见类型的比特币交易脚本（支付到公钥哈希：P2PKH（Pay-to-Public-Key-Hash））组合是这样：



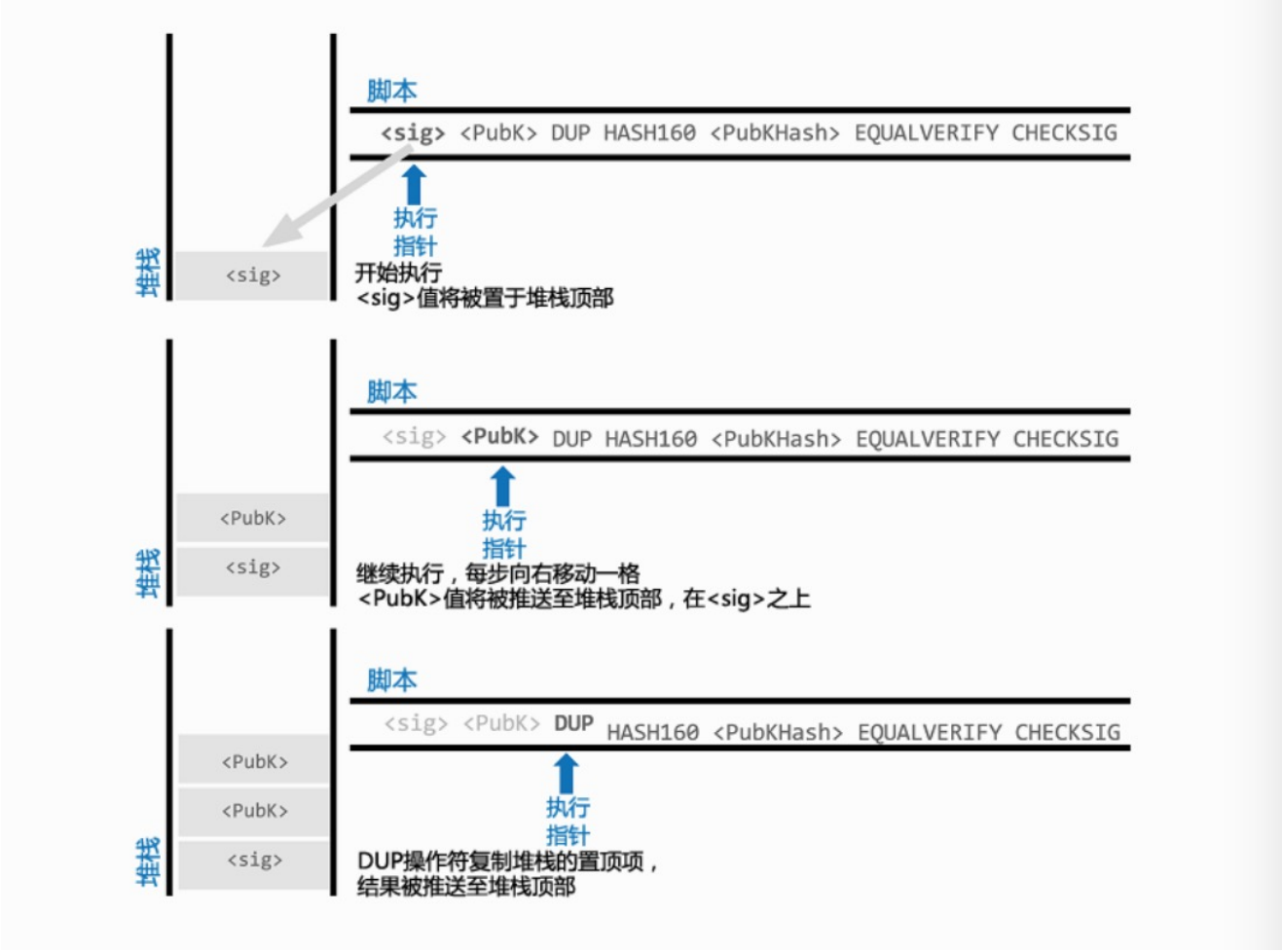
常见交易脚本验证过程

比特币交易脚本语言是一种基于逆波兰表示法的基于栈的执行语言（不知道逆波兰和栈的同学去翻大学数据结构课本，你也可跳过这个部分）。

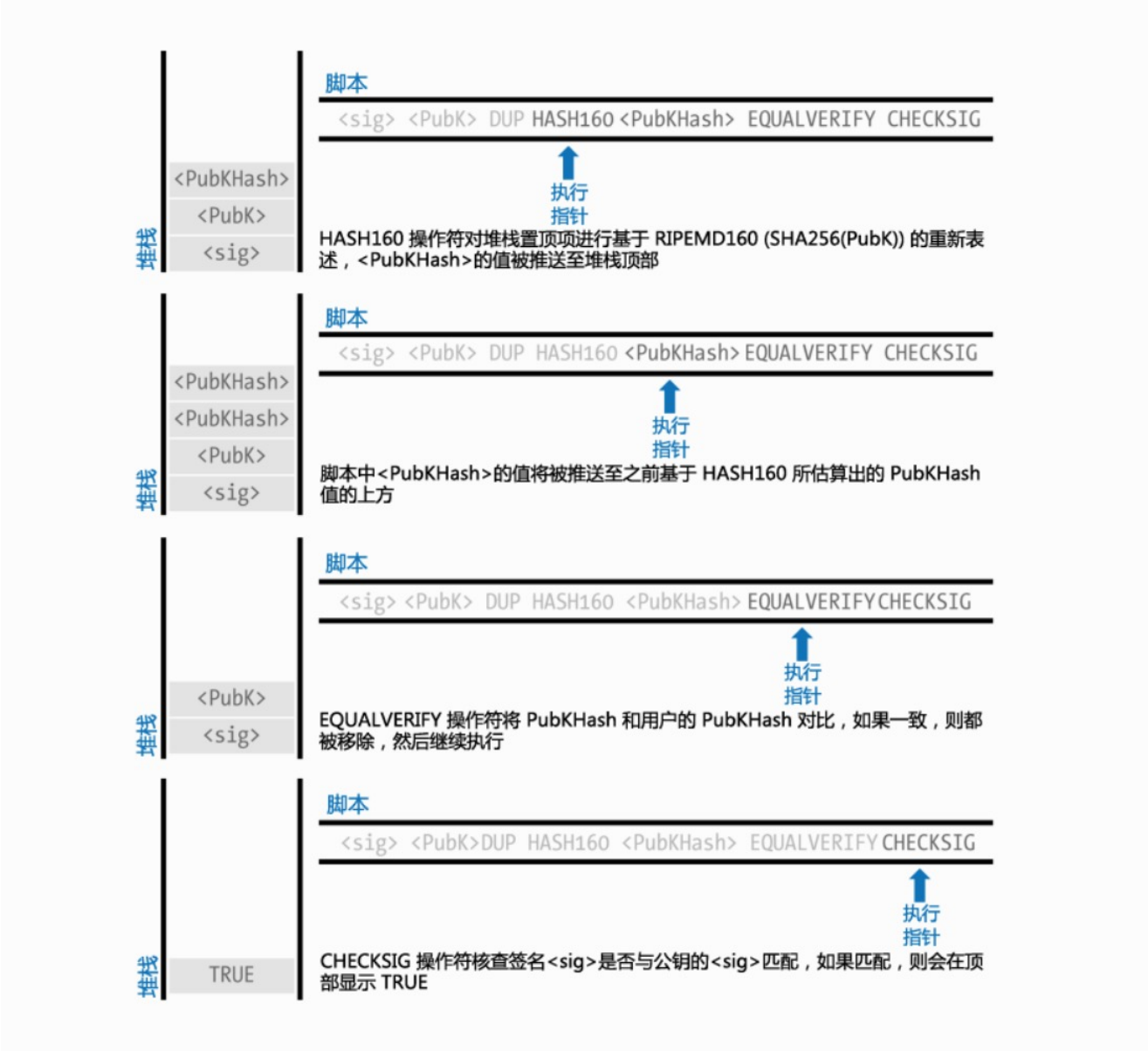
> 比特币脚本语言包含基本算数计算、基本逻辑(比如if...then)、报错以及返回结果和一些加密指令，不支持循环。想了解更多语言细节可参考:比特币脚本

脚本语言通过从左至右地处理每个项目的方式执行脚本。

下面用两个图说明下常见类型的比特币交易脚本验证执行过程：



上图为解锁脚本运行过程（主要是入栈）



上图为锁定脚本运行过程（主要是出栈），最后的结果为真，说明交易有效。

交易分析

实际上比特币的交易被设计为可以纳入多个输入和输出。

交易结构

我们来看看完整的交易结构，

字段	描述	大小
版本	这笔交易参照的规则	4 字节
输入数量	交易输入列表的数量	1 - 9 字节
输入列表	一个或多个交易输入	不定
输出数量	交易输出列表的数量	1 - 9 字节
输出列表	一个或多个交易输出	不定
锁定时间	锁定时间	4 字节

> 交易的锁定时间定义了能被加到区块链里的最早的交易时间。在大多数交易里，它被设置成0，用来表示立即执行。
> 如果锁定时间不是0并且小于5亿，就被视为区块高度，意指在这个指定的区块高度之前，该交易不会被包含在区块链里。
> 如果锁定时间大于5亿，则它被当作是一个Unix纪元时间戳（从1970年1月1日以来的秒数），并且在这个指定时间之前，该交易不会被包含在区块链里。

交易的数据结构没有交易费的字段，交易费通过所有输入的总和，以及所有输出的总和之间的差来表示，即：

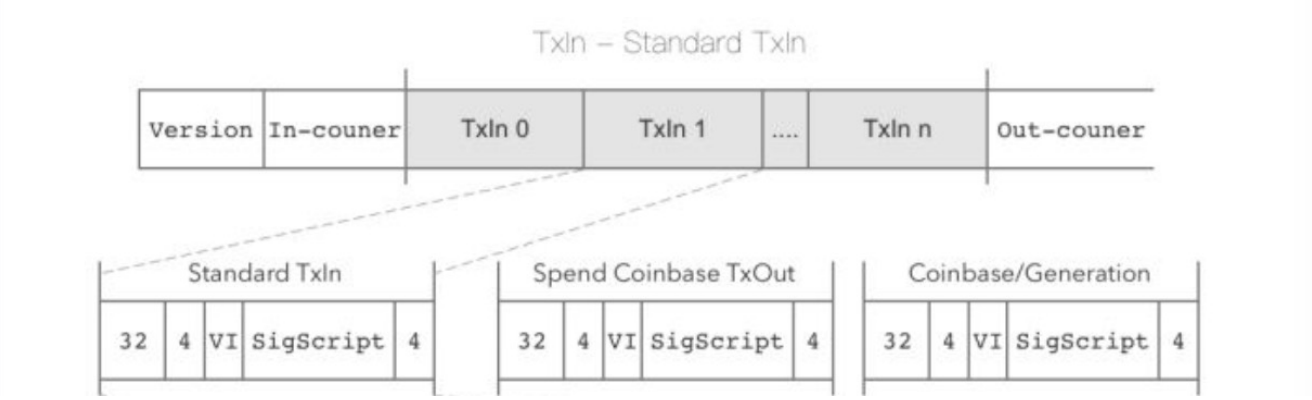
> 交易费 = 求和（所有输入） - 求和（所有输出）

交易输入结构

刚刚我们提过输入需要提供一个解锁脚本，现在来看看一个交易的输入结构：

字节	字段	描述
32	交易哈希值	指向被花费的UTXO所在的交易的哈希指针
4	输出索引	被花费的UTXO的索引号，第一个是0
1-9	解锁脚本大小	用字节表示的后面的解锁脚本长度
不定	解锁脚本	满足UTXO解锁脚本条件的脚本
4	序列号	目前未被使用的交易替换功能，设为0xFFFFFFFF

我们结合整个交易的结构里看输入结构就是这样子：

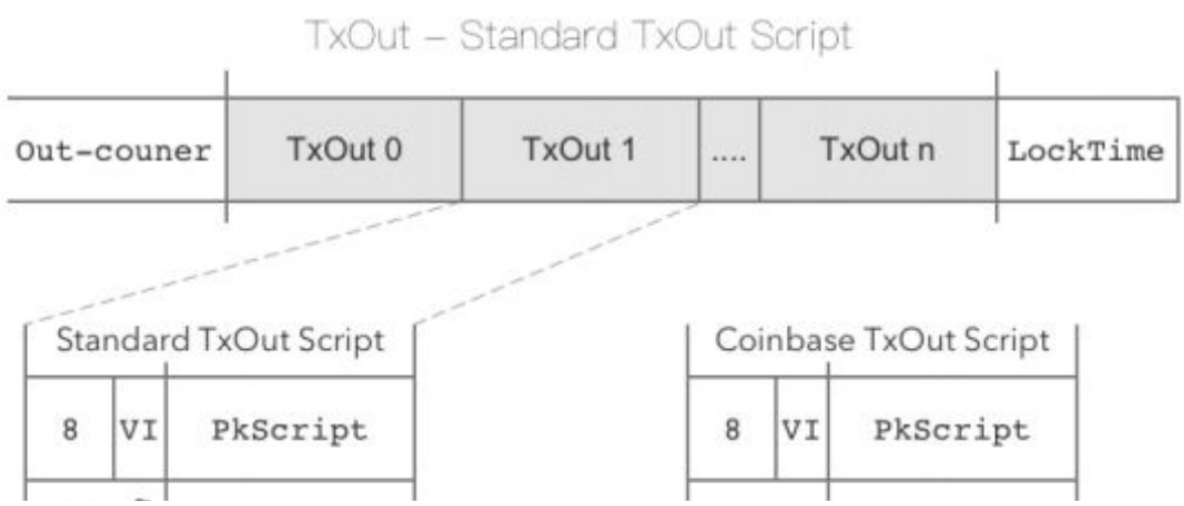


交易输出结构

刚刚我们提过输出是指向一个解锁脚本，具体交易的输出结构为：

字节	字段	描述
8	总量	用聪表示的比特币值
1-9	锁定脚本大小	用字节表示的后面的锁定脚本长度
不定	锁定脚本	一个定义了支付输出所需条件的脚本

我们结合整个交易的结构里看输出结构就是这样子：



交易哈希计算

在比特币区块结构Merkle 树及简单支付验证分析 讲到区块结构，区块结构包含多个交易的哈希。
那么交易哈希是怎么计算的呢？

1. 交易结构各字段序列化为字节数组
2. 把字节数组拼接为支付串
3. 对支付串计算两次SHA256 得到交易hash

了解详情可进一步参考如何计算交易Hash?及如何创建Hash?

现在是不是对完整的交易到区块有了更清晰的认识。

智能合约雏形 - 应用场景说明

由于交易是通过脚本来实现，脚本语言可以表达出无数的条件变种。

比特币的脚本目前常用的主要分为两种，一种是常见的P2PKH（支付给公钥哈希），另一种是P2SH（Pay-to-Script-Hash支付脚本哈希）。
P2SH支付中，锁定脚本被密码学哈希所取代，当一笔交易试图支付UTXO时，要解锁支付脚本，它必须含有与哈希相匹配的脚本。

这里不展开技术细节，下面说明一些应用场景，以便大家有更直观的认识。

- 多重签名应用
  合伙经营中，如只有一半以上的的股东同意签名就可以进行支付，可为公司治理提供管控便利，同时也能有效防范盗窃、挪用和遗失。

  用于担保和争端调解，一个买家想和他不认识或不信任的某人交易，在一般情况交易正常进行时，买家不想任何第三方参与。那交易双方可以发起支付，但如果交易出现问题时，那第三方就可以根据裁定，使用自己的签名和裁定认可的一方共同签名来兑现这笔交易。