比特币交易(Bitcoin Transactions)

本文译自比特币 WIKI：https://en.bitcoin.it/wiki/Transaction 和

https://en.bitcoin.it/wiki/Transaction_Malleability

译者：申屠青春　深圳大学 ATR 国防科技重点实验室博士　新浪微博　@我看比特币

注意：本文可随意转发，请留下译者信息，如果觉得本文对你有用，请给译者捐赠，以便翻译更多比特币的核心资料。捐赠地址：1faVxBp2KmST98p3tJjx2MQP98JLLnF2Q

译者前言

比特币在国内已经众所周知，但是技术研究并未有效开展，大部分人处于知道和了解程度，目前比特圈中许多人对比特币能做什么，同样了解不多。一个重要原因是大多数比特币核心资料都是英文，很少有人能静心看完如此繁杂的英文资料。本人博士论文的研究方向是比特币，在研究其英文技术的同时，拟对一些重要资料进行翻译，让更多的圈内人对比特币有更多的理解。

本文主题是比特币交易，交易是整个比特币体系的核心，没有交易就没有比特币，同时也说明了交易可塑性的原理。

正文

交易是签过名的数据块，该数据块在网络中广播，并且被收集到块中。它引用以前的交易，从该交易中发送特定数据的比特币到一个或多个公钥中(即比特币地址)，交易未被加密(比特币体系中没有加密任何数据)。

块链浏览器是指一个网站，在该网站上可以浏览到被包含在块链中的每一个交易，有助于理解交易操作的技术细节，对支付验证也很有用。

1 比特币交易的一般格式(在一个块中)

数据项	描述	大小
版本号	目前为 1	4 字节
输入数量	正整数 VI = VarInt	1 - 9 字节
输入列表	每块的第一个交易的第一个输入叫做 "coinbase" (早期版本中内容被忽略)	<in-counter>-许多输入

输出数量	正整数 VI = VarInt	1 - 9 字节
输出列表	块中的第一个交易的输出是花掉挖矿得到的比特币	<out-counter>-许多输出
锁定时间 lock_time	如果非 0 并且序列号小于 0xFFFFFFFF ，是指块序号；如果交易已经终结，则是指时间戳	4 字节

2 带有 1 个输入和 1 个输出的比特币交易的例子

2.1 数据

```
Input:
Previous tx: f5d8ee39a430901c91a5917b9f2dc19d6d1a0e9cea205b009ca73dd04470b9a6
Index: 0
scriptSig: 304502206e21798a42fae0e854281abd38bacd1aeed3ee3738d9e1446618c4571d10
90db022100e2ac980643b0b82c0e88ffdfec6b64e3e6ba35e7ba5fdd7d5d6cc8d25c6b241501

Output:
Value: 5000000000
scriptPubKey: OP_DUP OP_HASH160 404371705fa9bd789a2fcd52d2c580b65d35549d
OP_EQUALVERIFY OP_CHECKSIG
```

2.2 解释

该交易的输入从交易 f5d8ee39a430901c91a5917b9f2dc19d6d1a0e9cea205b009ca73dd04470b9a6 的 0 号输出中导入了 50 个比特币，其输出发送了 50 个比特币到一个比特币地址(这里用十六进制表示：404371705fa9bd789a2fcd52d2c580b65d35549d，而非正常的 base58 表示)。如果接收者想花掉这些钱，他首先创建自己的交易 B，再引用该交易 A 的 0 号输出作为 B 交易的输入。

2.2.1 输入

一个输入是对其他交易的输出的引用，多个输入通常列在一个交易中。所有被引用的输出值相加，该总和值在该交易 A 的输出中用到。Previous tx 是以前交易的 HASH 值，Index 是被引用的交易的特定输出号，ScriptSig 是一个脚本的前一半(脚本将在后续详细讨论)

脚本包含两个部分，一个签名和一个公钥，公钥属于交易输出的赎回者，并且证明交易创建者被允许赎回输出值，另一个部分是 ECDSA 签名，是通过对简化交易的 HASH 值进行 ECDSA 签名而得到。签名和公钥一起，证明原地址的真正所有者创建了该支付交易，许多指标定义了是如何简化，并且可

以用来创建不同类型的支付。

2.2.2 输出

一个输出包含发送比特币的指令，Value 是以聪(Satoshi，1BTC=100,000,000 聪)为单位的数值，当该输出被赎回时，这个数值是非常有价值的。ScriptPubKey 是脚本的另一半(在后面讨论)，还可以有多于一个输出，他们共享了输入的总和值。因为一个交易的每个输出只能被后来的交易当成输入引用一次。如果你不想丢币，需要把所有输入值的总和值发送到一个输出地址，如果输入是 50BTC，但你仅想发送 25BTC，比特币将创建 2 个 25BTC 的输出：一个发往目标地址，另一个回到你的地址(称之为“找零”，即使你是发送给自己了)。任何输入的作为交易费的比特币不能被赎回，将被生成这个块的矿工得到。

2.2.3 验证

为了验证某个交易的输入已经被授权，可以收集被引用的输出中的所有币值，比特币体系使用了一个类似于 Forth 的脚本系统，输入的 scriptSig 和被引用的输出 scriptPubKey 会被评估(按顺序)，评估 scriptPubKey 时会使用 scriptSig 留在堆栈里的值。 如果 scriptPubKey 返回真，则输入被授权。通过脚本系统，发送者可以创建非常复杂的条件，人们为了赎回输出值则必须满足这些条件。举个例子，可以创建一个能被任何人赎回而无需授权的输出，也可以创建一个需要 10 个不同签名的输入，或者无需公钥仅由密码赎回的输出。

3 交易类型

比特币目前创建两个不同的 scriptSig/scriptPubKey 对，描述如下。

创建更复杂的交易类型并且把他们关联成密码学加强的合同，这是完全可能的，我们称之为合同。

3.1 支付到公钥 HASH 地址

scriptPubKey: OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG

scriptSig: <sig> <pubKey>

一个比特币地址只是一个 HASH 值，因而发送者无法在 scriptPubKey 中提供完整的公钥，当要赎回已经被发送到一个比特币地址的比特币时，接收者需同时提供签名和公钥，脚本会验证公钥的

HASH 确实与 scriptPubKey 中的 HASH 值匹配，还会检查公钥和签名是否匹配。检查过程如下：

堆栈	脚本	描述
空	<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	scriptSig 和 scriptPubKey 联合
<sig> <pubKey>	OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	遇到常数，压入堆栈
<sig> <pubKey> <pubKey>	OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	复制栈顶元素
<sig> <pubKey> <pubHashA>	<pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	HASH 栈顶元素
<sig> <pubKey> <pubHashA> <pubKeyHash>	OP_EQUALVERIFY OP_CHECKSIG	遇到常数，压入堆栈
<sig> <pubKey>	OP_CHECKSIG	检查两个栈顶元素是否相等
true	Empty.	用两个栈顶元素，检查签名是否正确。

3.2 支付到脚本 HASH

（译者按：官方威客上本段没有内容，以下内容来自币付宝潘志彪 kevin 的博客：618.io）

该类交易目前不是很常见，但是非常有意义。未来应该会在某些场合频繁使用。该类交易的接受地址不是通常意义的地址，而是一个合成地址，以 3 开头（对，以 3 开头的也是比特币地址!）。三对公私钥，可以生成一个合成地址。在生成过程时指定 n of 3 中的 n，n 范围是[1, 3]，若 n=1，则仅需一个私钥签名即可花费该地址的币，若 n=3，则需要三把私钥依次签名才可以。

合成地址以 3 开头，可以实现多方管理资产，极大提高安全性，也可以轻松实现基于比特币原生的三方交易担保支付。一个 M-of-N 的模式：

m {pubkey}...{pubkey} n OP_CHECKMULTISIG

M 和 N 需满足：

1<=N<=3

1<=M<=N

可以是 1 of 1，1 of 2，2 of 3 等组合，通常选择 N=3：

- 1 of 3，最大程度私钥冗余。防丢私钥损失，3 把私钥中任意一把即可签名发币，即使丢失 2 把都可以保障不受损失；
- 2 of 3，提高私钥冗余度的同时解决单点信任问题。3 把私钥任意 2 把私钥可签名发币，三方不完全信任的情形，即中介交易中，非常适用；

- 3 of 3，最大程度解决资金信任问题，无私钥冗余。必须 3 把私钥全部签名才能发币，适用多方共同管理重要资产，但任何一方遗失私钥均造成严重损失；

合成地址的交易构造、签名、发送过程与普通交易类似，这里只介绍如何创建一个合成地址。大神 Gavin Andresen 已经演示过，请看：https://gist.github.com/gavinandresen/3966071 .

3.3 比特币生成交易

生成交易只有一个输入，该输入有一个”coinbase”参数没有 scriptSig，在”coinbase”中的数据可以是任意内容，它不会被使用。比特币把压缩的当前 HASH 目标值和任意精确度的” extraNonce”存贮在这儿， 区块头中的 Nonce 每次溢出，它们都会增长。输出可以是任何内容，但比特币创建了一个准确的类似于 IP 地址的交易。extraNonce 有助于扩大工作量证明函数的范围，矿工很容易修改 Nonce(4 个字节)、时间戳和 extraNonce(2-100 字节)。

译者按：(1)当前 HASH 目标值解释如下，挖矿软件随机生成一个随机数，放到正在生成的块里，对块进行 HASH，使得该 HASH 值小于或等于当前 HASH 目标值，就代表挖矿成功；如果不成功则重新生成外加随机数，再次 HASH) (2) Nonce 溢出：是指在对一个块进行 HASH 时，Nonce 从 0 开始，每计算一次 HASH 都要增长一次，因而有可能会超过数值范围的情况，extraNonce 就要相应增长以存贮 Nonce。

4 交易的每一个输入的一般格式(在块内)-Txin

数据项	描述	大小
以前交易的 HASH	以前交易的两次 SHA256HASH 值	32 字节
以前 交易的 Txout-index	非负整数，用来索引被引用的交易的输出	4 字节
Txin-script 长度	非负整数 VI = VarInt	1 - 9 字节
Txin-script / scriptSig	Script 脚本	<in-script length>-许多字节
序列号 sequence_no	正常是 0xFFFFFFFF; 当交易的 lock_time 大于 0 时有意义	4 字节

输入充分描述了去哪里拿到要赎回的币、以及怎么拿到。如果这个输入是块中的第一个交易的第一个输入，我们称之为生产交易，它的输入和内容完全被忽略。(一般情况，以前的交易 HASH 是 0，以前的 Txout-index 为-1)

5 交易的每一个输出的一般格式(在块内)-Txout

数据项	描述	大小
输出值	非负整数，以 Satoshi 聪为单位的数值，表示要被发送的币数量	8 字节
Txout- script 长度	非负整数	1 - 9 字节 VI = VarInt
Txin-script/ scriptPubKey	Script 脚本	<out-script length>-许多字节

在输出中设置以后释放这些比特币数量的条件，第一个交易的输出值的总和，等于被矿工挖到的该块比特币数量加上块中其他交易的交易费。

6 交易可塑性

当交易被签名时，该签名并没有覆盖交易中的所有数据。因而，在非正常情况下，一个网络节点可以使得 HASH 无效来改变你发送的交易。注意：这样只是改变了 HASH 值，交易的输出没有改变，比特币会被发送到先前指定的地址。然而，这并不意味着，例如，在任何条件下接受未确认的交易是不安全的，因为后续的交易要依赖以前交易的 HASH 值，这些 HASH 值可以被改变，直到它们在一个块中被确认后，才不会再改变。(如果块链重组，有可能要等到一个确认以后才不会再改变)。另外，钱包必须经常扫描与它相关的交易，如果是因为钱包创建了 txout 而假定该 txout 一定存在，这是不安全的。

6.1 签名可塑性

可塑性的第一个形式是签名本身，每个签名仅有一个 DER 编码的 ANS.1 的 8 进制表示，但是 openssl 并不强制要求，如果签名本身不是特别奇特，一般都会被接受。另外，对于每个 ECDSA 签名(r,s)，签名(r, -s (mod N))是对相同信息的有效签名。

正在努力使得比特币节点不转播非标准签名，最终达到完全不许它们被包括进新块。

6.2 scriptSig 可塑性

比特币中的签名算法，未把任何 scriptSig 包括在内，因为要对签名本身签名，这是不可能的。这就意味着可以把其他数据加入到交易中，附加的数据优先于签名和公钥，被压入堆栈。类似地，可以把 OP_DROP 加入到脚本中，以便堆栈恢复 scriptPubKey 操作之前的状态。

正在考虑阻止 scriptSig 可塑性，当前的交易，如果在 scriptSig 中有数据入栈之外的任何操作，都被认为是非标准交易，并且不会被转发，最终该无规则会强化到：在脚本执行完成后，堆栈中只能有一个项。然后，这样做可能影响到比特币后续扩展性。