交易结构

咖啡交易 0627052b6f28912f2703066a912ea577f2ce4da4caa5a5fbd8a57286c345c2f2

版本(0x01)　上个交易 Index

输入个数　　UTXO输入(来源交易哈希)

```
01000000011186f9f998a5aa6f048e51dd8419a14d8a0f1a8a2836dd734d2804fe65fa35779000000008b483045022100884d142d86652a3f47ba4746ec719bbfbd040a570b1deccbb6498c75c4ae24cb02204b9f039ff08df09cbe9f6addac960298cad530a863ea8f53982c09db8f6e3813014104484ecc0d46f1918b30928fa0e4ed99f16a0fb4fde0735e7ade8416ab9fe423cc5412336376789d172787ec3457eee41c04f4938de5cc17b4a10fa336a8d752adfffffffff0260e31600000000001976a914ab68025513c3dbd2f7b92a94e0581f5d50f654e788acd0ef8000000000001976a9147f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a888ac00000000
```

解锁脚本长度(0x8b, 139)　解锁脚本　输出1金额(0x0016e360, 1500000)

输出个数　　输出1锁定脚本

锁定时间(LockTime)

解锁脚本结构

签名长度(0x48, 72)

```
48304502210088 4d142d86652a3f47ba4746ec719bbfbd040a
570b1deccbb6498c75c4ae24cb02204b9f039ff08df09cbe9f
6addac960298cad530a863ea8f53982c09db8f6e3813014104
84ecc0d46f1918b30928fa0e4ed99f16a0fb4fde0735e7ade8
416ab9fe423cc5412336376789d172787ec3457eee41c04f49
38de5cc17b4a10fa336a8d752adf
```

公钥长度(0x41, 65)

```
<Signature 3045..1301> <PublicKey 0484ecc..52adf>
```

签名序列化（DER）


签名长度(0x45, 69)
整数R长度(0x21, 33)
签名前缀(0x30)
整数前缀(0x02)
R 序列
30450221 00884d142d86652a3f47ba4746ec719
bbfbd040a570b1deccbb6498c75c4ae24cb0220
4b9f039ff08df09cbe9f6addac960298cad530a
863ea8f53982c09db8f6e38130 1
整数前缀(0x02)
S 序列
签名类型(0x01, SIGHASH_ALL)
整数S长度(0x20, 32)

锁定脚本结构

咖啡交易UTXO来源交易 7957a35fe64f80d234d76d83a2a8f1a0d8149a41d81de548f0a65a8a999f6f18

```
0100000001524d288f25cada331c298e21995ad070e1d1a0793e818f2f7cfb5f6122
ef3e71000000008c493046022100a59e516883459706ac2e6ed6a97ef9788942d3c9
6a0108f2699fa48d9a5725d1022100f9bb4434943e87901c0c96b5f3af4e7ba7b83e
12c69b1edbfe6965f933fcd17d014104e5a0b4de6c09bd9d3f730ce56ff42657da3a
7ec4798c0ace2459fb007236bc3249f70170509ed663da0300023a5de700998bfec4
9d4da4c66288a58374626c8dffffffff0180969800000000001976a9147f9b1a7fb6
8d60c536c2fd8aeaa53a8f3cc025a888ac00000000
```

锁定脚本长度(0x19, 31)

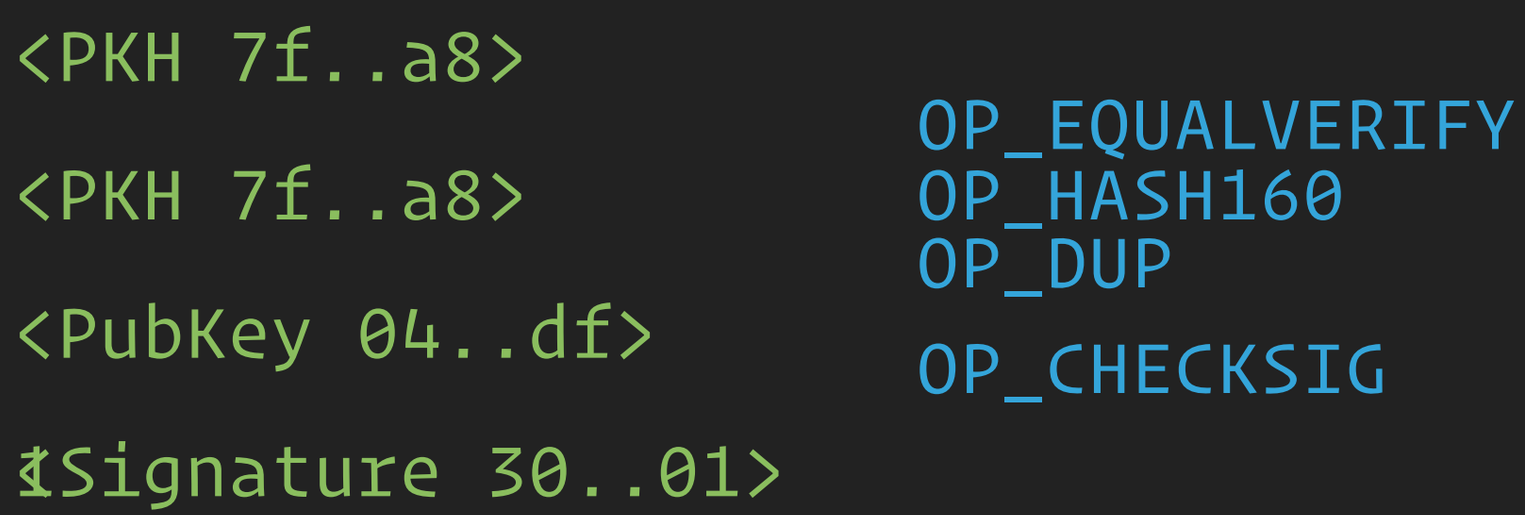
后面(0x14, 20)长度入栈

```
76a9147f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a888ac
```

```
OP_DUP OP_HASH160 7f9b1a7fb..cc025a8 OP_EQUALVERIFY OP_CHECKSIG
```

P2PKH 脚本执行栈

<Signature 30..01> <PubKey 04..df> OP_DUP OP_HASH160 <PKH 7f..a8> OP_EQUALVERIFY OP_CHECKSIG



验签过程

<Signature 3045..1301> <PublicKey 0484ecc..52adf>

```
0100000001524d288f25cada331c298e21995ad070e1d1a0793e818f2f7cfb5f6122ef3e71000000
008c493046022100a59e516883459706ac2e6ed6a97ef9788942d3c96a0108f2699fa48d9a5725d1
022100f9bb4434943e87901c0c96b5f3af4e7ba7b83e12c69b1edbfe6965f933fcd17d014104e5a0
b4de6c09bd9d3f730ce56ff42657da3a7ec4798c0ace2459fb007236bc3249f70170509ed663da03
00023a5de700998bfec49d4da4c66288a58374626c8dffffffff0180969800000000001976a9147f
9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a888ac00000000
```

UTXO 来源的交易

```
0100000001186f9f998a5aa6f048e51dd8419a14d8a0f1a8a2836dd734d2804fe65fa35779000000008b483
04502210084d142d86652a3f47ba4746ec719bbfbd040a570b1deccbb6498c75c4ae24cb02204b9f039ff0
8df09cbe9f6addac960298cad530a863ea8f53982c09db8f6e381301410484ecc0d46f1918b30928fa0e4ed
99f16a0fb4fde0735e7ade8416ab9fe423cc5412336376789d172787ec3457eee41c04f4938de5cc17b4a10
fa336a8d752adfffffffff0260e31600000000001976a914ab68025513c3dbd2f7b92a94e0581f5d50f654e
788acd0ef8000000000001976a9147f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a888ac00000000
```

```
0100000001186f9f998a5aa6f048e51dd8419a14d8a0f1a8a2836dd734d2804fe65fa35779
00000000001976a9147f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a888acffffffff0260e3
1600000000001976a914ab68025513c3dbd2f7b92a94e0581f5d50f654e788acd0ef800000
0000001976a9147f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a888ac0000000001000000
```

验签过程

```
0100000001186f9f998a5aa6f048e51dd8419a14d8a0f1a8a2836dd734d28
04fe65fa35779000000001976a9147f9b1a7fb68d60c536c2fd8aeaa53a8f
3cc025a888acffffffff0260e3160000000001976a914ab68025513c3dbd
2f7b92a94e0581f5d50f654e788acd0ef8000000000001976a9147f9b1a7f
b68d60c536c2fd8aeaa53a8f3cc025a888ac0000000001000000
```

$SHA256^2$

```
83cb5dc661ba879af76a741308ef7b1d87d55e046f0c8640f8ff4c17ac080730
```

```
ecdsa_verify_signature(message, <Signature 3045..1301>, <PublicKey 0484ecc..52adf>)
```

签名类型

0x01 SIGHASH_ALL

0x02 SIGHASH_NONE

0x03 SIGHASH_SINGLE

0x81 SIGHASH_ANYONECANPAY | SIGHASH_ALL

0x82 SIGHASH_ANYONECANPAY | SIGHASH_NONE

0x83 SIGHASH_ANYONECANPAY | SIGHASH_SINGLE

ECDSA 原理

随机私钥 r 和对应公钥 R 有关系 $R = Gr$

签名私钥 k 和对应公钥 K 有关系 $K = Gk$

待签名消息哈希 h

计算签名 $S = \frac{h + k X_R}{r}$ 签名则由 S 和 R 组成

验证签名 $R = \frac{h}{S}G + \frac{X_R}{S}K$

$$R = \frac{h}{S}G + \frac{X_R}{S}Gk = \frac{G(h + k X_R)}{S} = rG$$

多重签名脚本

```
0 <Sign B> <Sign C>

2 <PublicKey A> <PublicKey B> <PublicKey C> <PublicKey D> <PublicKey E> 5 OP_CHECKMULTISIG
```

- 锁定脚本复杂，发送方交易费用高；
- 全节点存储 UTXO 的负担大；
- Public Key 存储在锁定脚本，影响安全性；

P2SH 脚本

```
0 <Sign B> <Sign C> <2 PubKeyA PubKeyB PubKeyC 3 CHECKMULTISIG> HASH160 <Hash 54..7e> EQUAL
```

1. [0] 0 入栈
2. [0, <Sign B>] 签名 B 入栈
3. [0, <Sign B>, <Sign C>] 签名 C 入栈
4. [0, <Sign B>, <Sign C>, <2 PubKeyA PubKeyB PubKeyC 3 CHECKMULTISIG>] 脚本完整入栈
5. [0, <Sign B>, <Sign C>, <Hash>] 计算栈顶 HASH160
6. [0, <Sign B>, <Sign C>, <Hash>, <Hash 54..7e>] 目标脚本哈希入栈
7. [0, <Sign B>, <Sign C>] EQUAL 比较栈顶两个哈希，相同准备运行原始脚本
8. [0, <Sign B>, <Sign C>, 2, PubKeyA, PubKeyB, PubKeyC, 3] 脚本数据依次入栈
9. [0, <Sign B>, <Sign C>, 2, PubKeyA, PubKeyB, PubKeyC] CHECKMULTISIG 取出公钥数 3
10. [0, <Sign B>, <Sign C>, 2] 再取出 3 个公钥
11. [0, <Sign B>, <Sign C>] 再取出签名数 2
12. [1] 接着取出 2 个签名和数字 0，根据交易验证签名，入栈结果

P2SH 脚本

```
0 <Sign B> <Sign C> <2 PubKeyA PubKeyB PubKeyC 3 CHECKMULTISIG> HASH160 <Hash 54..7e> EQUAL
```

- 复杂的多重签名部分从锁定脚本转移到解锁脚本，发送方交易费用负担转移到接受方；
- 全节点存储 UTXO 的负担变小，链存储交易负担从当前转移到未来；
- Public Key 不存储在锁定脚本，提高安全性；