The Bitcoin programming language

crypto251.0 Cryptocurrency and the Smileycoin

Gunnar Stefansson (editor)

November 20, 2019

From input to output

Tx cc3b743938e485578315b2f6848c1a416c917585ea2f75d5d3e09f21a95008b0

output 0


```
{
   "txid": "cc3b743938e485578315b2f6848c1a416c917585ea2f75d5d3e09f21a95008b0",
   "version": 1,
   "locktime": 0,
   "vin": [
      {
         "txid": "5376633c095470e5a1aafab742226355ac2b3e2b7c8c10bd4c0191bd2ceb8ea8",
         "vout": 0,
         "scriptSig": {
            "asm": "304402206156d5bee4c07be3a52b253ce063b0d2c8d46a9ed6c8cbeb5b1b3f775f9ee50e022004f6151556d7974e27a94b596207ff1dc8f0789e776d1ea3677b
            "hex": "47304402206156d5bee4c07be3a52b253ce063b0d2c8d46a9ed6c8cbeb5b1b3f775f9ee50e022004f6151556d7974e27a94b596207ff1dc8f0789e776d1ea367
         },
         "sequence": 4294967295
      }
   ],
   "vout": [
      {
         "value": 164613.17791999,
         "n": 0,
         "scriptPubKey": {
            "asm": "OP_DUP OP_HASH160 df75ee5b514b5253979ed29524fde386482f05cf OP_EQUALVERIFY OP_CHECKSIG",
            "hex": "76a914df75ee5b514b5253979ed29524fde386482f05cf88ac",
            "reqSigs": 1,
            "type": "pubkeyhash",
            "addresses": [
               "BQpdaVJhc5YLbfEYocV3iRzwYGqYqcAFyc"
            ]
         }
      },
      {
         "value": 35385.82208001,
         "n": 1,
         "scriptPubKey": {
            "asm": "OP_DUP OP_HASH160 92a5c66c3299fbb2102439c4645590f06505b719 OP_EQUALVERIFY OP_CHECKSIG",
            "hex": "76a91492a5c66c3299fbb2102439c4645590f06505b71988ac",
            "reqSigs": 1,
            "type": "pubkeyhash",
            "addresses": [
               "BHpUryfdq8rzktFz1scyDVVbZgG5Nj3weo"
            ]
         }
      }
   ],
   "blockhash": "d2d5ed7a860eb617793cd7e7a35d006b64444d5b3b2a007c1cf85b35ec5ef038",
   "confirmations": 585,
   "time": 1523514754,
   "blocktime": 1523514754
}
```


“asm:” “OP_DUP OP_HASH160 **df75ee5b514b5253979ed29524fde3864**
OP_EQUALVERIFY OP_CHECKSIG”,
“hex:” “76a914df75ee5b514b5253979ed29524fde386482f05cf88ac”,
i.e.

The assembler

where we see (from `https://en.bitcoin.it/wiki/Script`) the meaning of the sequence
`OP_DUP OP_HASH160 OP_EQUALVERY OP_CHECKSIG OP_EQUAL OP_VERIFY`
in the table in the handout.

Simple example

```
2 7 OP_ADD 3 OP_SUB 1 OP_ADD 7 OP_EQUAL
```

spending

Then the value gets used in
Tx e870614afe3cb9fde97566b024a72f11d22ce08dbd89a971655b15f71d6e203b
(see handout)

A more detailed look inside the spending transaction

Looking at SMLY transactions
dee018b9be101c519ad326c581807581a4e46711f242b4878afa1d98c5f52
and
08a5430a667a700e7b913d1895908c56d035559cd32a999a5c8cd56f409f4
The former's vout:0 gets spent in the latter.

A more detailed look at P2SH

A case study in P2SH is given below in the form of using a **multisig address**
Several tutorials are available on this topic
`https://www.soroushjp.com/2014/12/20/bitcoin-multisig-the-har`

Copyright 2019, Gunnar Stefansson (editor)
This work is licensed under the Creative Commons Attribution-ShareAlike License. To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/1.0/ or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.