The transaction

crypto251.0 Cryptocurrency and the Smileycoin

Gunnar Stefansson (editor)

November 29, 2020

Background

The concept of a transaction as a description of transfer of funds is simple but not enough
How does one guarantee that the funds are not sent twice?
How does one ensure that the sender is authorised to spend the funds?
To see how this is done we need to look inside the transactions and study their structure

A typical transaction

Consider a specific SMLY transaction, eg e870614afe3cb9fde97566b024a72f11d22ce08dbd89a971655b15f71d6e20
which can be seen in block 332353, at
https://chainz.cryptoid.info/smly/block.dws?33e1da4929acfa4cbf
A summary of the transaction is given at
https://chainz.cryptoid.info/smly/tx.dws?e870614afe3cb9fde9756
but we want to see some of the detail.


```
{
   "txid": "e870614afe3cb9fde97566b024a72f11d22ce08dbd89a971655b15f71d6e203b",
   "version": 1,
   "locktime": 0,
   "vin": [
      {
         "txid": "cc3b743938e485578315b2f6848c1a416c917585ea2f75d5d3e09f21a95008b0",
         "vout": 0,
         "scriptSig": {
            "asm": "304502210094115 66e0a7965fc5b8a73ca788a52c4fc0d4f6eaf3089812e55e7def520c79502205d7db54c0851d650d80e9236ecfbe9a94ea0b17976255f2170
            "hex": "48304502210094115 66e0a7965fc5b8a73ca788a52c4fc0d4f6eaf3089812e55e7def520c79502205d7db54c0851d650d80e9236ecfbe9a94ea0b17976255f21
         },
         "sequence": 4294967295
      }
   ],
   "vout": [
      {
         "value": 103018.4900285,
         "n": 0,
         "scriptPubKey": {
            "asm": "OP_DUP OP_HASH160 a4d6b6e2e262e97590564a24b523d993765525fb OP_EQUALVERIFY OP_CHECKSIG",
            "hex": "76a914a4d6b6e2e262e97590564a24b523d993765525fb88ac",
            "reqSigs": 1,
            "type": "pubkeyhash",
            "addresses": [
               "BKUfenzHcFab9pCzz64XYYbiex1VUcQi6J"
            ]
         }
      },
      {
         "value": 61593.68789149,
         "n": 1,
         "scriptPubKey": {
            "asm": "OP_DUP OP_HASH160 96cd095b0654b8945af7b196e614b0f77932326b OP_EQUALVERIFY OP_CHECKSIG",
            "hex": "76a91496cd095b0654b8945af7b196e614b0f77932326b88ac",
            "reqSigs": 1,
            "type": "pubkeyhash",
            "addresses": [
               "BJCSbgLL9q8729BfTz4YN51BqFcuLnqWNw"
            ]
         }
      }
   ],
   "blockhash": "33e1da4929acfa4cbf2dceb28f469c5179e67077a9642a6c9e2e3bfe590cce1a",
   "confirmations": 51
```

Inside the transaction: The output

Consider the outputs from transaction e870614afe3cb9fde97566b024a72f11d22ce08dbd89a971655b15f71d6e203b

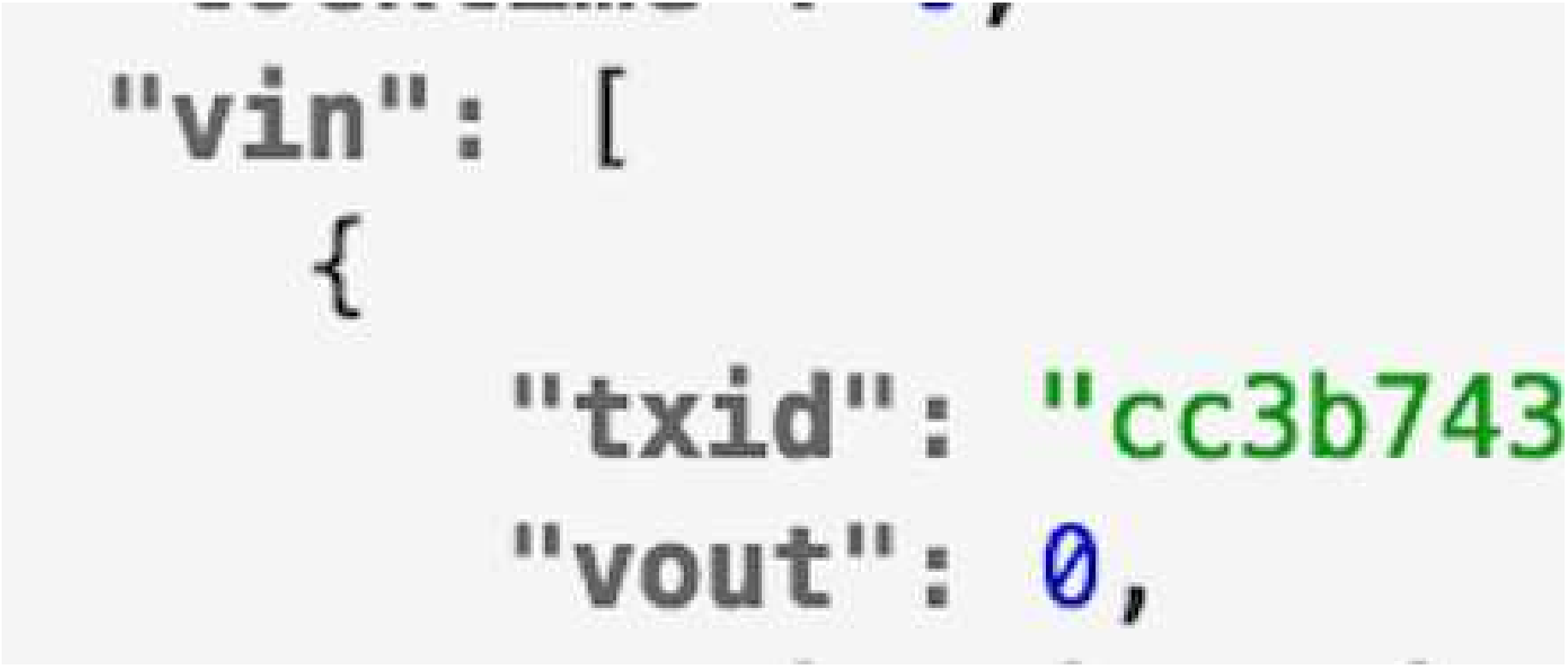

```
"vout": [
   {
      "value": 103018.4900285,
      "n": 0,
      "scriptPubKey": {
         "asm": "OP_DUP OP_HASH160 a4d6b6e2e262e9
         "hex": "76a914a4d6b6e2e262e97590564a24b5
         "reqSigs": 1,
         "type": "pubkeyhash",
         "addresses": [
            "BKUfenzHcFab9pCzz64XYYbiex1VUcQi6J"
         ]
      }
   },
   {
      "value": 61593.68789149,
      "n": 1,
```


The outputs form two UTXOs: “n”=0 og “n”=1

These can later be referenced, e.g. as UTXO n=0 from Tx=e870614afe3cb9fde97566b024a72f11d22ce08dbd89a971655b15f71d6e203

Inside the transaction: The input

TxId: e870614afe3cb9fde97566b024a72f11d22ce08dbd89a971655b15f71d6e2



The input is only defined as an older output, which has not been spent, UTXO, as the following components:

- Start of input description: vin
- The input transaction refers to an older transaction: TxId
- "vout" refers to a numbered output ("n") in that transaction

The UTXO

We have seen that

- the input to transaction e870614afe3cb9fde97566b024a72f11d22ce08dbd89a971655b15f71d6e20 is
- the UTXO from transaction cc3b743938e485578315b2f6848c1a416c917585ea2f75d5d3e09f21a95008

To verify this we can look up that UTXO as seen in the handout.

Keys

Cryptocurrencies use cryptographic keys
For example, ownership is demonstrated using a combination of keys and addresses

- *public-private key pairs*
- *Private key -> public key -> address*

This will be explained in more detail later.

- An address can be freely distributed
- The private key is never disclosed
- A transaction can be signed using the private key
- A signature can be **verified** using the public key
- The public key is only disclosed when a transaction is spent

A **spending transaction** publishes the public key and a signature.

Spending the UTXO

The permission to spend the UTXO is determined by the programming code written into the transaction.
Will be described later in the course, but a short code snippet is seen in every transaction.
It is an incomplete snippet, usually with components of the form

- OP_DUP
- OP_HASH160
- a4d6b6e2e262e97590564a24b523d993765525fb
- OP_EQUALVERIFY
- OP_CHECKSIG

To spend this UTXO the spending transaction needs to prepend to this another snippet so the combined code can be run and will return "TRUE" and nothing else.
Completion of this particular snippet is done with

- signature

The transaction on the command line

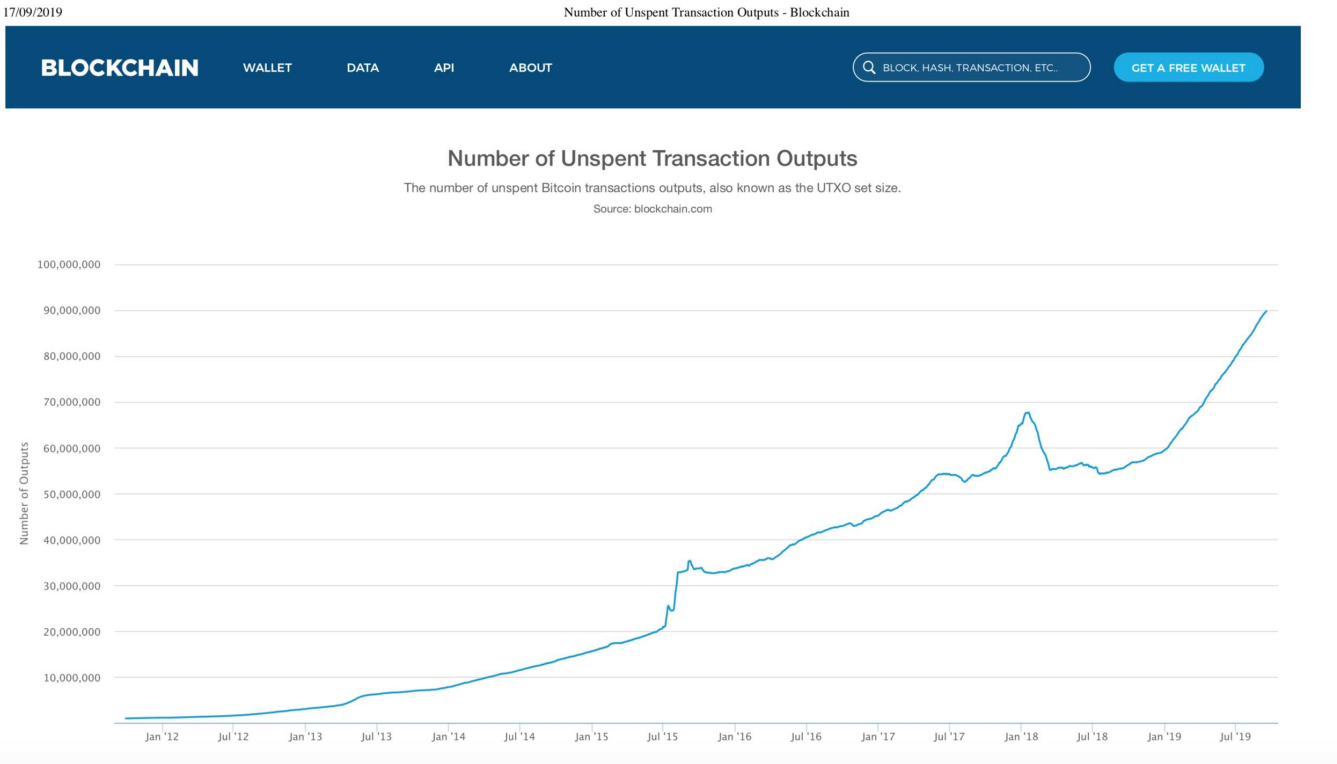
Step-by-step example of how to generate, sign, check, announce and inspect a transaction - to be done in detail in class

- `listunspent`
- `createrawtransaction '[{"txid" : "fbd60d37acfb30eba7153db741dce7d1ebf71c0ee0ec8802fba29e865 "vout" : 1}]' '{"B79tjNk8oZktdd7DLnznKXu9UA67GMWP9g" : 2000, "BHgx5rehx2Wkx4wME2DXwZAHL7KskUjXmK" : 2499}'`
- `signrawtransaction 01000000018fba0254869ea2fb0288ece00e1cf7ebd1e7dc41b73d15a7`
- `decoderawtransaction 01000000018fba0254869ea2fb0288ece00e1cf7ebd1e7dc41b73d15a7`
- `sendrawtransaction 01000000018fba0254869ea2fb0288ece00e1cf7ebd1e7dc41b73d15a7`
- `getrawtransaction e98b533cf3290fa58c23074aa0b1e273e25e4756321155e7ad165f2d3e`

The UTXO set

The UTXO set has a tendency to increase in size.

For Bitcoin (from `https://www.blockchain.com/charts/utxo-count?ti`



The transaction fee

Most transactions include a transaction fee
The fee is simply the difference between the inputs and the outputs
The fee is not explicitly specified

Manual transaction example - maintaining a fund

If a wallet is asked to send x SMLY it will just find some unspent transactions and aggregate them as input, send x to the destination and make a new address for the change, after taking some for the transaction fee.
There are many instances when one wants to do things differently. For example one may want to maintain all the funds under a single address for transparency.
This is how the Pineapple Fund worked and this is how the SmileyCoin Fund works.
`https://www.blockchain.com/btc/tx/081f68e146922f23039bf67a5bda`