CMPSC 390
Bitcoin Transactions

Janyl Jumadinova

Credit: Authors of “Bitcoin and Cryptocurrency Technologies”

January 28, 2021
Where we left off ...

**Bitcoin consensus**

- Append-only ledger.
- Decentralized consensus.
- Miners to validate transactions.
Bitcoin consensus

- Append-only ledger.
- Decentralized consensus.
- Miners to validate transactions.

assuming a currency exists to motivate miners!
UTXO Model

- Unspent Transaction Output Model.
UTXO Model

- Unspent Transaction Output Model.
- Transactions map inputs to outputs.
- An account holds a set of
  - Transactions contain signature of fund's owner.
  - Spending bitcoin is redeeming previous transaction outputs.
An account-based ledger (not Bitcoin)

Create 25 coins and credit to Janyl

Transfer 17 coins from Janyl to Will

Transfer 8 coins from Will to Kyrie

Transfer 5 coins from Kyrie to Janyl

Transfer 15 coins from Janyl to Zoe

SIMPLIFICATION: only one transaction per block

might need to scan backwards until genesis!

is this valid?
Transaction-based ledger (Bitcoin)

1. Inputs: Ø
   Outputs: 25.0 → Janyl

2. Inputs: 1[0]
   Outputs: 17.0 → Will, 8.0 → Janyl

3. Inputs: 2[0]
   Outputs: 8.0 → Kyrie, 7.0 → Will

4. Inputs: 2[1]
   Outputs: 6.0 → Zoe, 2.0 → Janyl

SIMPLIFICATION: only one transaction per block

we implement this with hash pointers

finite scan to check for validity

is this valid?
Merging Value

1. Inputs: ...
   Outputs: 17.0 → Will, 8.0 → Janyl
   SIGNED(Janyl)

2. Inputs: 1[1]
   Outputs: 6.0 → Kyrie, 2.0 → Will
   SIGNED(Kyrie)

3. Inputs: 1[0], 2[1]
   Outputs: 19.0 → Will
   SIGNED(Will)

SIMPLIFICATION: only one transaction per block
Joint Payments

1. Inputs: ...
   Outputs: 17.0→Will, 8.0→Janyl
   SIGNED(Janyl)

2. Inputs: 1[1]
   Outputs: 6.0→Kyrie, 2.0→Will
   SIGNED(Kyrie)

3. Inputs: 2[0], 2[1]
   Outputs: 8.0→Zoe
   SIGNED(Kyrie), SIGNED(Will)

SIMPLIFICATION: only one transaction per block
Bitcoin transaction

```
{
  "hash":"5a42590fbe0a90ee8e8747244d6c84f0db1a3a24e8f1b95b10c9e050990b8b6b",
  "ver":1,
  "vin_sz":2,
  "vout_sz":1,
  "lock_time":0,
  "size":404,
  "in":[
    {
      "prev_out":{
        "hash":"3be4ac9728a0823cf5e2deb2e86fc0bd2aa503a91d307b42ba76117d792b060",
        "n":0
      },
      "scriptSig":"30440…"
    },
    {
      "prev_out":{
        "hash":"7508e6ab259b4df0fd5147bab0c949d81473db4518f81af5c3f52f91ff6b34e",
        "n":0
      },
      "scriptSig":"3f3a4ce81…"
    }
  ],
  "out":[
    {"value":"10.12287097",
      "scriptPubKey":"OP_DUP OP_HASH160 69e02e18b5705a05dd6b28ed517716c894b3d42e OP_EQUALVERIFY OP_CHECKSIG"
    }
  ]
}``
```
Bitcoin transaction: metadata

```json
{
  "hash": "5a42590...b8b6b",
  "ver": 1,
  "vin_sz": 2,
  "vout_sz": 1,
  "lock_time": 0,
  "size": 404,
  ...
}
```
Bitcoin transaction: inputs

```
"in": [  
  
  
  
  previous transaction

  ],

signature

  "scriptSig": "30440....3f3a4ce81",

(more inputs)

  "prev_out": {  
  "hash": "3be4...80260",  
  "n": 0  
  },

],
```
Bitcoin transaction: outputs

```
"out": [
    {
        "value": "10.12287097",
        "scriptPubKey": "OP_DUP OP_HASH160 69e...3d42e
                            OP_EQUALVERIFY OP_CHECKSIG"
    },
    ...
    (more outputs)
]
```
Output “addresses” are actually scripts.
Output “addresses” are actually scripts.

Input “addresses” are also scripts.
Bitcoin Script

- Output “addresses” are actually scripts.
- Input “addresses” are also scripts.

```
30440220...
0467d2c9...
```

```
OP_DUP
OP_HASH160
69e02e18...
OP_EQUALVERIFY OP_CHECKSIG
```
Bitcoin Scripting Language ("Script")

- Built for Bitcoin (inspired by Forth).
- Simple, compact.
- Support for cryptography.
- Stack-based.
- Limits on time/memory.
- No looping.
Bitcoin Script Example

<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash?> OP_EQUALVERIFY OP_CHECKSIG
Bitcoin Script Example

<sig>  <pubKey>  OP_DUP  OP_HASH160  <pubKeyHash?>  
OP_EQUALVERIFY  OP_CHECKSIG

<pubKeyHash?>
<pubKeyHash>
<pubKey>
true
Bitcoin Scripting instructions

256 opcodes total
- Arithmetic
- If/then
- Logic/data handling
Bitcoin Scripting instructions

256 opcodes total

- Arithmetic
- If/then
- Logic/data handling
- Hashes. Signature verification. Multi-signature verification
Bitcoin Blocks

Hash chain of blocks

Hash tree (Merkle tree) of transactions in each block
Bitcoin Blocks

```
{
  "hash" : "000000000000000001aadd2...",
  "ver" : 2,
  "prev_block" : "00000000000000003043..",
  "time" : 1391279636,
  "bits" : 419558700,
  "nonce" : 459459841,
  "mrkl_root" : "8977e...",
  "n_tx" : 354,
  "size" : 181520,
  "tx" : [
    ...
  ],
  "mrkl_tree" : [
    "6bd5eb25...",
    ...
    "89776cde..."
  ]
}
```
```json
{
    "hash":"0000000000000000000000000000000000000000000000000000000000000000",
    "ver":2,
    "prev_block":"00000000000000000000000000000000",
    "time":1391279636,
    "bits":419558700,
    "nonce":459459841,
    "mrkl_root":"89776…
    ...
}
```
```json
{
    "hash":"000000000000000001aad2…",
    "ver":2,
    "prev_block":"000000000000000003043…",
    "time":1391279636,
    "bits":419558700,
    "nonce":459459841,
    "mrkl_root":"89776…",
    ...
}
```