Mechanics of Bitcoin

Bitcoin transactions

An account-based ledger (*not* Bitcoin)

time

Create 25 coins and credit to Alice$_{\text{ASSERTED BY MINERS}}$
Transfer 17 coins from Alice to Bob$_{\text{SIGNED(Alice)}}$
Transfer 8 coins from Bob to Carol$_{\text{SIGNED(Bob)}}$
Transfer 5 coins from Carol to Alice$_{\text{SIGNED(Carol)}}$
Transfer 15 coins from Alice to David$_{\text{SIGNED(Alice)}}$

is this valid?

might need to scan backwards until genesis!

SIMPLIFICATION: only one transaction per block

A transaction-based ledger (Bitcoin)

time

1

Inputs: Ø

Outputs: 25.0→Alice

2

Inputs: 1[0]

Outputs: 17.0→Bob, 8.0→Alice

SIGNED(Alice)

change address

3

Inputs: 2[0]

Outputs: 8.0→Carol, 9.0→Bob

SIGNED(Bob)

4

Inputs: 2[1]

Outputs: 6.0→David, 2.0→Alice

SIGNED(Alice)

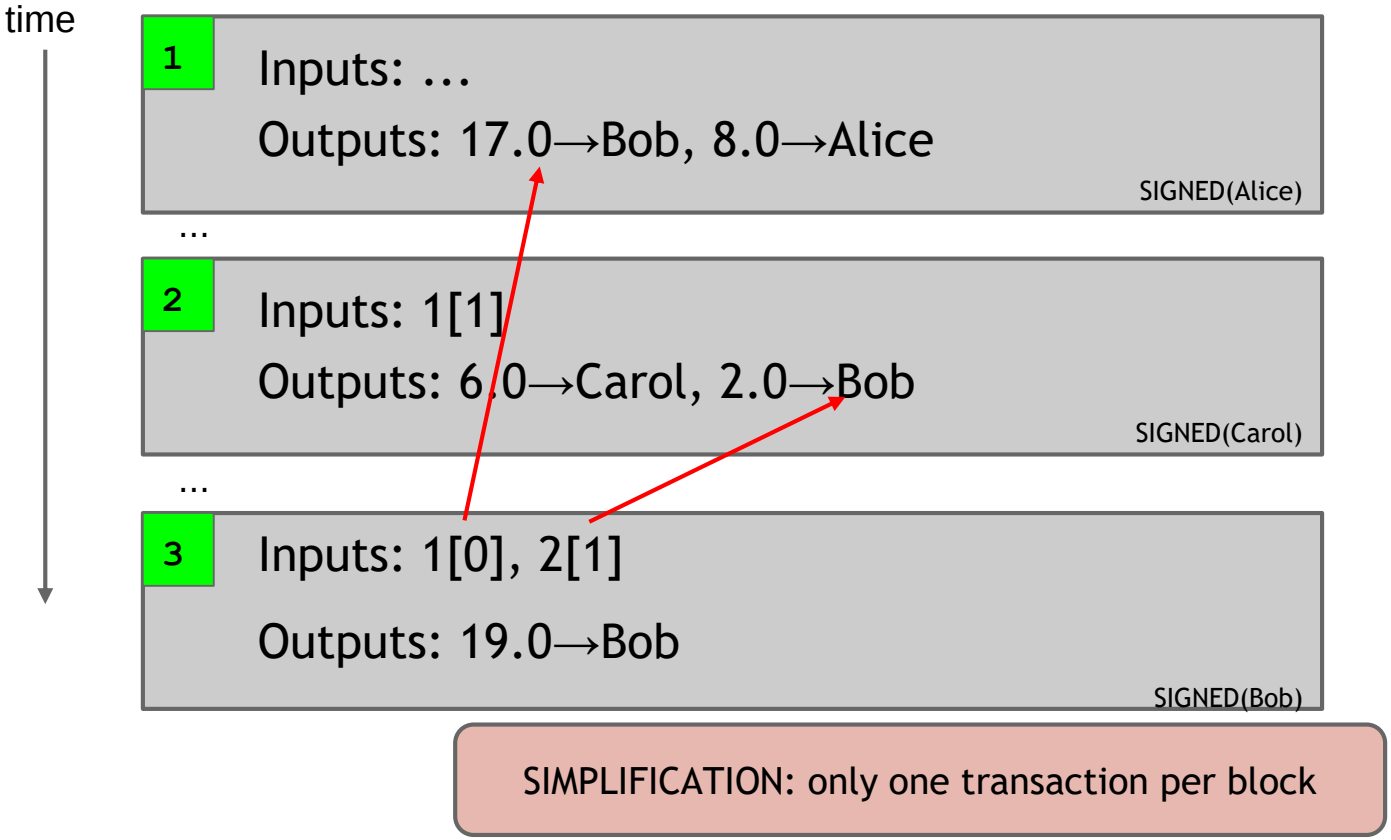
we implement this with hash pointers

finite scan to check for validity

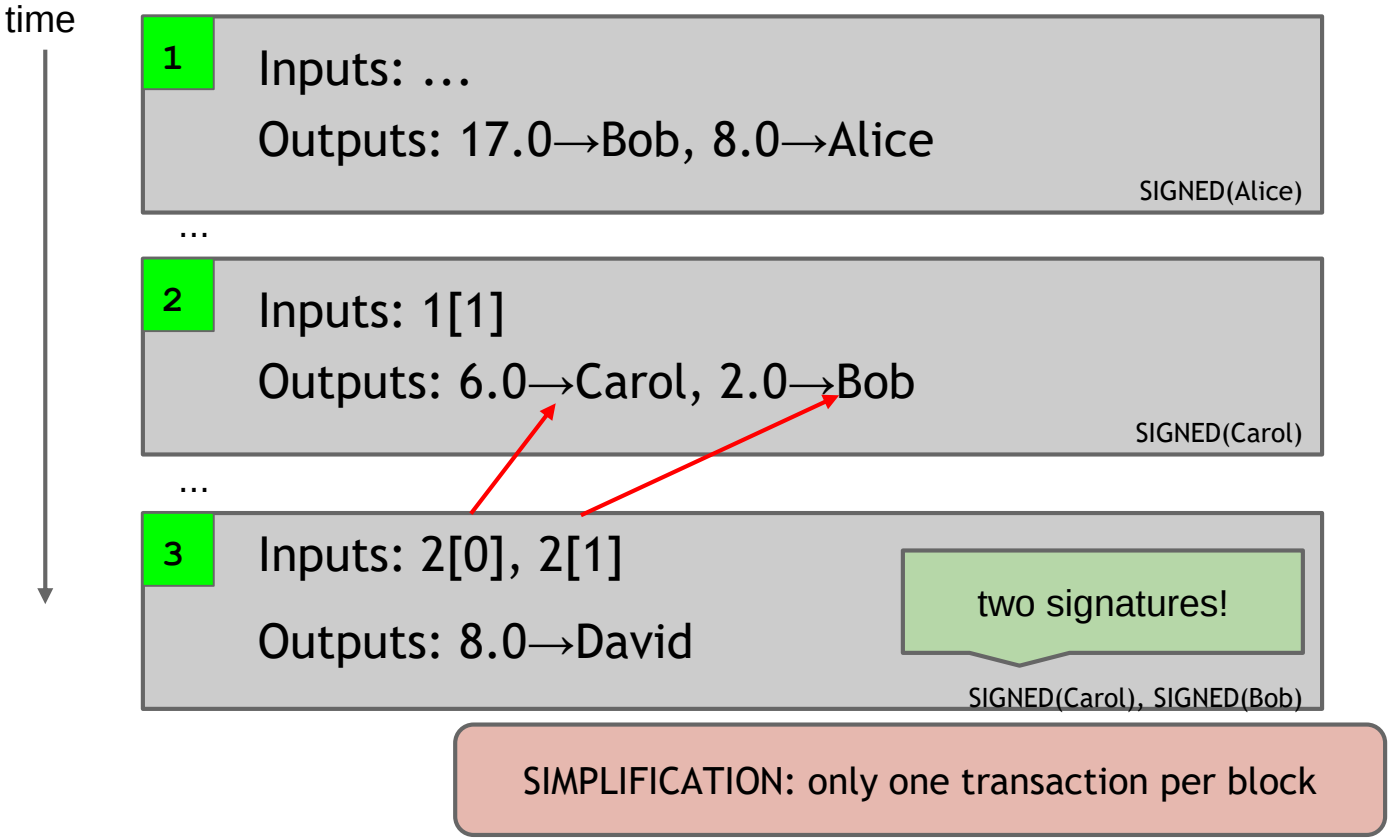
is this valid?

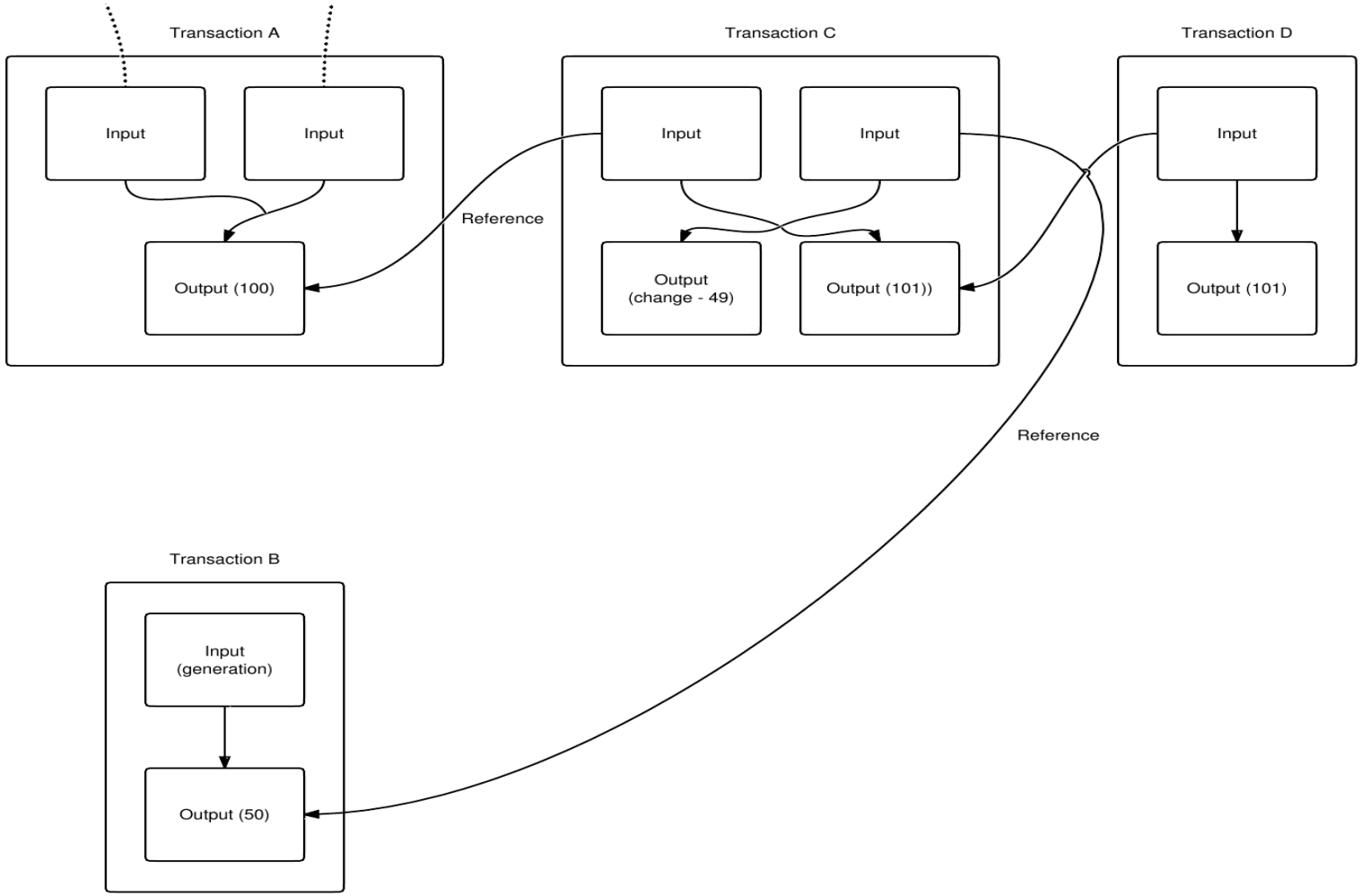
SIMPLIFICATION: only one transaction per block

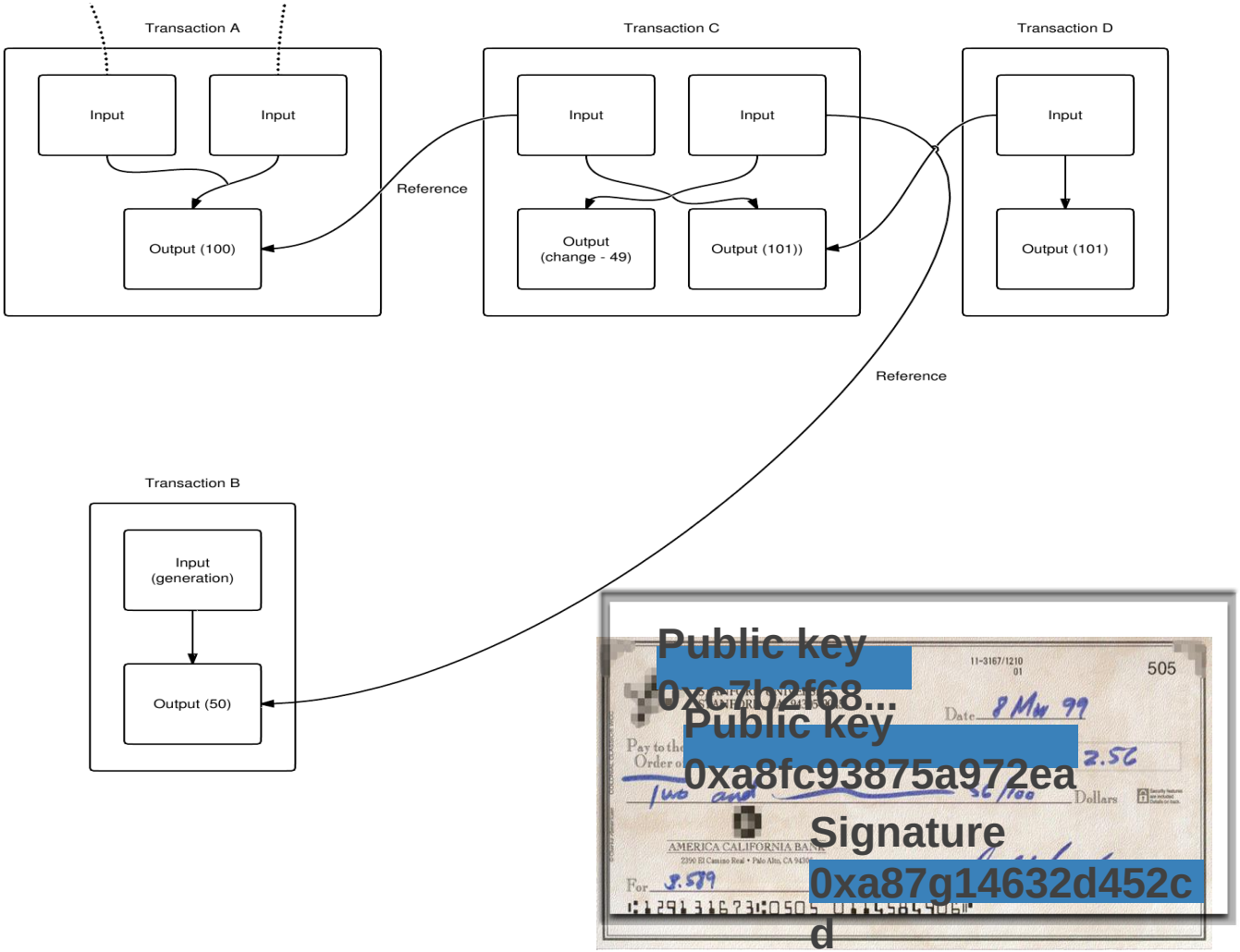
Merging value



Joint payments




Transaction A
Input
Input
Output (100)
Reference
Transaction C
Input
Input
Output
(change - 49)
Output (101))
Transaction D
Input
Output (101)
Reference
Transaction B
Input
(generation)
Output (50)


Transaction A
Input
Input
Output (100)
Reference
Transaction C
Input
Input
Output
(change - 49)
Output (101))
Transaction D
Input
Output (101)
Reference
Transaction B
Input
(generation)
Output (50)
Public key
0xc7b2f68...
Public key
0xa8fc93875a972ea
Signature
0xa87g14632d452c
d
11-3167/1210
01
505
Date 8 Mar 99
Pay to the
Order of
2.56
Two and 56/100 Dollars
AMERICA CALIFORNIA BANK
2390 El Camino Real • Palo Alto, CA 94306
For 8.589

Transaction validity Checking
Using Bitcoin scripts

The real deal: a Bitcoin transaction

metadata

input(s)

output(s)


```
{
    "hash":"5a42590fbe0a90ee8e8747244d6c84f0db1a3a24e8f1b95b10c9e050990b8b6b",
    "ver":1,
    "vin_sz":2,
    "vout_sz":1,
    "lock_time":0,
    "size":404,
    "in":[

      {
        "prev_out":{
          "hash":"3be4ac9728a0823c[illegible]b2e86fc0bd2aa503a91d307b42ba76117d79280260",
          "n":0
        },

          "scriptSig":"30440..."
      },
      {
        "prev_out":{
          "hash":"7508e6ab259b4df0fd5147bab[illegible]a1473db4518f81afc5c3f52f91ff6b34e",
          "n":0
        },
        "scriptSig":"3f3a4ce81...."
      }
    ],

    "out":[
      {
        "value":"10.12287097",
        "scriptPubKey":"OP_DUP OP_HASH160 69e02e18b5705a05dd6b28ed517716c894b3d42e OP_EQUALVERIFY OP_CHECKSIG"
      }
    ]
}
```


Owner's Signature

Owner's Signature

Recipient

Alice → Bob

Transaction Created by Alice and already in Blockchain

metadata

input(s)

output(s)


```
{
  "hash":"5a42590fbe0a90ee8e8747244d6c84f0db1a3a24e8f1b95b10c9e050990b8b6b",
  "ver":1,
  "vin_sz":2,
  "vout_sz":1,
  "lock_time":0,
  "size":404,
  "in":[
   {
    "prev_out":{
     "hash":"3be4ac9728a0823cf5e2deb2e86fc0bd2aa503a91d307b42ba76117d79280260",
     "n":0
    },
     "scriptSig":"30440..."
   },
   {
    "prev_out":{
     "hash":"7508e6ab259b4df0fd5147bab0c949d81473db4518f81afc5c3f52f91ff6b34e",
     "n":0
    },
    "scriptSig":"3f3a4ce81...."
   }
  ],
  "out":[
   {
    "value":"10.12287097",
    "scriptPubKey":"OP_DUP OP_HASH160 69e02e18b5705a05dd6b28ed517716c894b3d42e OP_EQUALVERIFY OP_CHECKSIG"
   }
  ]
```


Signed by Alice

Recipient is Bob

2

Bob → Carol

metadata

input(s)

output(s)


```
{
  "hash":"5a42590fbe0a90ee8e8747244d6c84f0db1a3a24e8f1b95b10c9e050990b8b6b",
  "ver":1,
  "vin_sz":2,
  "vout_sz":1,
  "lock_time":0,
  "size":404,
  "in":[
   {
    "prev_out":{
     "hash":"3be4ac9728a0823cf5e2deb2e86fc0bd2aa503a91d307b42ba76117d79280260",
     "n":0
    },
     "scriptSig":"30440..."
   },
   {
    "prev_out":{
     "hash":"7508e6ab259b4df0fd5147bab0c949d81473db4518f81afc5c3f52f91ff6b34e",
     "n":0
    },
    "scriptSig":"3f3a4ce81...."
   }
  ],
  "out":[
   {
    "value":"10.12287097",
    "scriptPubKey":"OP_DUP OP_HASH160 69e02e18b5705a05dd6b28ed517716c894b3d42e OP_EQUALVERIFY OP_CHECKSIG"
   }
  ]
```


Signed by Bob

Recipient is Carol

1

New Transaction created by Bob

Input “addresses” are *also* scripts

scriptSig

30440220…
0467d2c9…

1

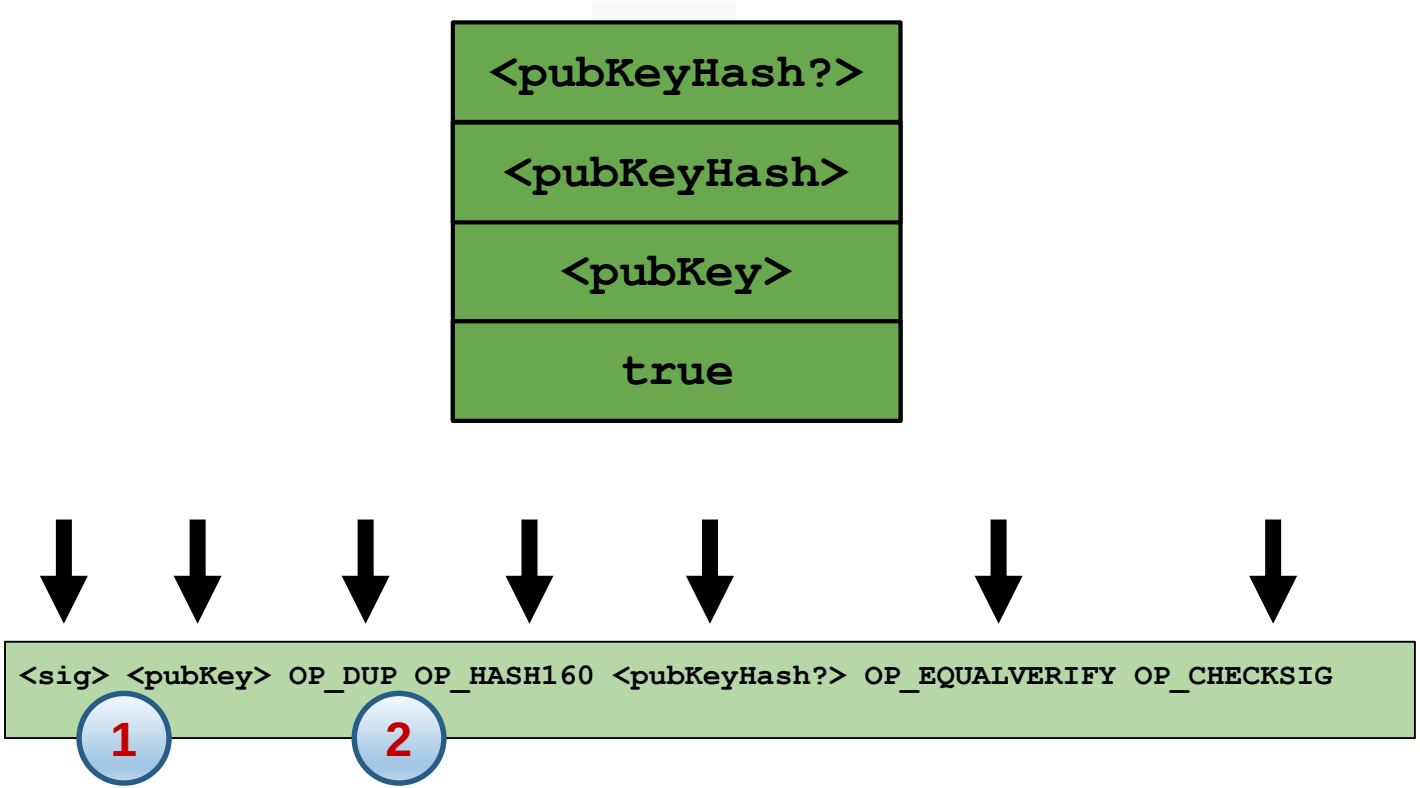
scriptPubKey

OP_DUP
OP_HASH160
69e02e18…
OP_EQUALVERIFY OP_CHECKSIG

2

TO VERIFY: Concatenated script must execute completely with no errors

Bitcoin script execution example



The real deal: a Bitcoin transaction

metadata

input(s)

output(s)


```
{
    "hash":"5a42590fbe0a90ee8e8747244d6c84f0db1a3a24e8f1b95b10c9e050990b8b6b",
    "ver":1,
    "vin_sz":2,
    "vout_sz":1,
    "lock_time":0,
    "size":404,
    "in":[
      {
        "prev_out":{
          "hash":"3be4ac9728a0823cf5e2deb2e86fc0bd2aa503a91d307b42ba76117d79280260",
          "n":0
        },

          "scriptSig":"30440..."
      },
      {
        "prev_out":{
          "hash":"7508e6ab259b4df0fd5147bab0c949d81473db4518f81afc5c3f52f91ff6b34e",
          "n":0
        },
        "scriptSig":"3f3a4ce81...."
      }
    ],
    "out":[
      {
        "value":"10.12287097",
        "scriptPubKey":"OP_DUP OP_HASH160 69e02e18b5705a05dd6b28ed517716c894b3d42e OP_EQUALVERIFY OP_CHECKSIG"
      }
    ]
```

The real deal: transaction metadata

transaction hash

housekeeping

“not valid before”

housekeeping


```
{
    "hash":"5a42590...b8b6b",
      "ver":1,
      "vin_sz":2,
      "vout_sz":1,
      "lock_time":0,
      "size":404,

...
}
```

The real deal: transaction inputs

previous transaction

signature

(more inputs)

```
"in":[
  {
    "prev_out":{
      "hash":"3be4...80260",
      "n":0
    },
"scriptSig":"30440....3f3a4ce81"
  },
...
],
```

The real deal: transaction outputs

output value

recipient address??

(more outputs)

```
"out":[
    {
      "value":"10.12287097",
      "scriptPubKey":"OP_DUP OP_HASH160 69e...3d42e
OP_EQUALVERIFY OP_CHECKSIG"
    },
        ...
  ]
```

See for yourself!

Transaction View information about a bitcoin transaction

151b750d1f13e76d84e82b34b12688811b23a8e3119a1cba4b4810f9b0ef408d

1KryFUt9tXHvaoCYTNPbqpWPJKQ717YmL5	1KvrdrQ3oGqMAiDTMEYCcdDSnVaGNW2YZh	1.0194 BTC
	1KryFUt9tXHvaoCYTNPbqpWPJKQ717YmL5	3.458 BTC

9 Confirmations 4.4774 BTC

Summary	
Size	257 (bytes)
Received Time	2014-08-05 01:55:25
Included In Blocks	314018 (2014-08-05 02:00:40 +5 minutes)
Confirmations	9 Confirmations
Relayed by IP	Blockchain.info
Visualize	View Tree Chart

Inputs and Outputs	
Total Input	4.4775 BTC
Total Output	4.4774 BTC
Fees	0.0001 BTC
Estimated BTC Transacted	1.0194 BTC
Scripts	Show scripts & coinbase

blockchain.info (and many other sites)

Transaction View information about a bitcoin transaction

9cd03f530b83b67eee52bbbd2e9067e79e31513cffb5535c7463d96a8c5d96ae **Transaction ID (TX ID)**

1J29P1ceAfJHpG2jPQN1QxdHgCGEnLHd3u

Input Address

34auLDAG8skCooDAPpWFm69JuDz3rYnaDG	0.1 BTC
16XAfbSNEkkkwshkcusFJS4JxyHs74nudp	0.77 BTC
1AW2YoNvhAwatTjUcnzYWPETb3WSonZUD8	0.58 BTC
1L5a3gfb8FNJQn2MexVEjSzvXkXCp7mEBU	2.87094476 BTC

Output Addresses

1 Confirmations | 4.32094476 BTC

Block Information:

Summary	
Size	292 (bytes)
Weight	1168
Received Time	2018-02-02 07:45:17
Included In Blocks	507234 (2018-02-02 08:12:38 + 27 minutes)
Confirmations	1 Confirmations
Visualize	View Tree Chart

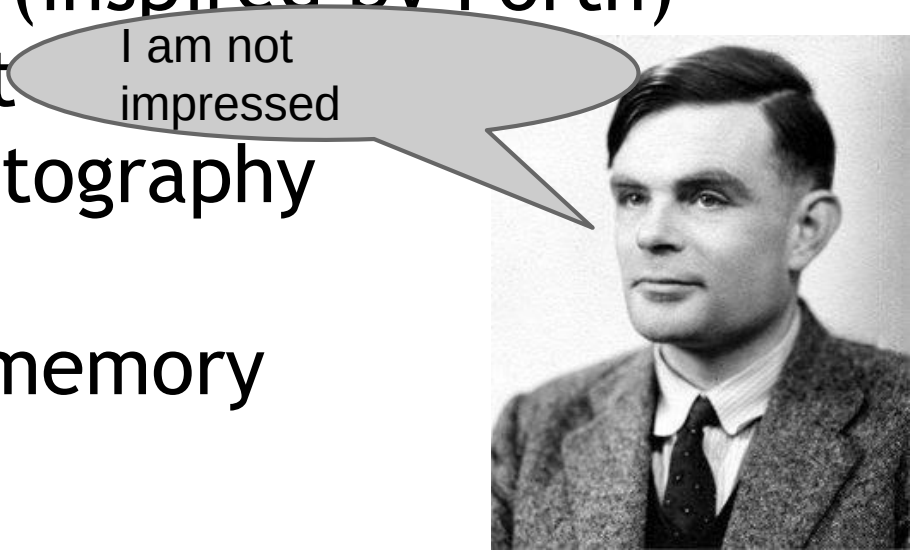
Transaction information:

Inputs and Outputs	
Total Input	4.32123876 BTC
Total Output	4.32094476 BTC
Fees	0.000294 BTC
Fee per byte	100.685 sat/B
Fee per weight unit	25.171 sat/WU
Estimated BTC Transacted	0.1 BTC
Scripts	Show scripts & coinbase

Bitcoin scripting language (“Script”)

Design goals

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



OP_DUP	Duplicates the top item on the stack
OP_HASH160	Hashes twice: first using SHA-256 and then RIPEMD-160
OP_EQUALVERIFY	Returns true if the inputs are equal. Returns false and marks the transaction as invalid if they are unequal
OP_CHECKSIG	Checks that the input signature is a valid signature using the input public key for the hash of the current transaction
OP_CHECKMULTISIG	Checks that the *k* signatures on the transaction are valid signatures from *k* of the specified public keys.

Figure 3.6 a list of common Script instructions and their functionality.

Bitcoin script instructions

256 opcodes total (15 disabled, 75 reserved)

- Arithmetic
- If/then
- Logic/data handling
- Crypto!
    - Hashes
    - Signature verification
    - Multi-signature verification

Proof-of-burn

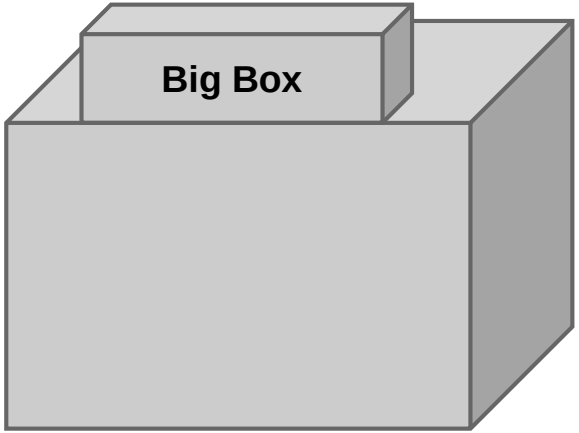
nothing’s going to redeem that ☹

OP_RETURN
<arbitrary data>

Should senders specify scripts?

I’m ready to pay for my purchases!

Cool! Well we’re using MULTISIG now, so include a script requiring 2 of our 3 account managers to approve. Don’t get any of those details wrong. Thanks for shopping at Big Box!



Idea: use the hash of redemption script

<signature>
< redemption script >

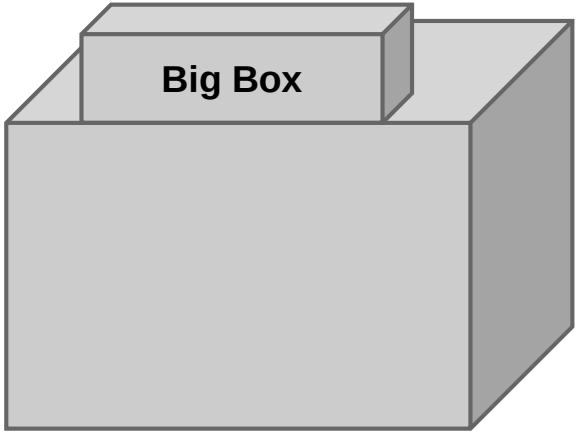
OP_HASH160
<hash of redemption script>
OP_EQUAL

“Pay to Script Hash”

Pay to script hash

I’m ready to pay for my purchases!

Great! Here’s our address: 0x3454

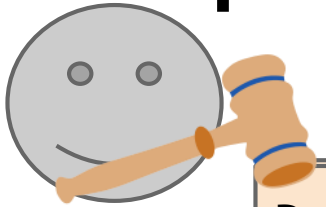


Applications of Bitcoin scripts

Example 1: Escrow transactions

(normal case)

(disputed case)



Judy

Pay *x* to Alice

SIGNED(ALICE, JUDY)

Alice

Bob

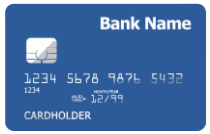
Pay *x* to 2-of-3 of Alice, Bob, Judy (MULTISIG)

SIGNED(ALICE)

Bob doesn't want to ship until after Alice pays.

Example 2: Green addresses

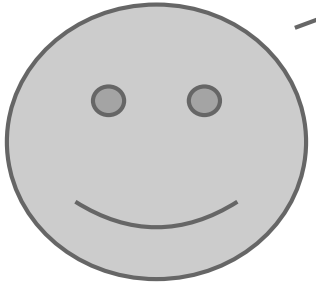
InstaWallet, Mt. Gox Collapsed!



It's me, Alice! Could you make out a green payment to Bob?

Bank

You may Trust me. We do not do double spend!



Alice

Pay x to Bob, y to Bank

No double spend

SIGNED(BANK)

Bob

PROBLEM: Alice wants to pay Bob.
Bob can't wait 6 verifications to guard against double-spends, or is offline completely.

Example 3: Efficient micro-payments

What if Bob never signs??

all of these could be double-spends!

Input: *x*; Pay 42 to Bob, 58 to Alice
SIGNED(ALICE) SIGNED(BOB)

...

Alice demands a timed refund transaction before starting

Input: *x*; Pay 100 to Alice, LOCK until time *t*
SIGNED(ALICE) SIGNED(BOB)

I'm done!

Input: *x*; Pay 03 to Bob, 97 to Alice
SIGNED(ALICE)_____

I'll publish!

Input: *x*; Pay 02 to Bob, 98 to Alice
SIGNED(ALICE)_____

Input: *x*; Pay 01 to Bob, 99 to Alice
SIGNED(ALICE)_____

PROBLEM: Alice wants to pay Bob for each

Input: *y*; Pay 100 to Bob/Alice (MULTISIG)
SIGNED(ALICE)

Alice

Bob

lock_time


```
{
    "hash":"5a42590...b8b6b",
     "ver":1,
     "vin_sz":2,
     "vout_sz":1,
     "lock_time":315415,
     "size":404,
...
}
```


Block index or real-world timestamp before which this transaction can’t be published

OP_CHECKMULTISIG

- Built-in support for joint signatures
- Specify *n* public keys
- Specify *t*
- Verification requires *t* signatures

BUG ALERT: Extra data value popped from the stack and ignored

Bitcoin scripts in practice (as of 2014)

- Most nodes whitelist known scripts
- 99.9% are simple signature checks
- ~0.01% are MULTISIG
- ~0.01% are Pay-to-Script-Hash
- Remainder are errors, proof-of-burn

More advanced scripts

- Multiplayer lotteries
- Hash pre-image challenges
- Coin-swapping protocols

“Smart contracts”