Blockchains & Cryptocurrencies

Bitcoin Mechanics

Instructor: Matthew Green & Abhishek Jain
Johns Hopkins University - Spring 2023

*Many slides based on NBFMG

Last Time (Matt)

Last Time (Matt)

- Proof of Work (PoW) puzzles

Last Time (Matt)

- Proof of Work (PoW) puzzles
- Consensus mechanism in Bitcoin using PoW

Last Time (Matt)

- Proof of Work (PoW) puzzles
- Consensus mechanism in Bitcoin using PoW
- Difficulty Parameter Adjustability

Last Time (Matt)

- Proof of Work (PoW) puzzles
- Consensus mechanism in Bitcoin using PoW
- Difficulty Parameter Adjustability
- Longest Chain Rule

Today

Today

- Bitcoin Transaction Format

Today

- Bitcoin Transaction Format
- Simple Smart-Contracts in Bitcoin

But first: wrapping up last lecture

…What if there's a collision?

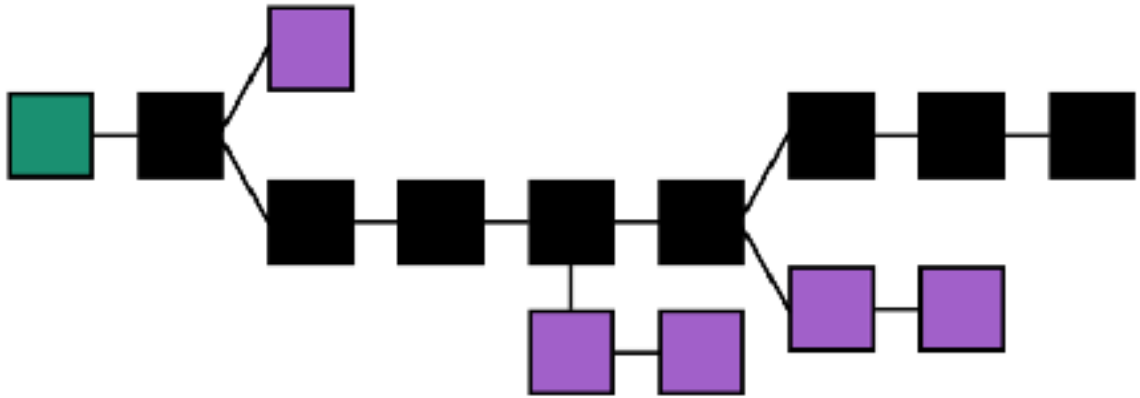


Image CC-BY-3 Theymos taken from the Bitcoin wiki

“Longest chain rule”

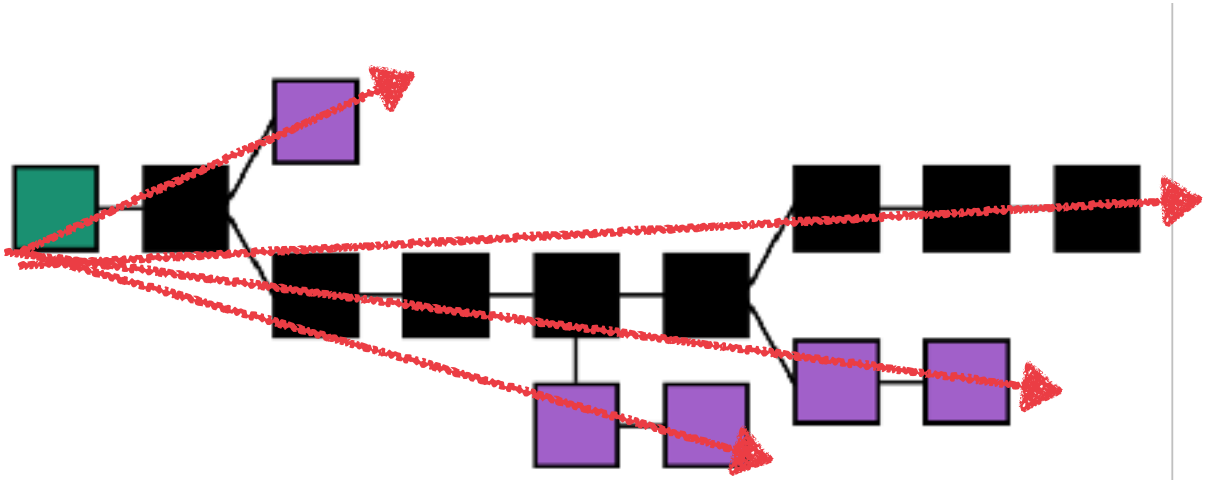


Image CC-BY-3 Theymos taken from the Bitcoin wiki

A: "Chain with most hashwork"

Bitcoin doesn't exactly use the longest chain rule

Instead, it employs a calculation that takes block difficulty into account

Each block has a difficulty. Convert to expected # of hashes to find block. Total these values. Chain with largest total is "longest".

Most of the time, this is equivalent to longest chain

This is good and bad

Good: if we experience a "chain fork" and the network is connected (I.e., not totally partitioned), then eventually we will learn about both forks

Good: if the "hash power" behind the two chains is unequal, we will probably end up with one chain getting longer

Even if the hash power is equal, the inherent randomness of the puzzle (PoW) will likely cause an advantage

As one chain grows longer, other nodes will adopt it, and start adding to it

What's the bad?

When a chain becomes longer than the "current chain" a node thinks is the longest chain, they must abandon that older chain

Finality

"Finality is the assurance or guarantee that cryptocurrency transactions cannot be altered, reversed, or canceled after they are completed."

https://academy.binance.com/glossary/finality

Finality

"Finality is the assurance or guarantee that cryptocurrency transactions cannot be altered, reversed, or canceled after they are completed."

Bitcoin's finality is probabilistic (and computational)

Reorganizations become less probable (and more expensive) over time, but they never become impossible*

How many blocks can the adversary make?

Consider an adversary that controls a r-fraction of the hash power

How many blocks in expectation can they build in a t-block window?

What is the probability that they dominate that t-block window?

Look at papers (reading list) for detailed analysis

How do we incentivize mining?

How do we incentivize mining?

Two answers to this question in Bitcoin:

1. "Transaction fees" Each transaction has a "tip" that can be collected by the node who mines it into a block (incentivizes inclusion of transactions)

2. "Block reward" 50/25/12.5/6.3/… BTC made from scratch (in a special Coinbase transaction) and given to the miner

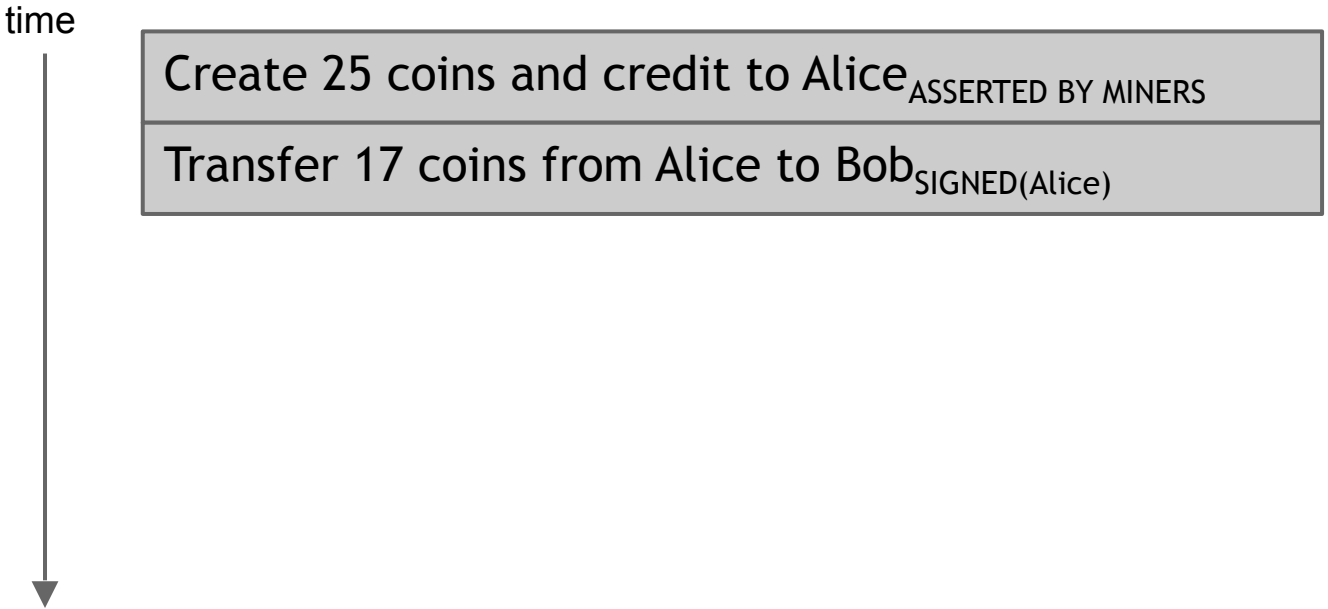
Bitcoin transactions

An account-based ledger (*not* Bitcoin)

Create 25 coins and credit to Alice ASSERTED BY MINERS

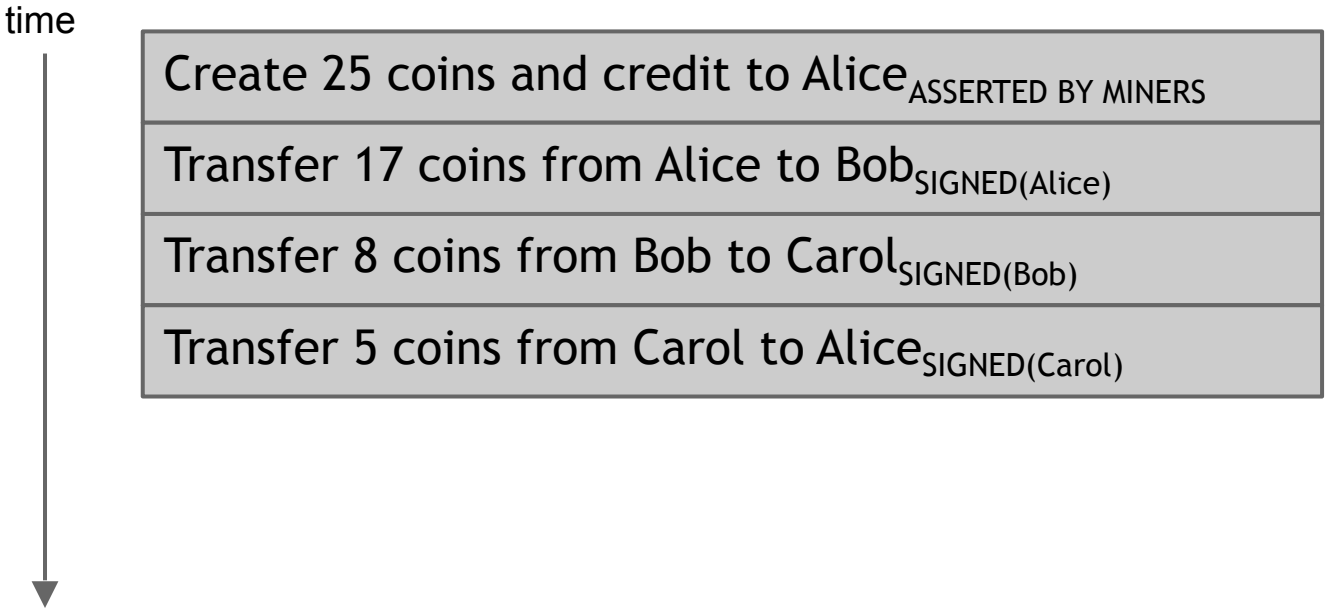
SIMPLIFICATION: only one transaction per block

An account-based ledger (*not* Bitcoin)



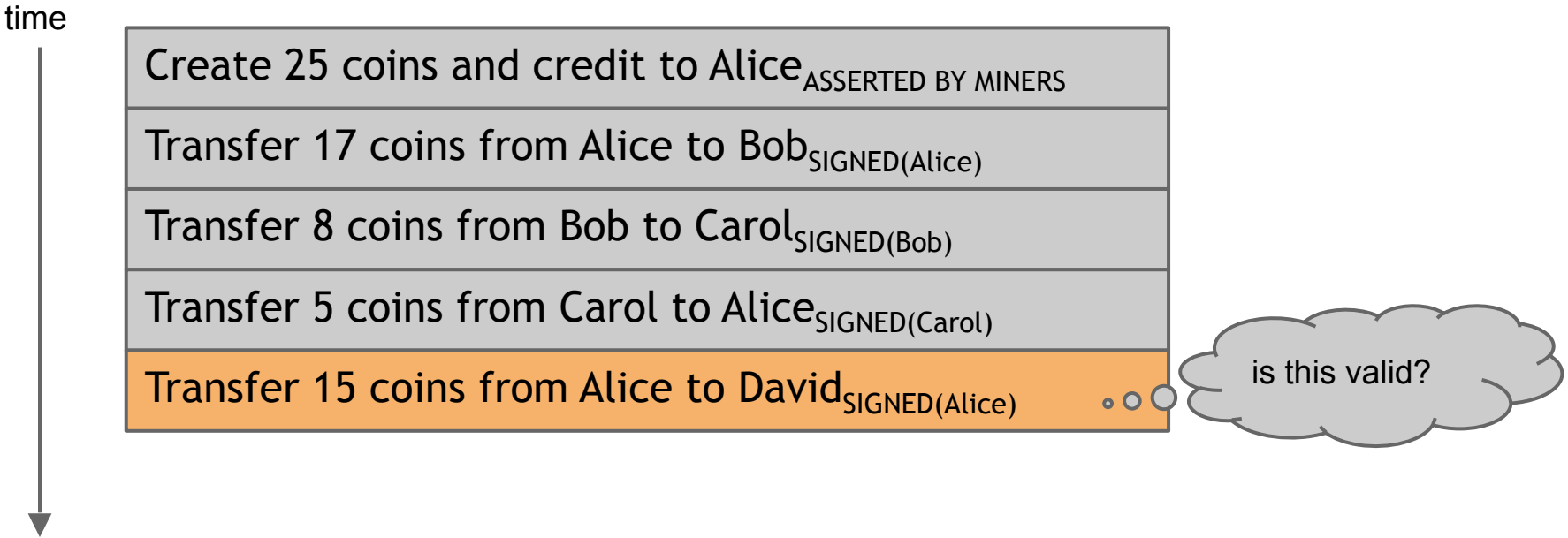
SIMPLIFICATION: only one transaction per block

An account-based ledger (*not* Bitcoin)



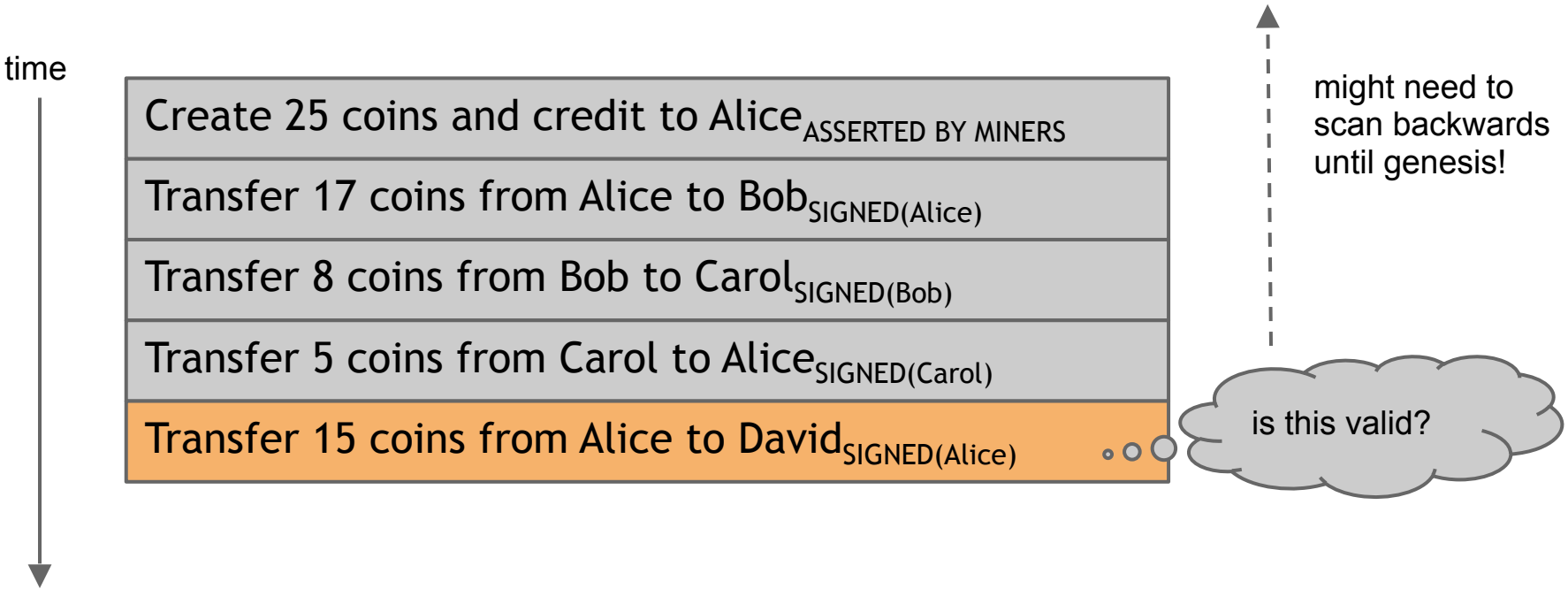
SIMPLIFICATION: only one transaction per block

An account-based ledger (*not* Bitcoin)



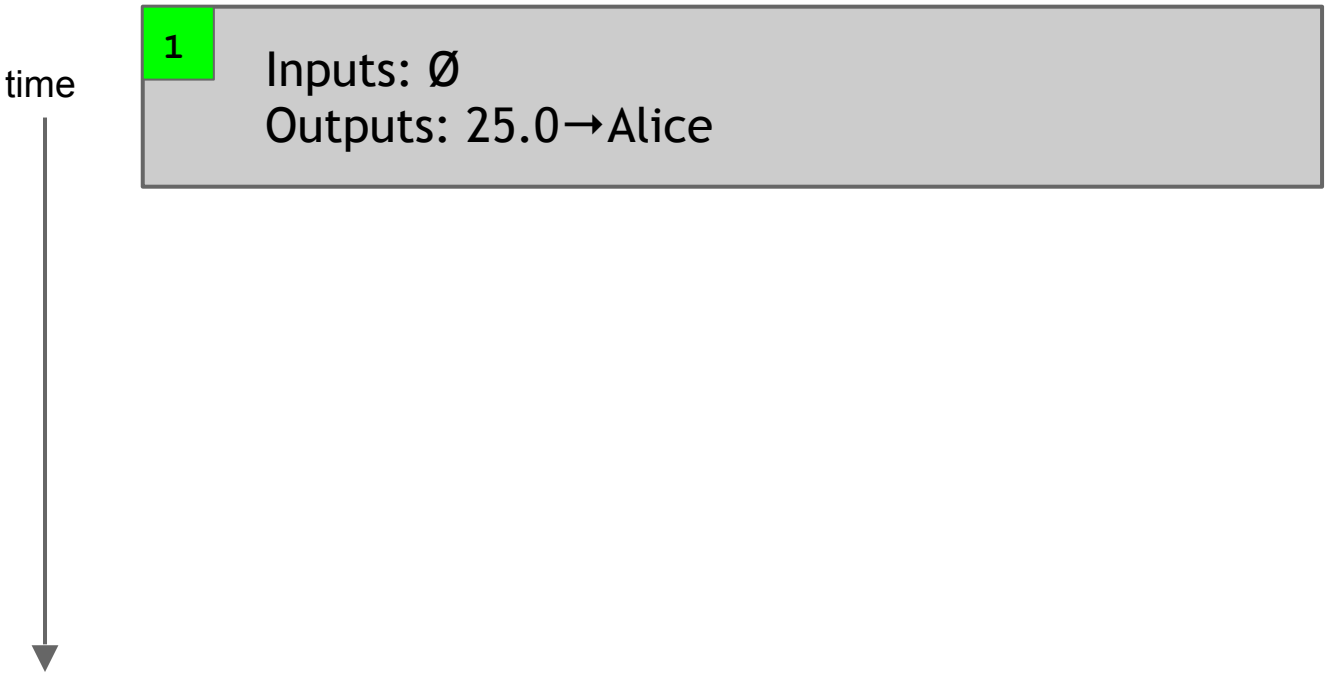
SIMPLIFICATION: only one transaction per block

An account-based ledger (*not* Bitcoin)



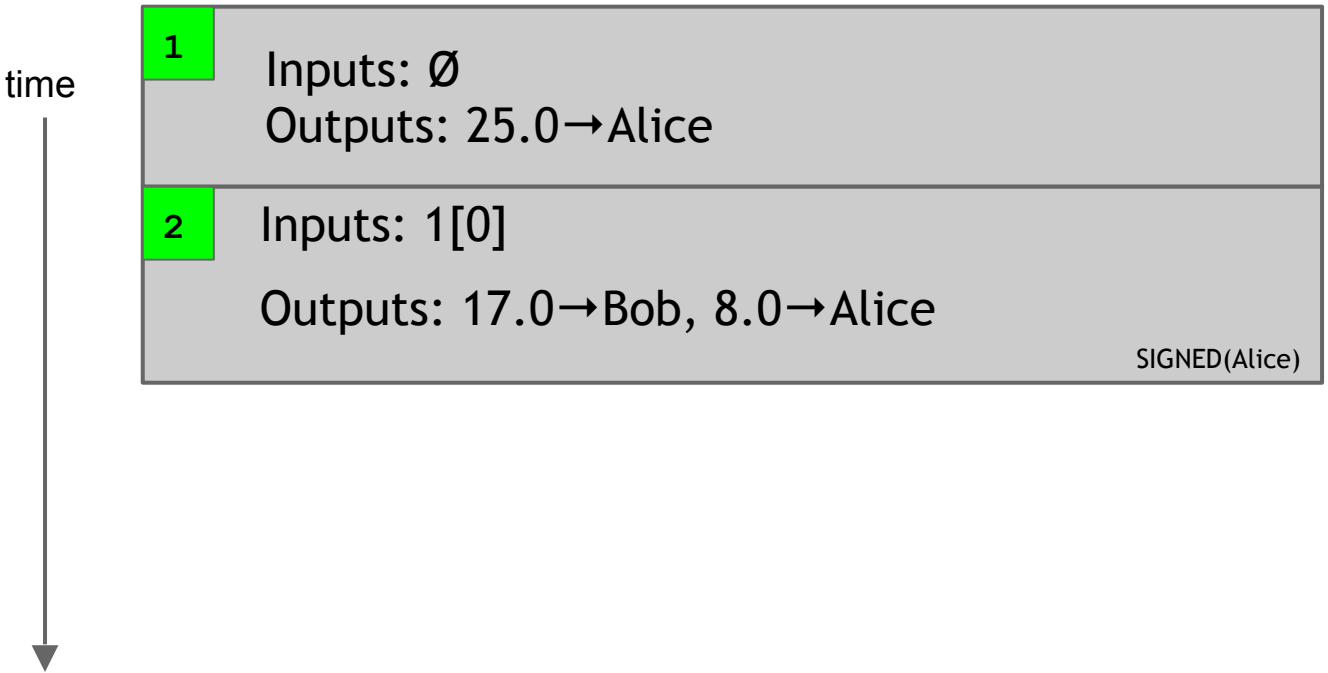
SIMPLIFICATION: only one transaction per block

A transaction-based ledger (Bitcoin)



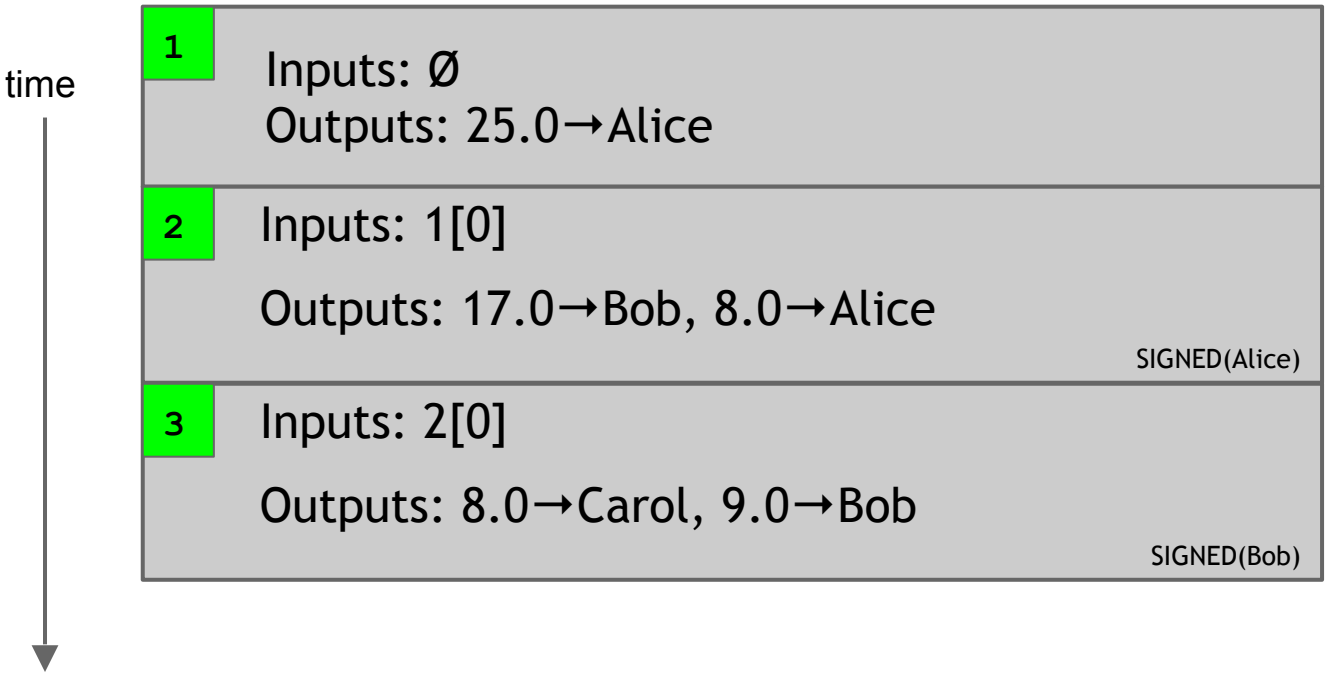
SIMPLIFICATION: only one transaction per block

A transaction-based ledger (Bitcoin)



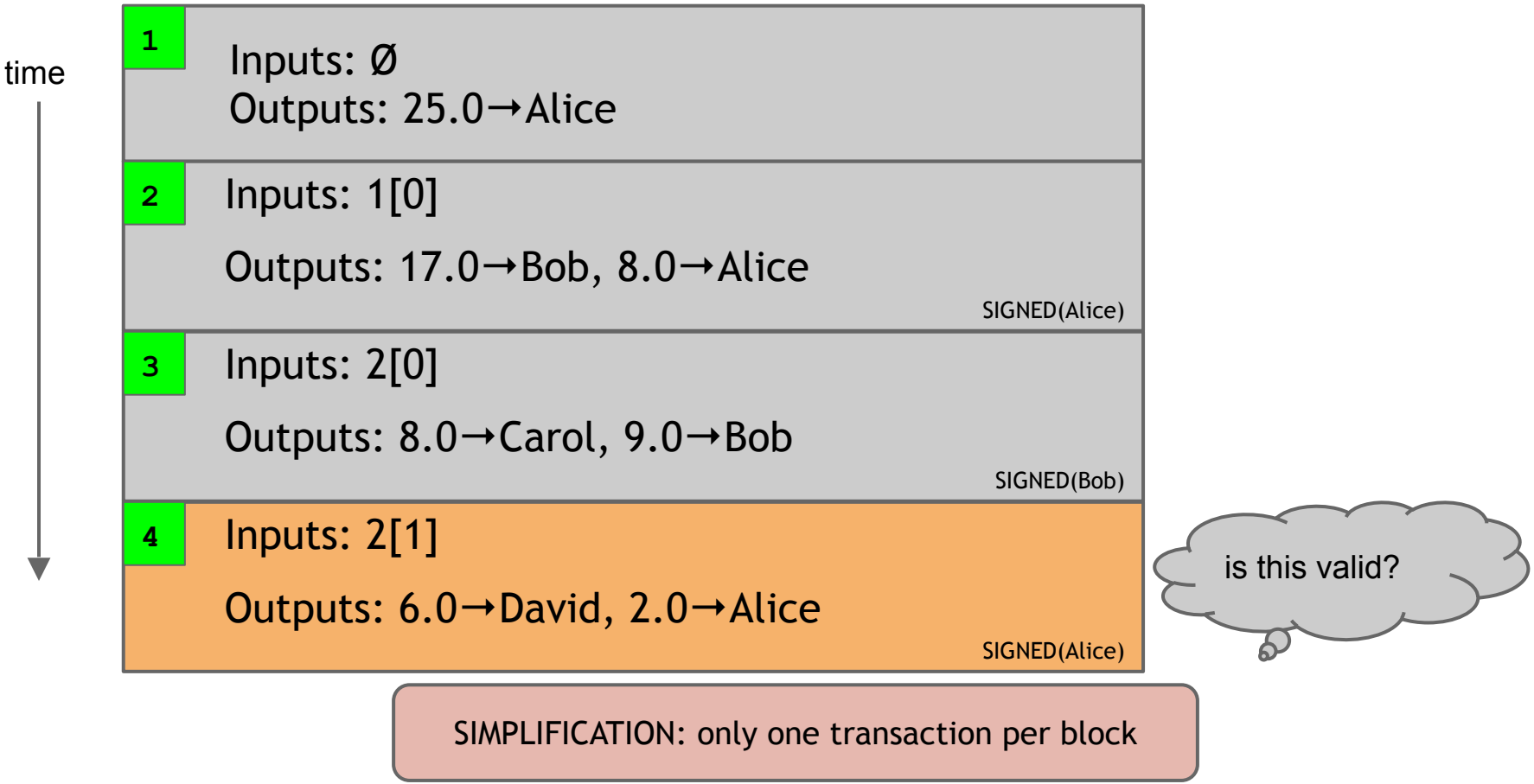
SIMPLIFICATION: only one transaction per block

A transaction-based ledger (Bitcoin)

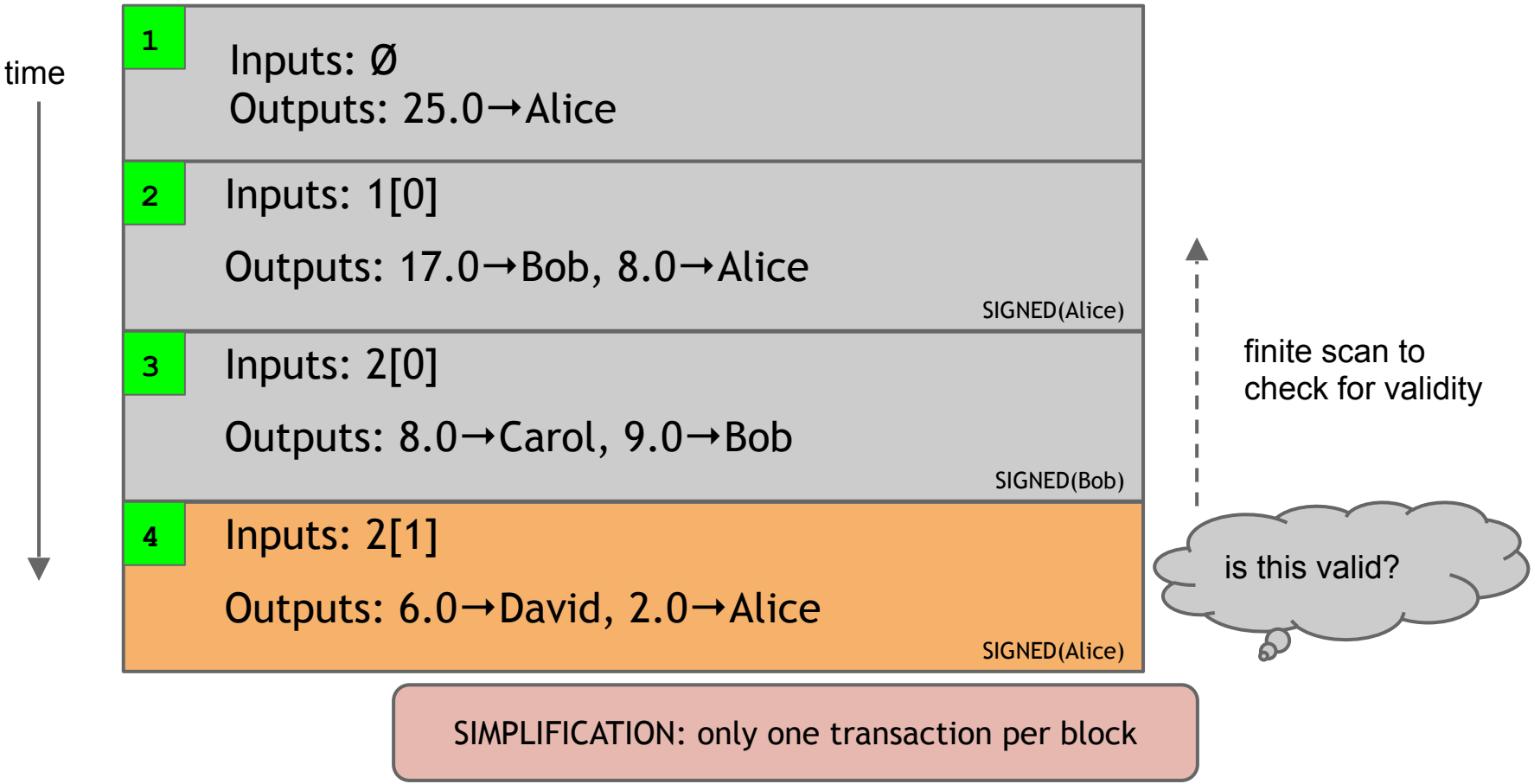


SIMPLIFICATION: only one transaction per block

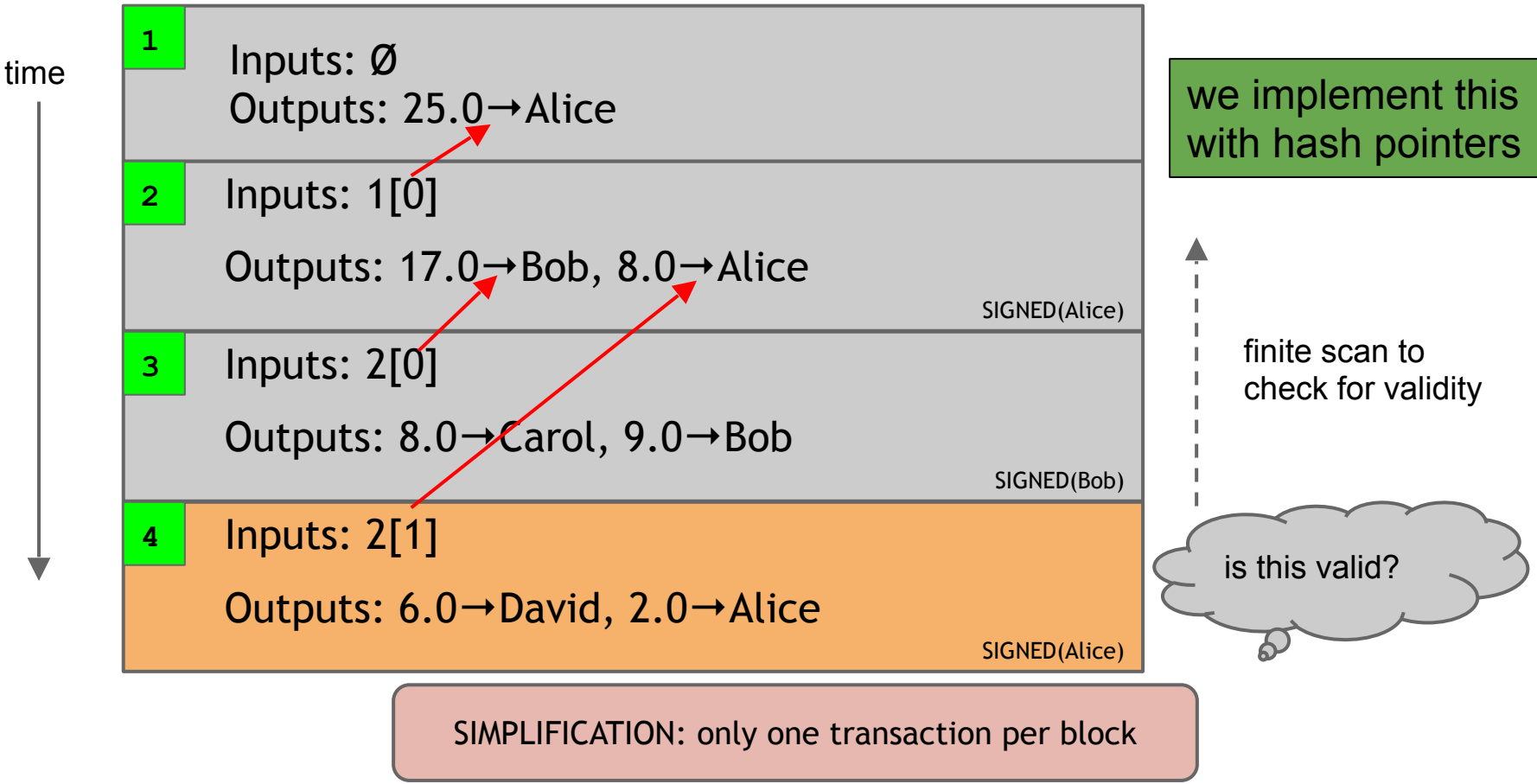
A transaction-based ledger (Bitcoin)



A transaction-based ledger (Bitcoin)



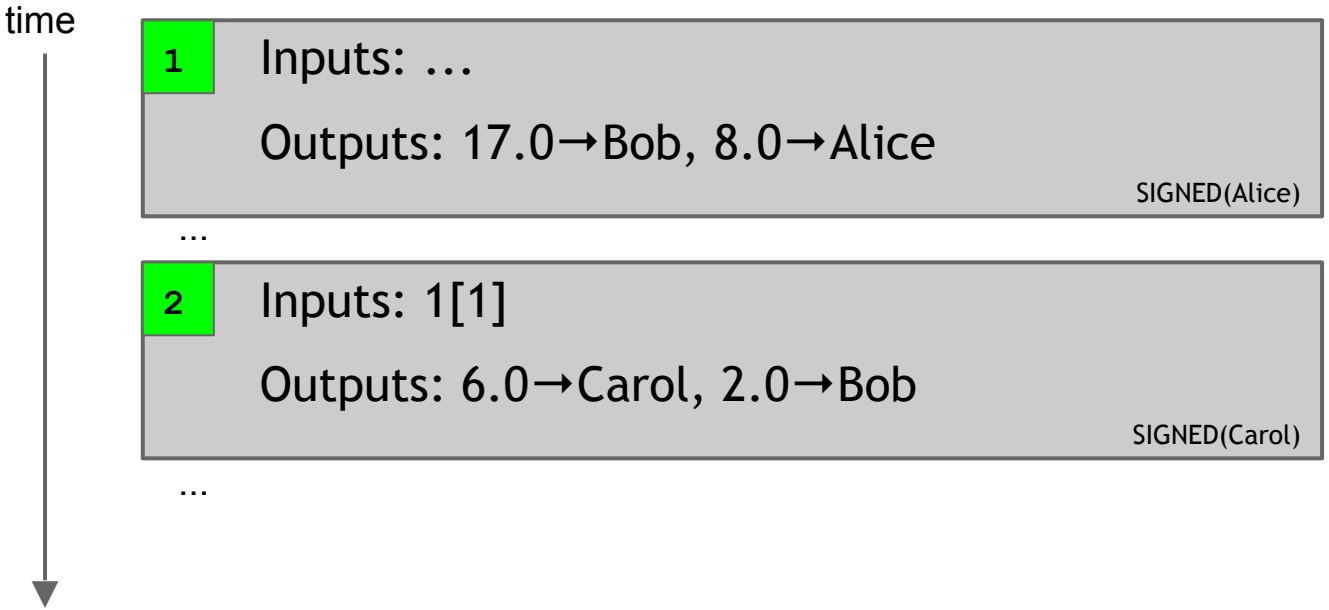
A transaction-based ledger (Bitcoin)



Referencing Transactions

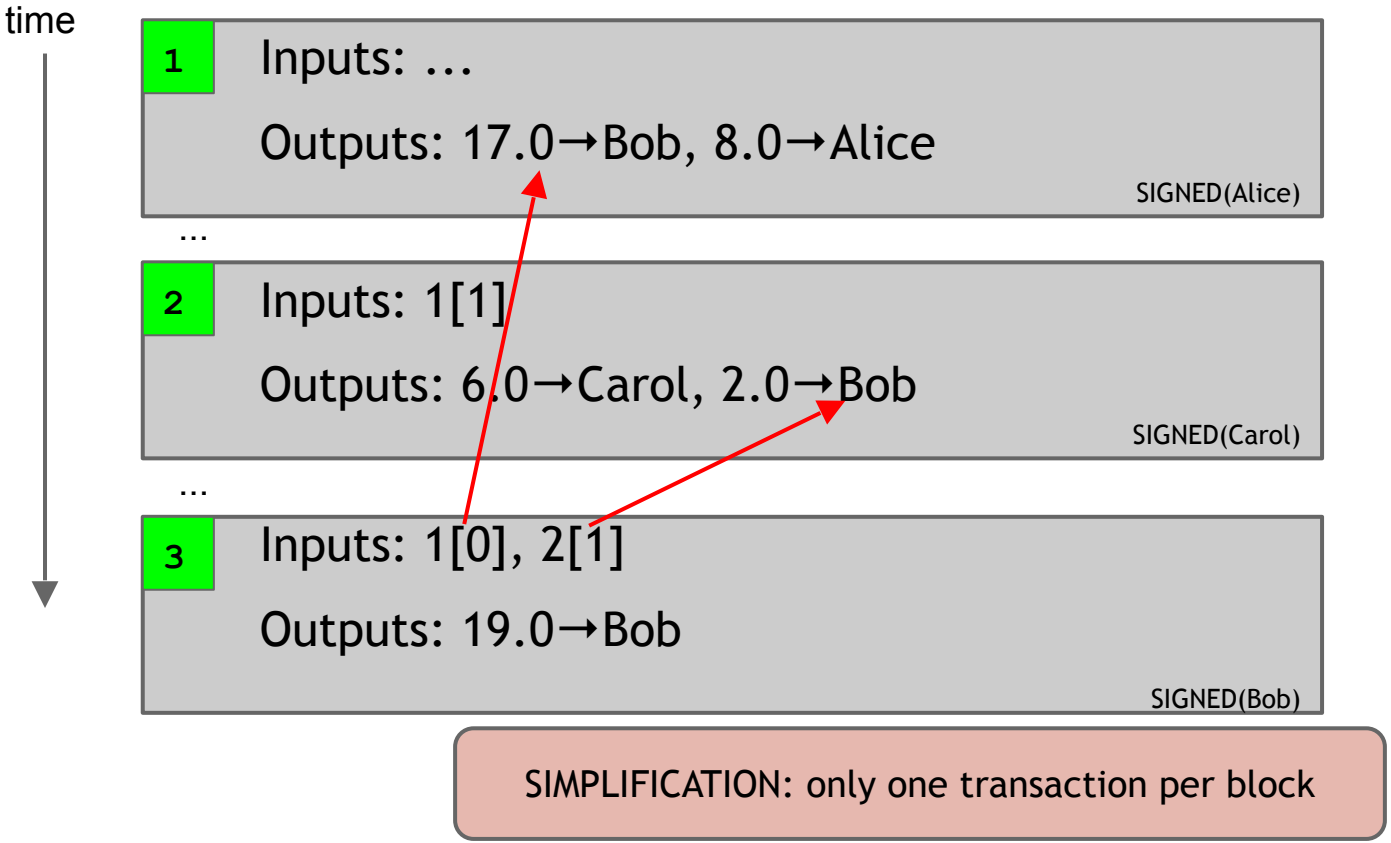
- Hash pointers for transactions
- Within a transaction, refer to a particular output via serial numbers

Merging value

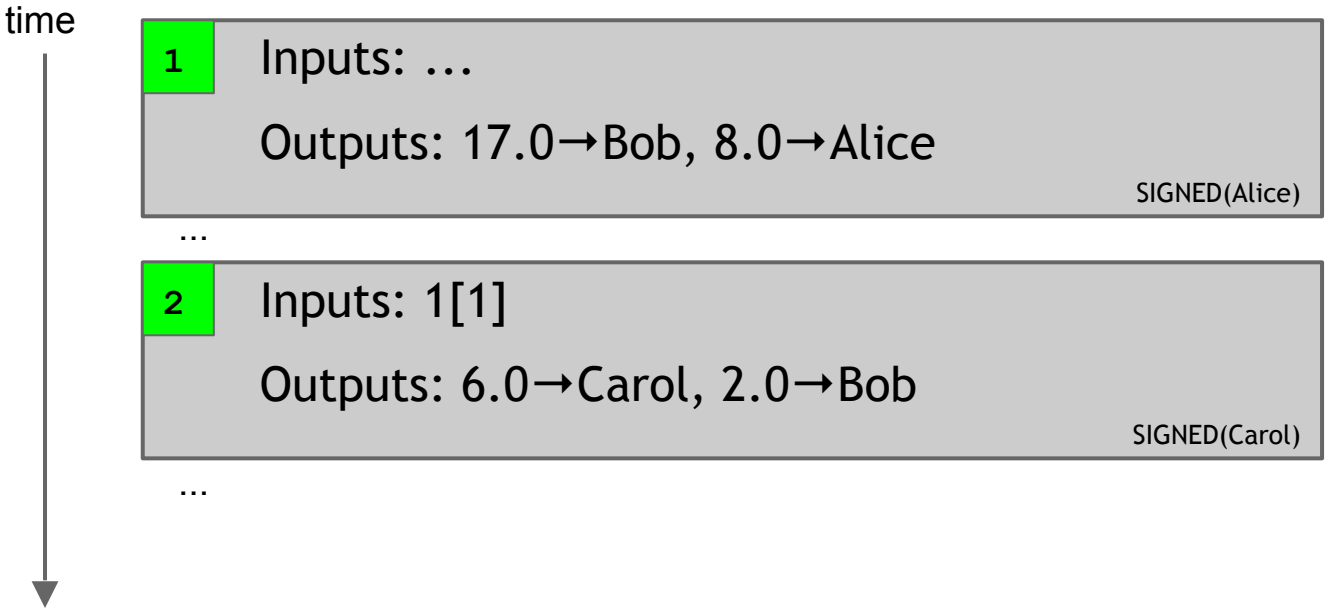


SIMPLIFICATION: only one transaction per block

Merging value

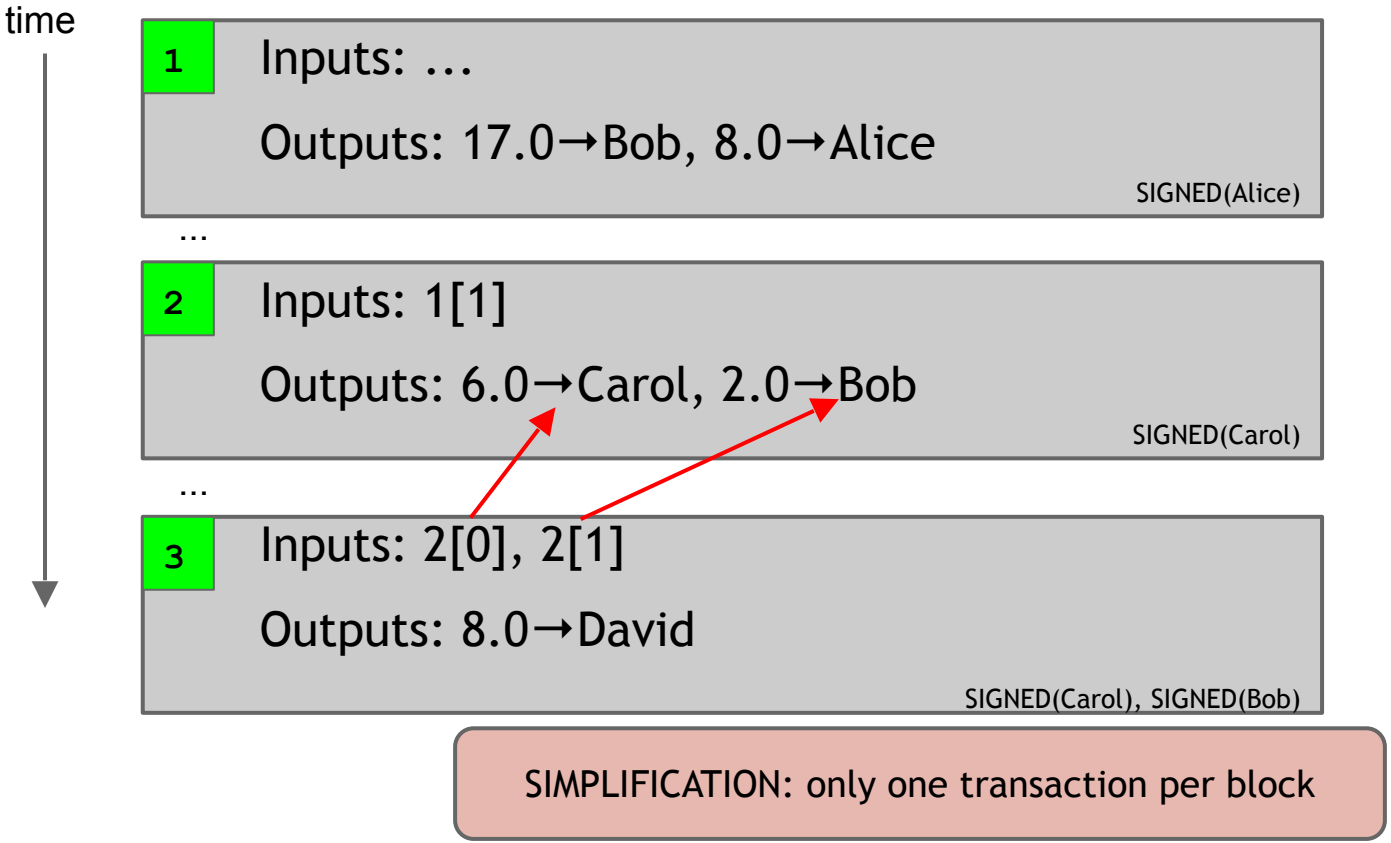


Joint payments

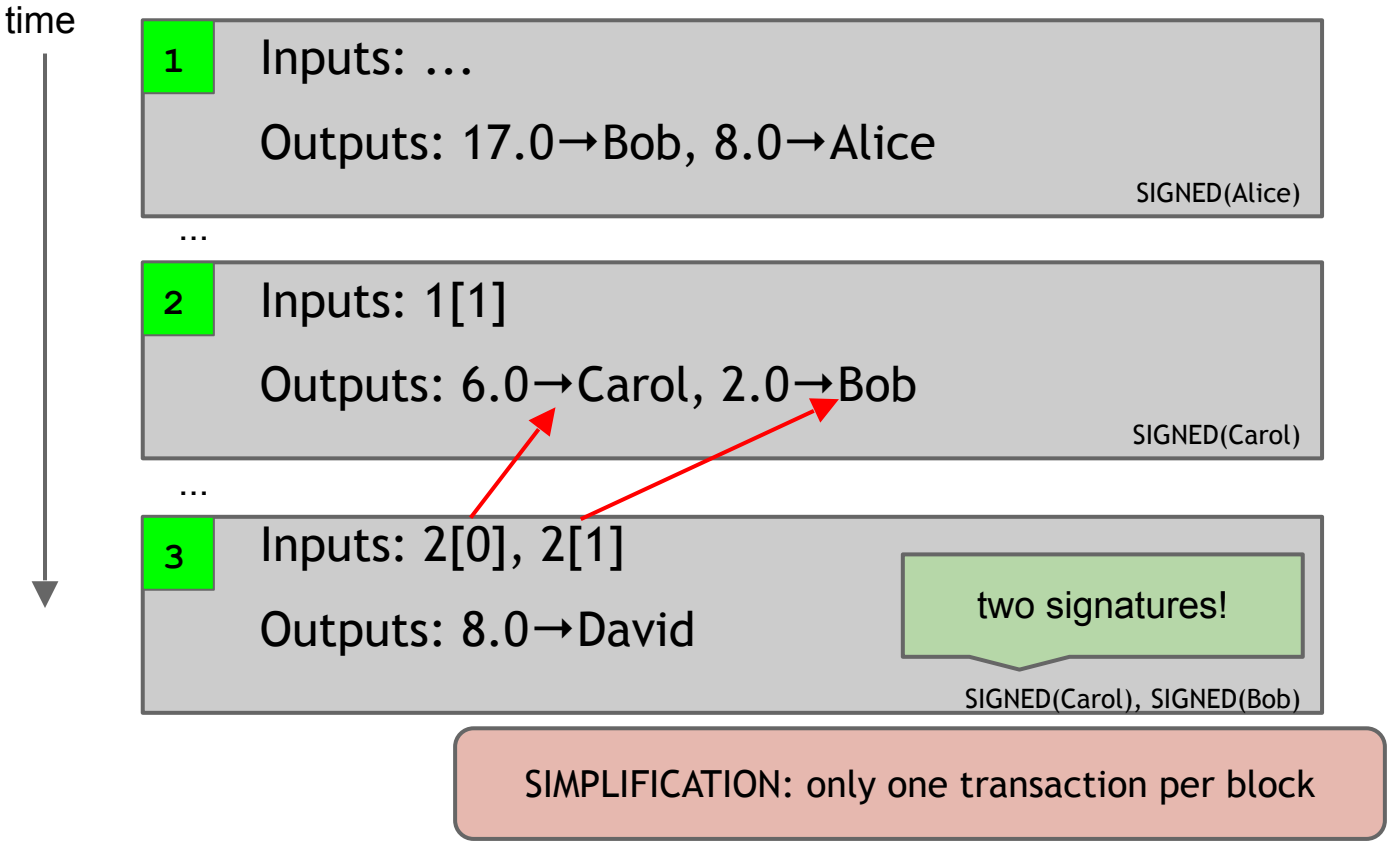


SIMPLIFICATION: only one transaction per block

Joint payments



Joint payments



The real deal: a classical Bitcoin transaction


```
{
    "hash":"5a42590fbe0a90ee8e8747244d6c84f0db1a3a24e8f1b95b10c9e050990b8b6b",
    "ver":1,
    "vin_sz":2,
    "vout_sz":1,
    "lock_time":0,
    "size":404,
    "in":[
      {
        "prev_out":{
          "hash":"3be4ac9728a0823cf5e2deb2e86fc0bd2aa503a91d307b42ba7617d79280260",
          "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: a classical 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":"3be4ac9728a0823cf5e2deb2e86fc0bd2aa503a91d307b42ba7617d79280260",
          "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


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

The real deal: transaction metadata

housekeeping

housekeeping

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

The real deal: transaction metadata

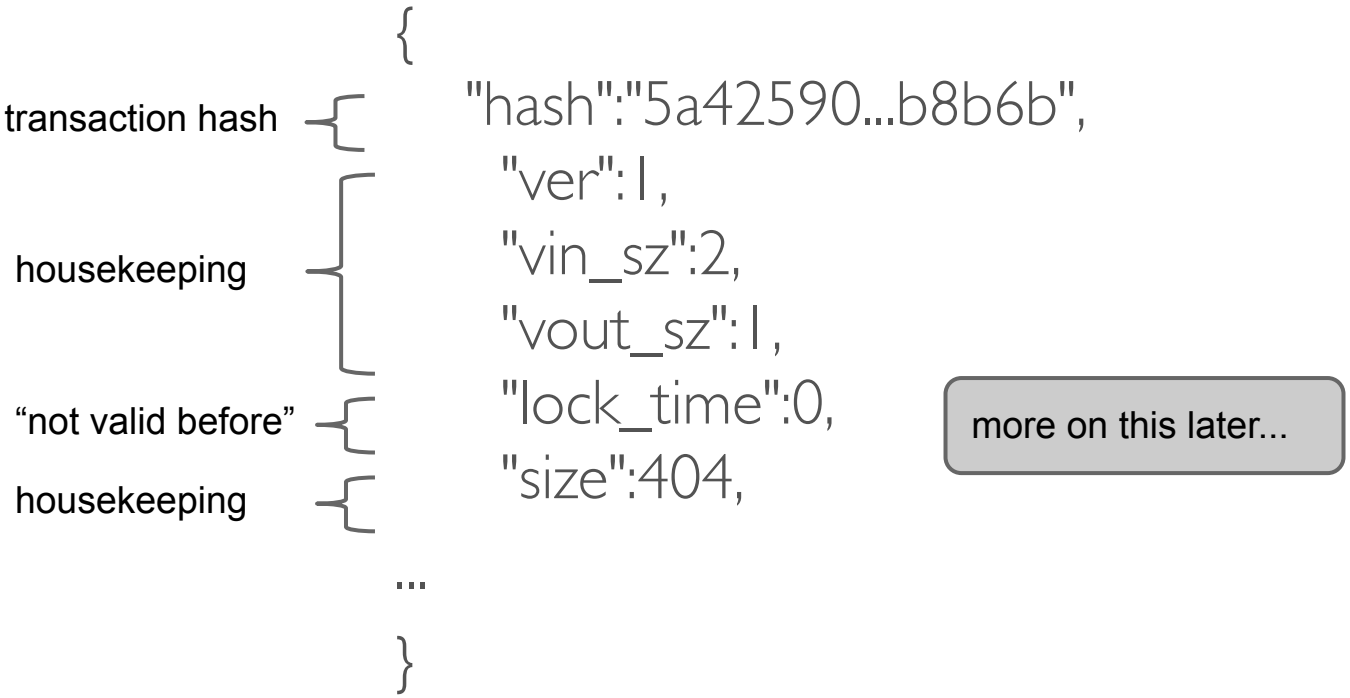
transaction hash

housekeeping

housekeeping

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

The real deal: transaction metadata



The real deal: transaction inputs


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

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


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

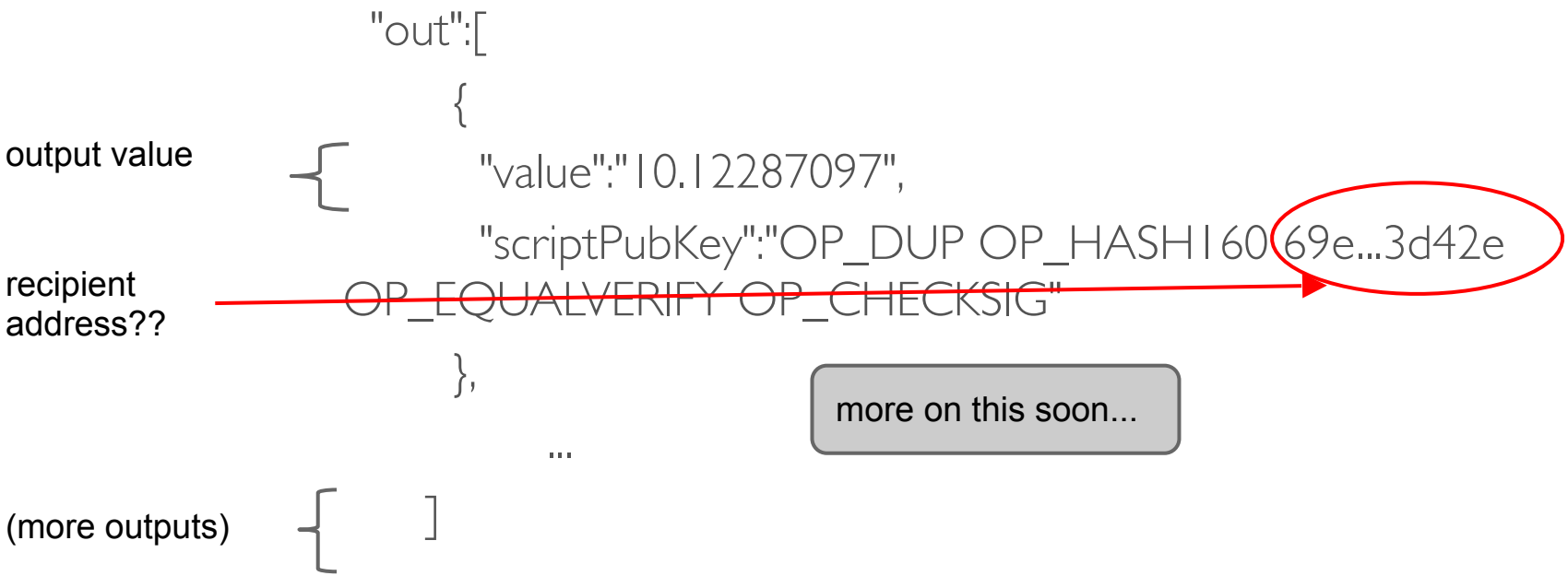
The real deal: transaction outputs

output value

(more outputs)

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

The real deal: transaction outputs



Bitcoin scripts

Output “addresses” are really *scripts*

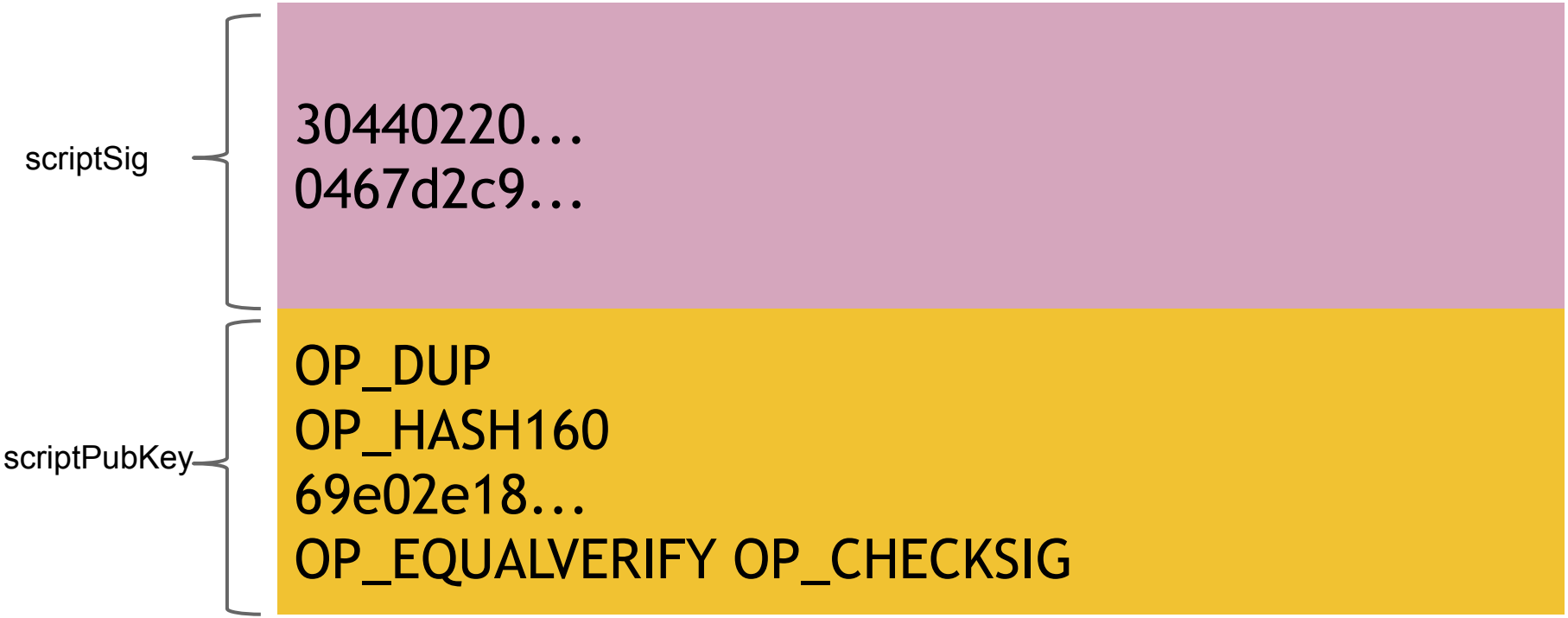
```
OP_DUP
OP_HASH160
69e02e18...
OP_EQUALVERIFY OP_CHECKSIG
```

Input “addresses” are *also* scripts

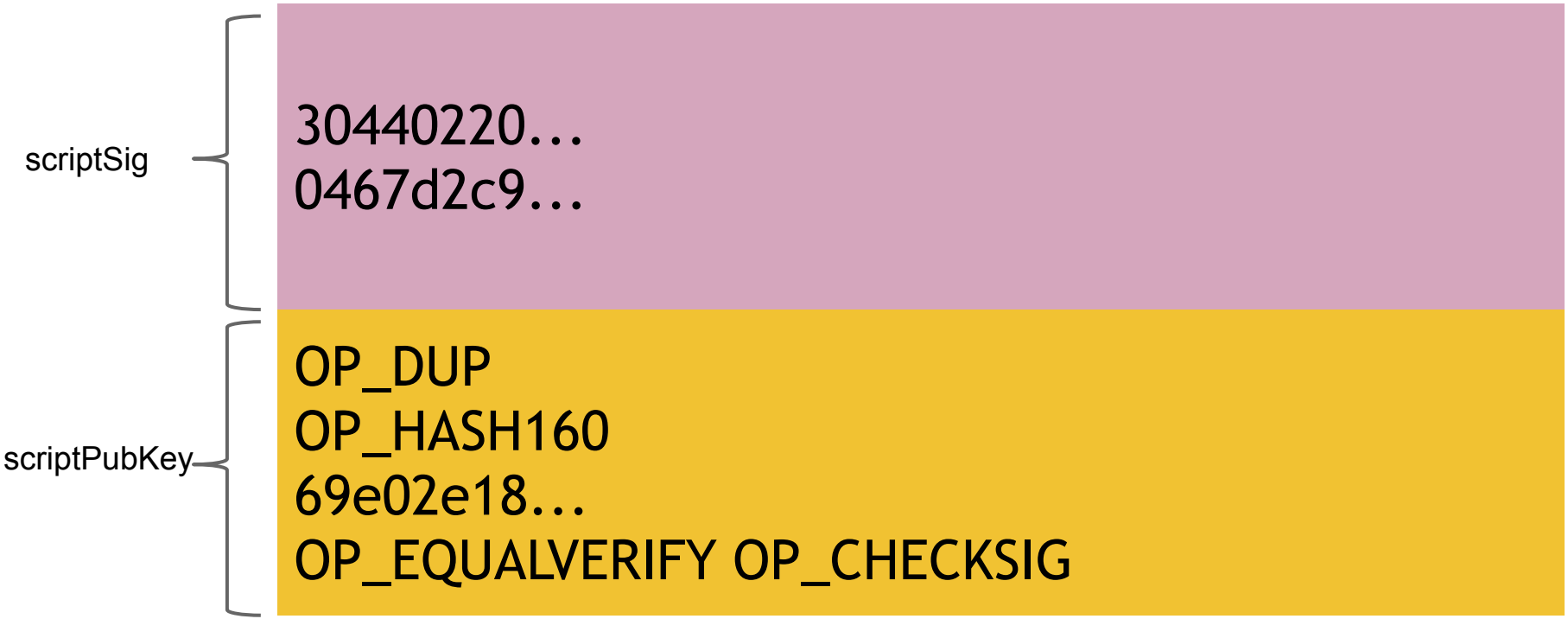
30440220…
0467d2c9…

OP_DUP
OP_HASH160
69e02e18…
OP_EQUALVERIFY OP_CHECKSIG

Input "addresses" are *also* scripts



Input "addresses" are *also* scripts



Bitcoin scripting language ("Script")

Design goals

- Built for Bitcoin
- Simple, compact
- Support for cryptography
- Stack-based
- Limits on time/memory
- No looping

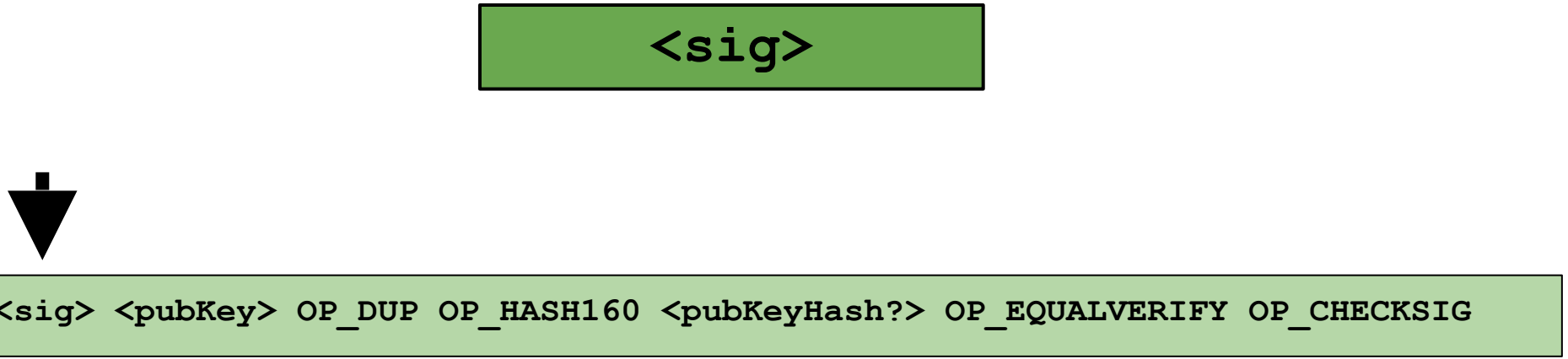
Bitcoin script execution example

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

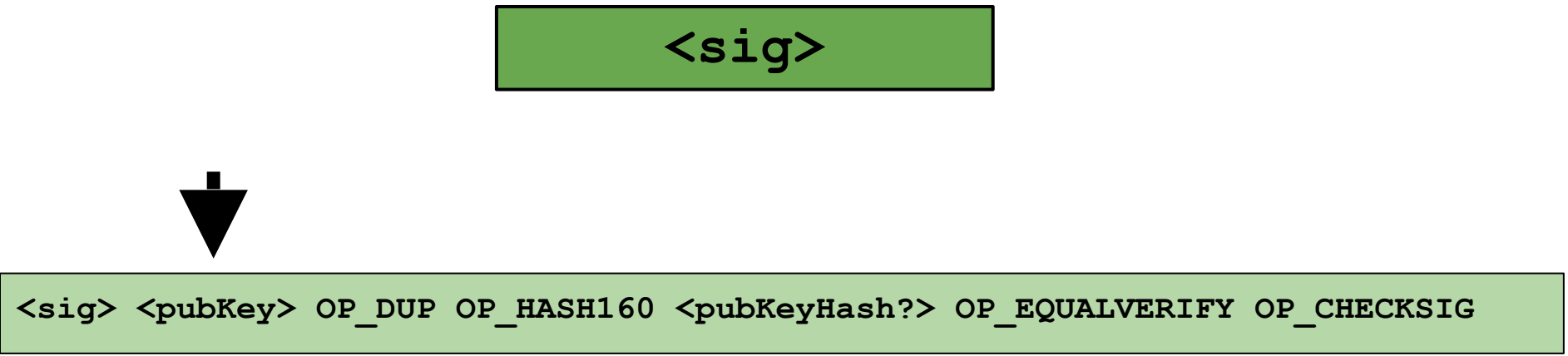
Bitcoin script execution example

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

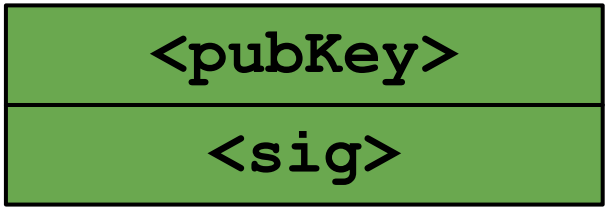
Bitcoin script execution example



Bitcoin script execution example

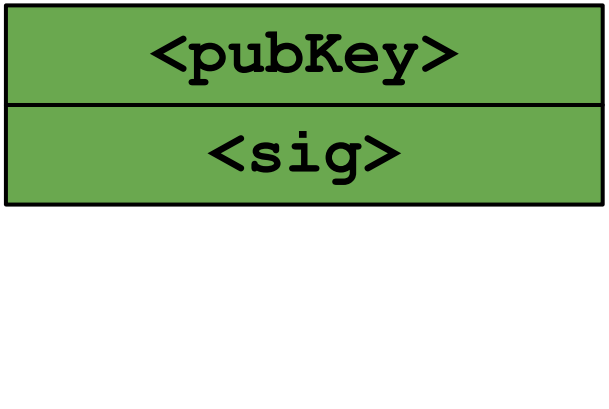


Bitcoin script execution example



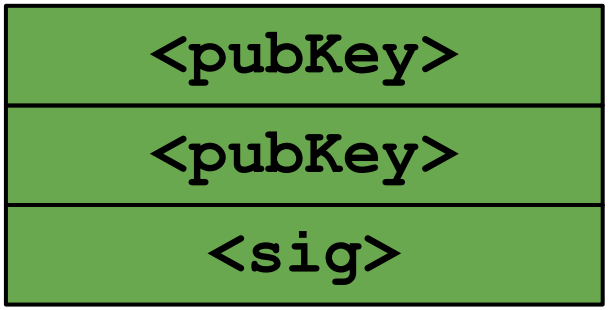
```
<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash?> OP_EQUALVERIFY OP_CHECKSIG
```

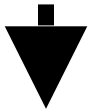
Bitcoin script execution example




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

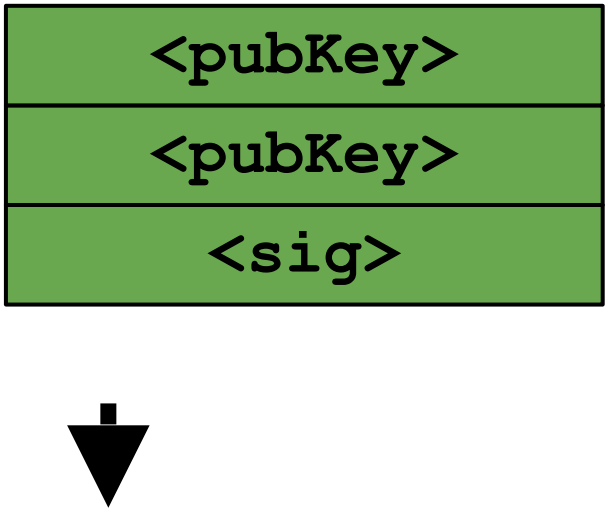
Bitcoin script execution example





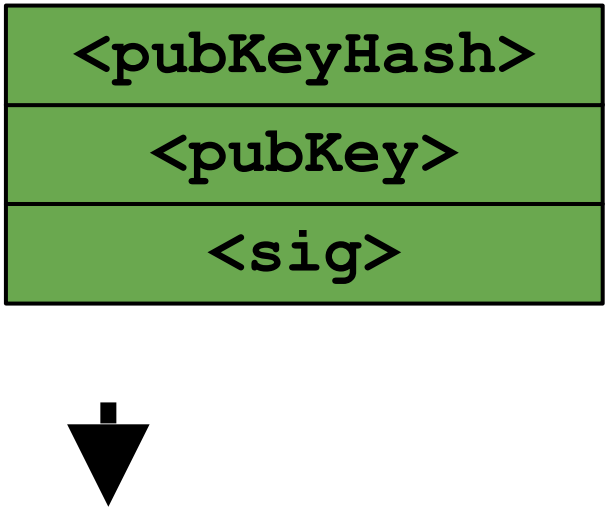
```
<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash?> OP_EQUALVERIFY OP_CHECKSIG
```

Bitcoin script execution example



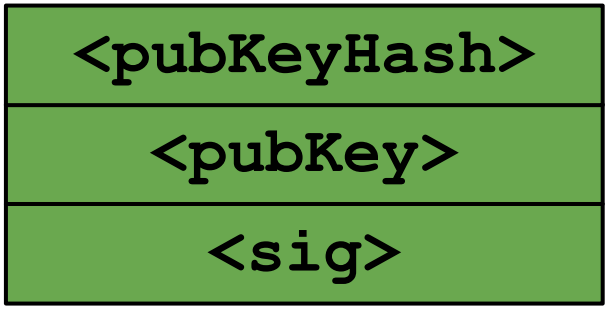

```
<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash?> OP_EQUALVERIFY OP_CHECKSIG
```

Bitcoin script execution example



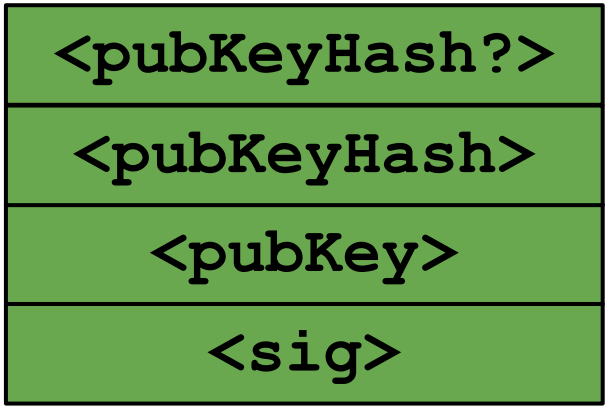

```
<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash?> OP_EQUALVERIFY OP_CHECKSIG
```

Bitcoin script execution example



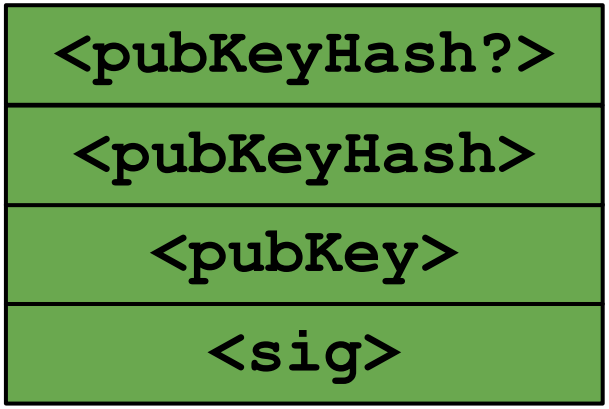

```
<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash?> OP_EQUALVERIFY OP_CHECKSIG
```

Bitcoin script execution example



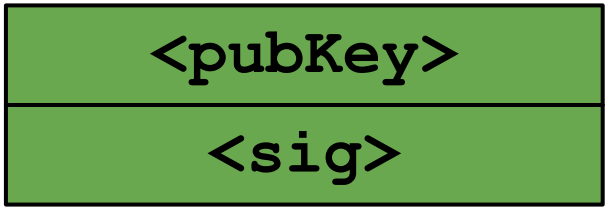

```
<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash?> OP_EQUALVERIFY OP_CHECKSIG
```

Bitcoin script execution example



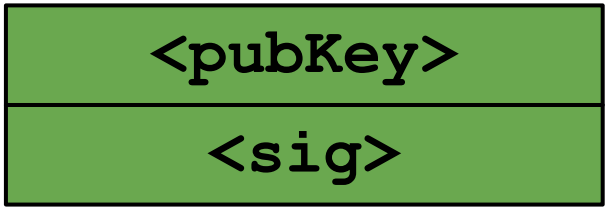
```
<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash?> OP_EQUALVERIFY OP_CHECKSIG
```

Bitcoin script execution example



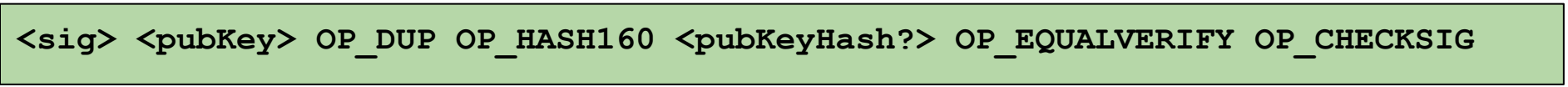
```
<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash?> OP_EQUALVERIFY OP_CHECKSIG
```

Bitcoin script execution example

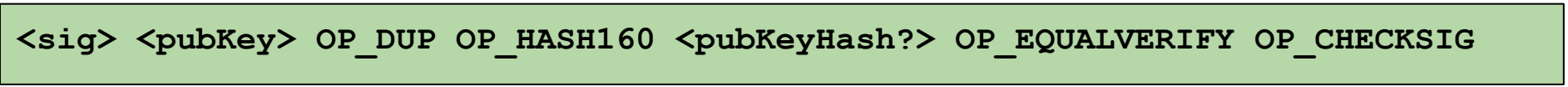


```
<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash?> OP_EQUALVERIFY OP_CHECKSIG
```

Bitcoin script execution example



Bitcoin script execution example



Bitcoin script execution example

true

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

Bitcoin script instructions

256 opcodes total (15 disabled, 75 reserved)

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

OP_CHECKMULTISIG

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

Bitcoin scripts in practice ("original")

- 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

* numbers from NBFMG and slightly out of date

Bitcoin scripts in practice ("original")

- 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 on this soon

* numbers from NBFMG and slightly out of date

Proof-of-burn

OP_RETURN
<arbitrary data>

Proof-of-burn

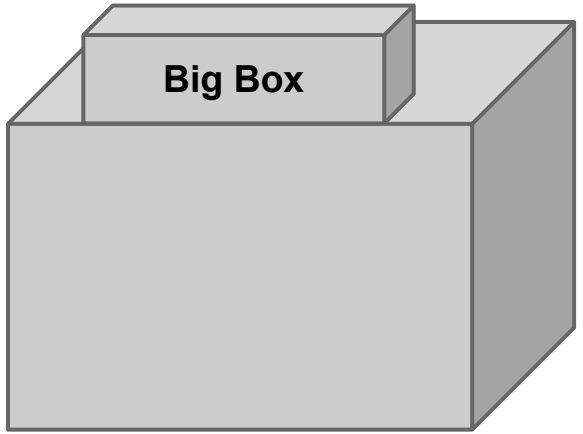
nothing’s going to redeem that ☹

OP_RETURN
<arbitrary data>

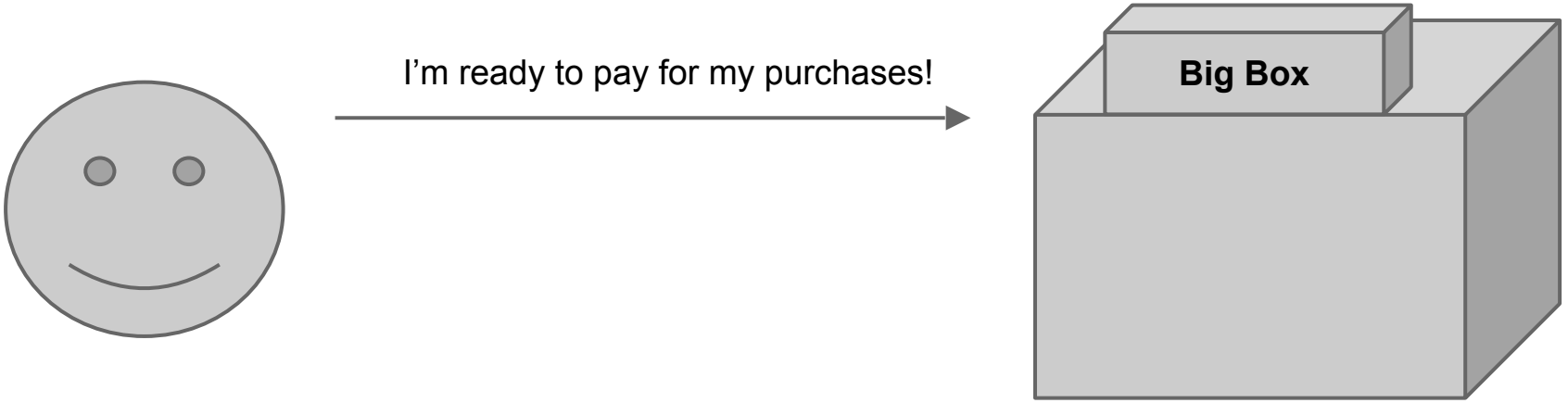
Proof-of-burn: Applications

- Can be used to publish arbitrary data on the blockchain (e.g., timestamping a document)
- Bootstrap Altcoins by requiring people to destroy bitcoins in order to get new altcoins

Should senders specify scripts?



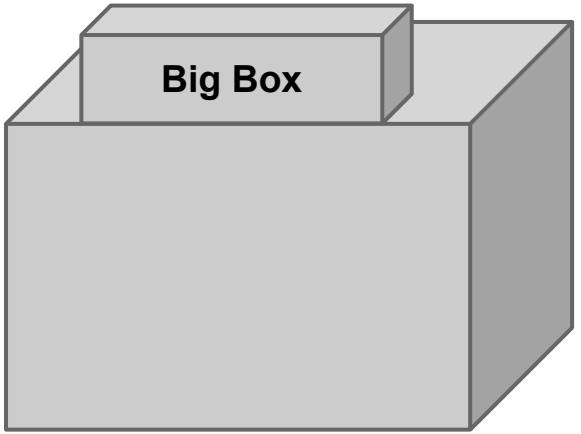
Should senders specify scripts?



Should senders specify scripts?

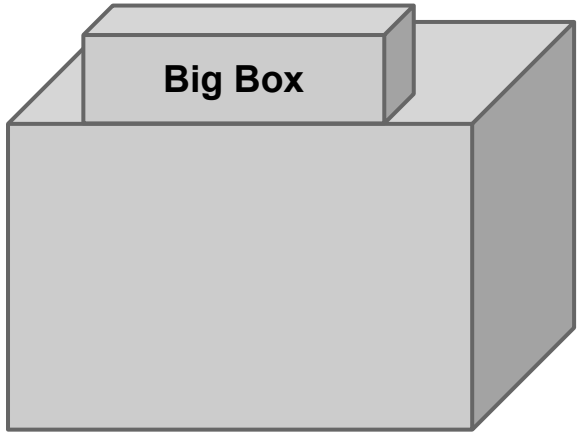
I’m ready to pay for my purchases!

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!



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

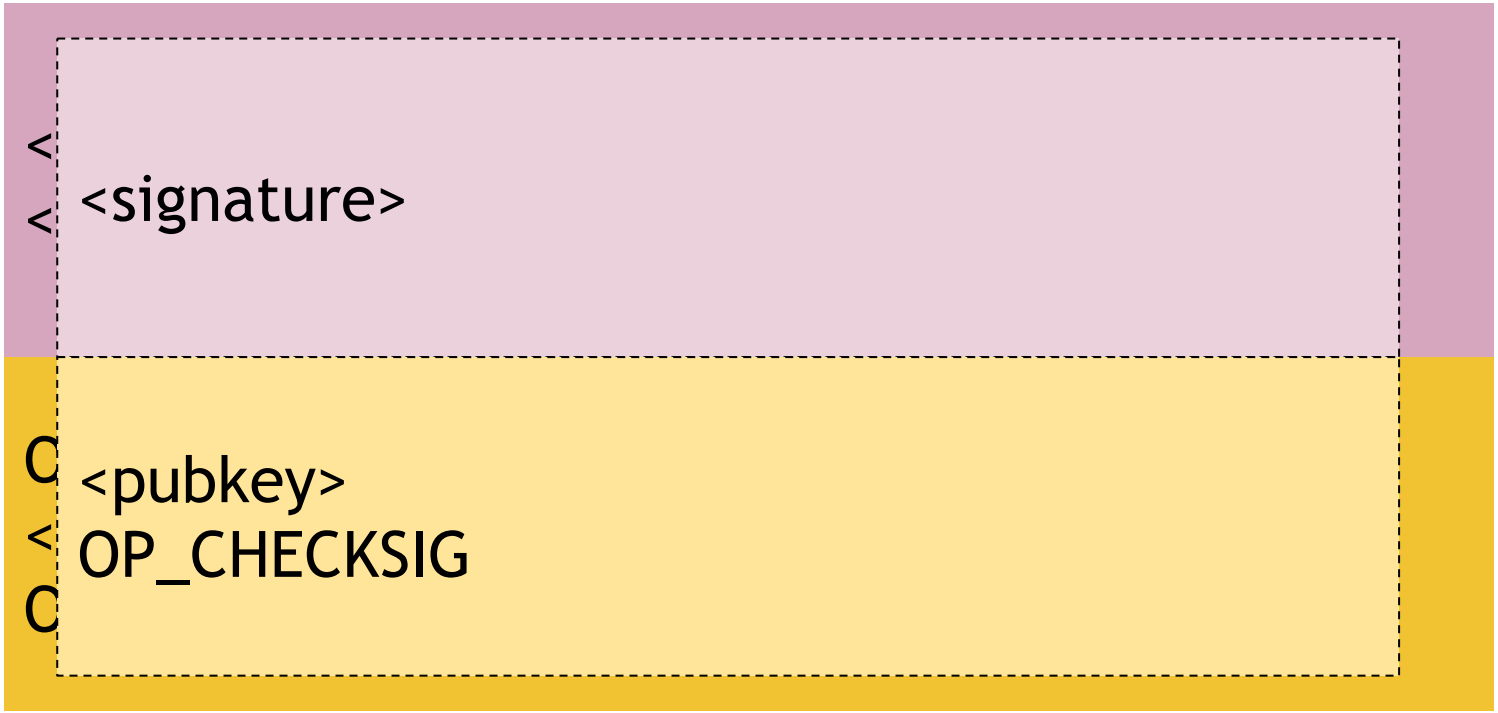
```
OP_HASH160
<hash of redemption script>
OP_EQUAL
```

Idea: use the hash of redemption script

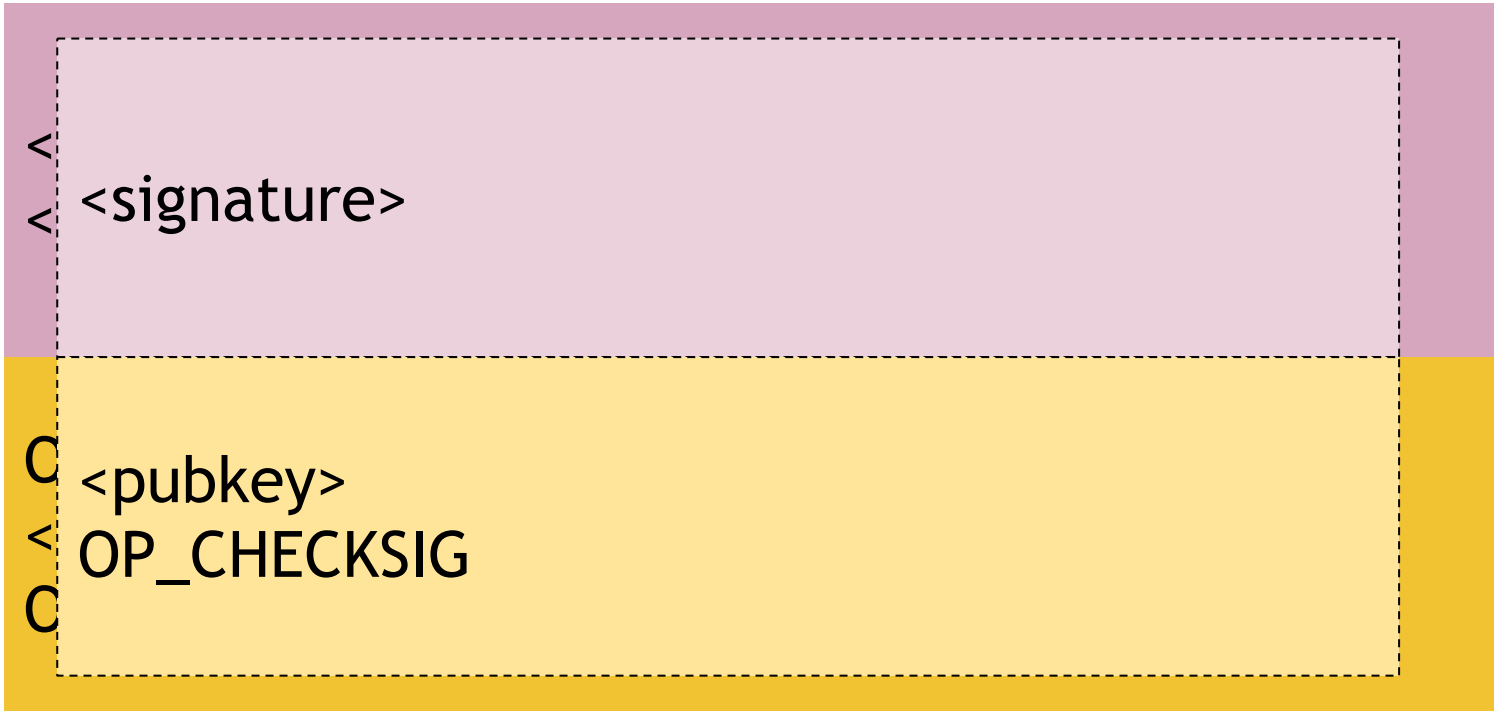
<signature>
<<pubkey> OP_CHECKSIG>

OP_HASH160
<hash of redemption script>
OP_EQUAL

Idea: use the hash of redemption script

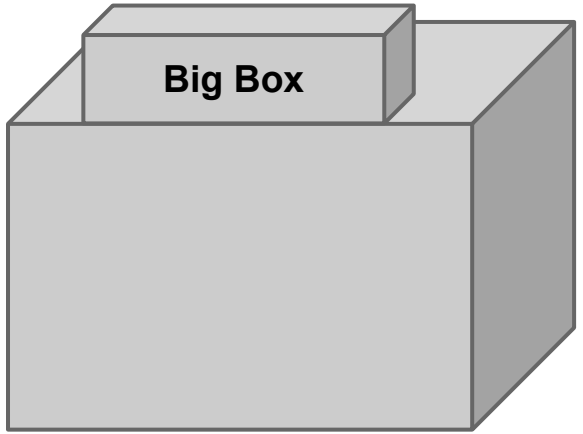


Idea: use the hash of redemption script



Pay to script hash

I’m ready to pay for my purchases!



Pay to script hash