Blockchains & Cryptocurrencies

Mechanics of Bitcoin

Instructor: Matthew Green
Johns Hopkins University - Fall 2024

Many slides based on NBFMG

Housekeeping

- Pace of material?
- Office hours
- A1 is due 9/18 11:59pm
- **Project groups and proposal is due 10/8 end of day**
    - Project ideas up on main Github Wiki page

News?

Today

- Finish talking about Bitcoin at a high level
- Talk about the low-level mechanics of real Bitcoin

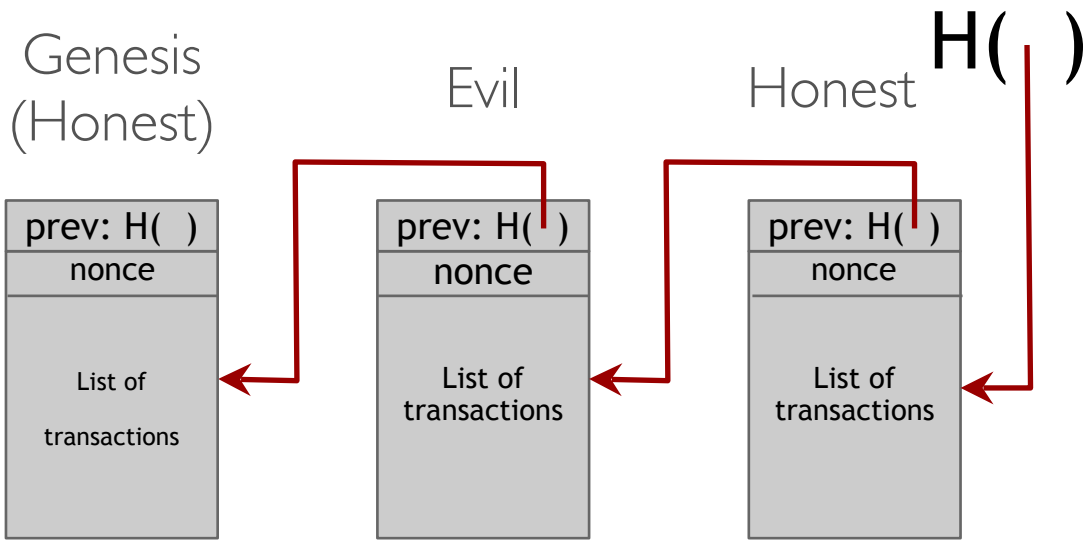
Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

1. Introduction

Review


Genesis
(Honest)
Evil
Honest
H( )
prev: H( )
nonce
List of
transactions
prev: H( )
nonce
List of
transactions
prev: H( )
nonce
List of
transactions

Selecting T (Bitcoin)

Every block contains a (packed) encoding of T (target) which corresponds to ***"difficulty" (d)***

d = ((2^{16} - 1) * 2^{8*26}) / T <- relationship between d, T

Goals:

- Bitcoin block time should average 10 minutes
- 2016 blocks * 10 minutes == 2 weeks
- Everyone in the network agrees on *T*
- *Hence must be a function of chain data*

This brings back a notion of time

What if there’s a collision?

Sometimes two separate nodes find a valid solution simultaneously

This can result in network partition

What if there’s a collision?

What if there’s a collision?

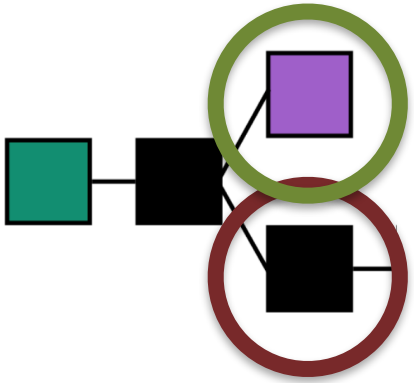
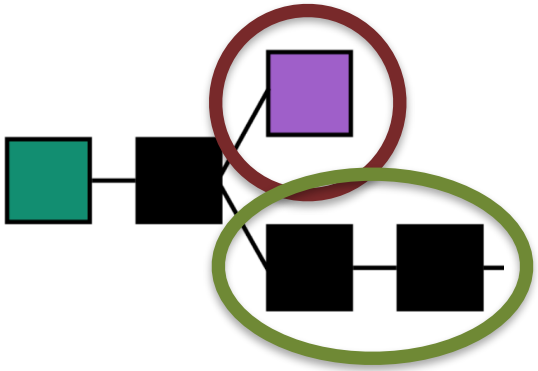


Image CC-BY-3 Theymos taken from the Bitcoin wiki

What if there's a collision?



“Longest chain rule”

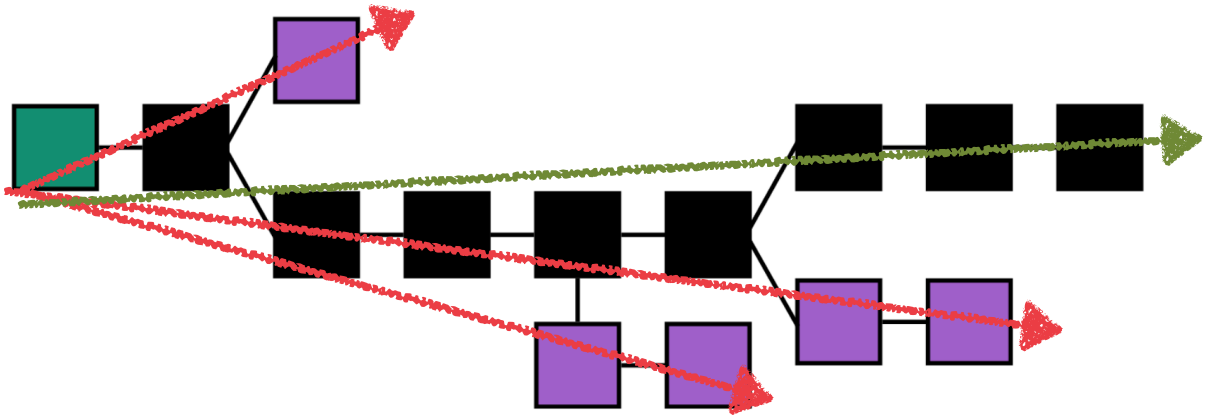


Image CC-BY-3 Theymos taken from the Bitcoin wiki

“Longest chain rule”

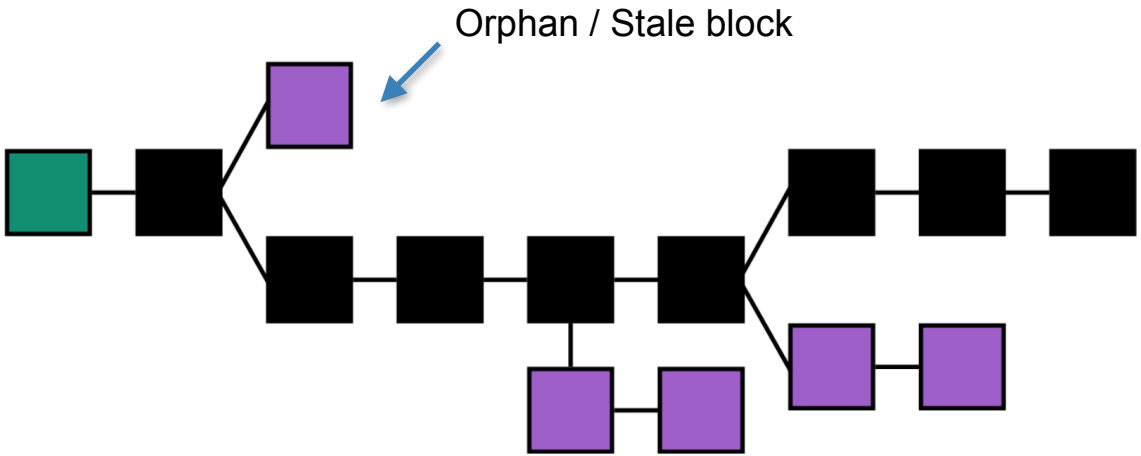


Image CC-BY-3 Theymos taken from the Bitcoin wiki

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 one chain to advance

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

What’s the bad?

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*

Q: What if difficulty changes?

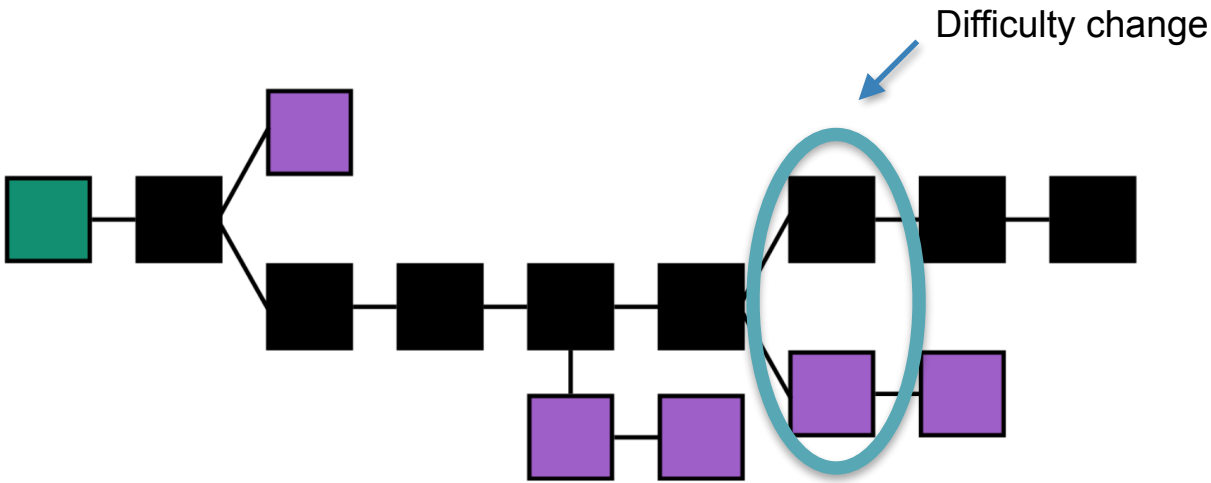
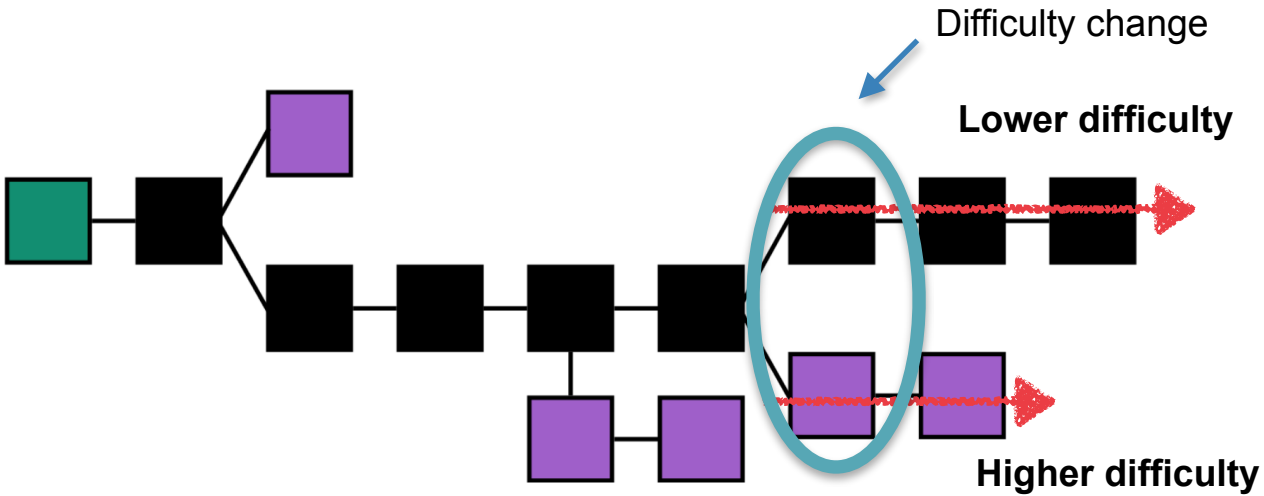


Image CC-BY-3 Theymos taken from the Bitcoin wiki

Q: What if difficulty changes?



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

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?

We will see some attacks that leverage these simplified calculations later….

How do we incentivize mining?

How do we incentivize mining?

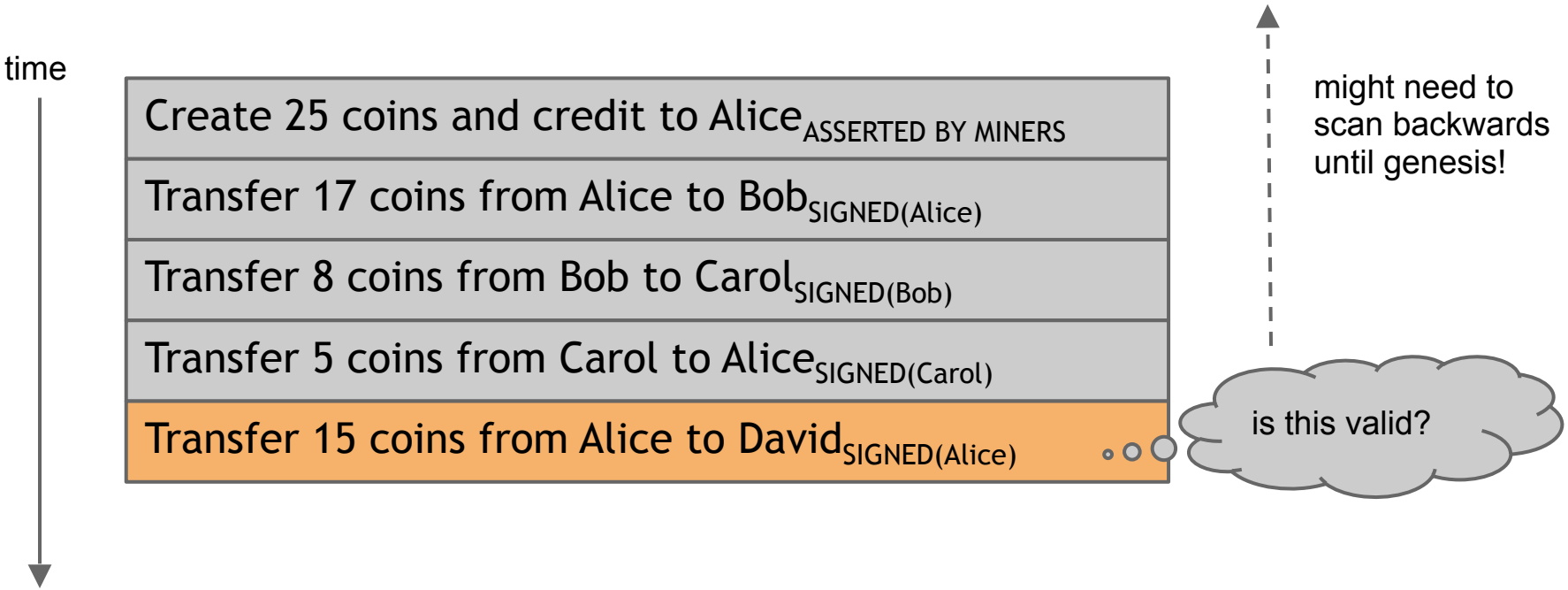
Two answers to this question in Bitcoin:

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/… BTC made from scratch (in a special Coinbase transaction) and given to the miner

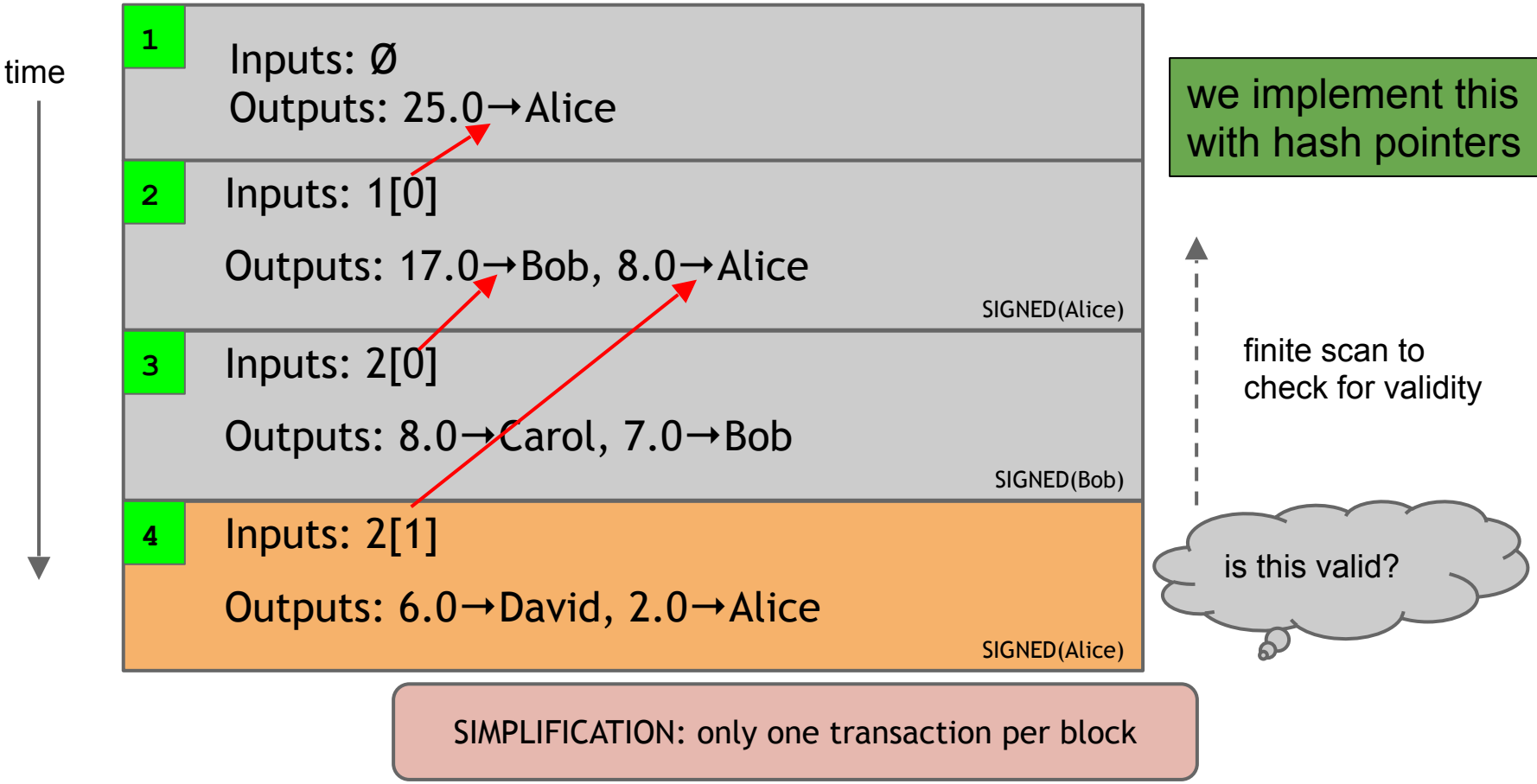
Bitcoin transactions

An account-based ledger (*not* Bitcoin)



SIMPLIFICATION: only one transaction per block

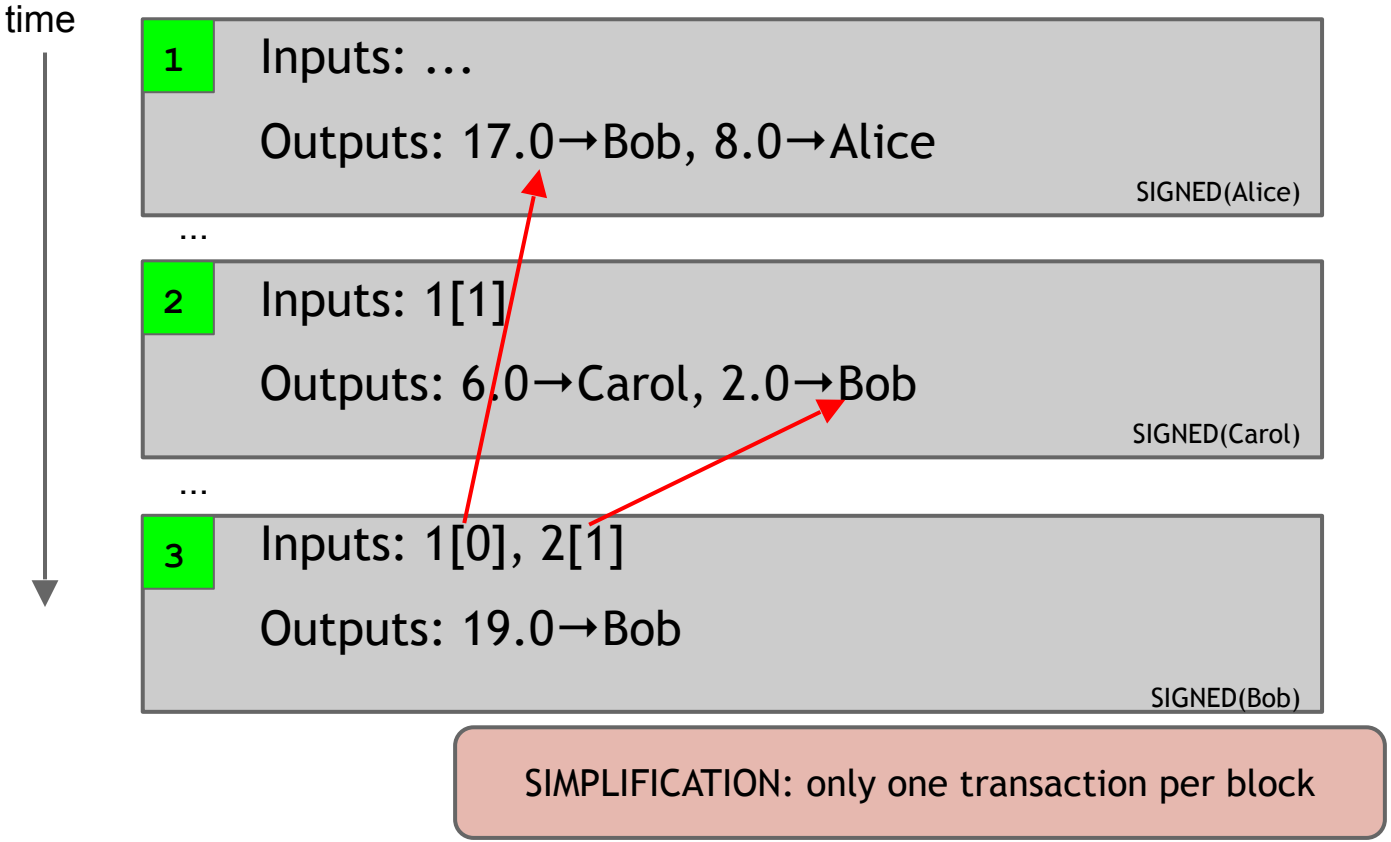
A transaction-based ledger (Bitcoin)



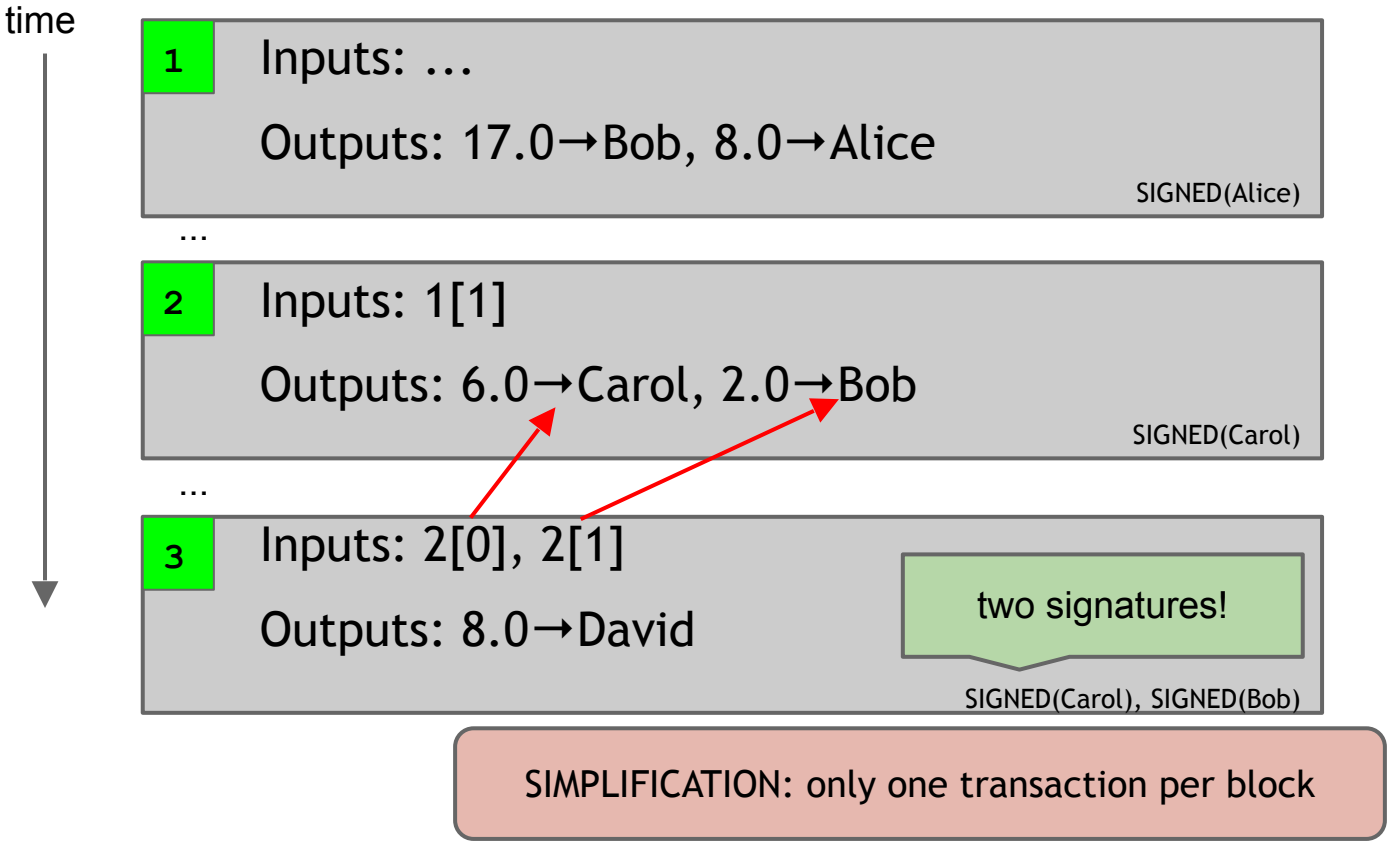
Referencing Transactions

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

Merging value



Joint payments



The real deal: a classical Bitcoin transaction

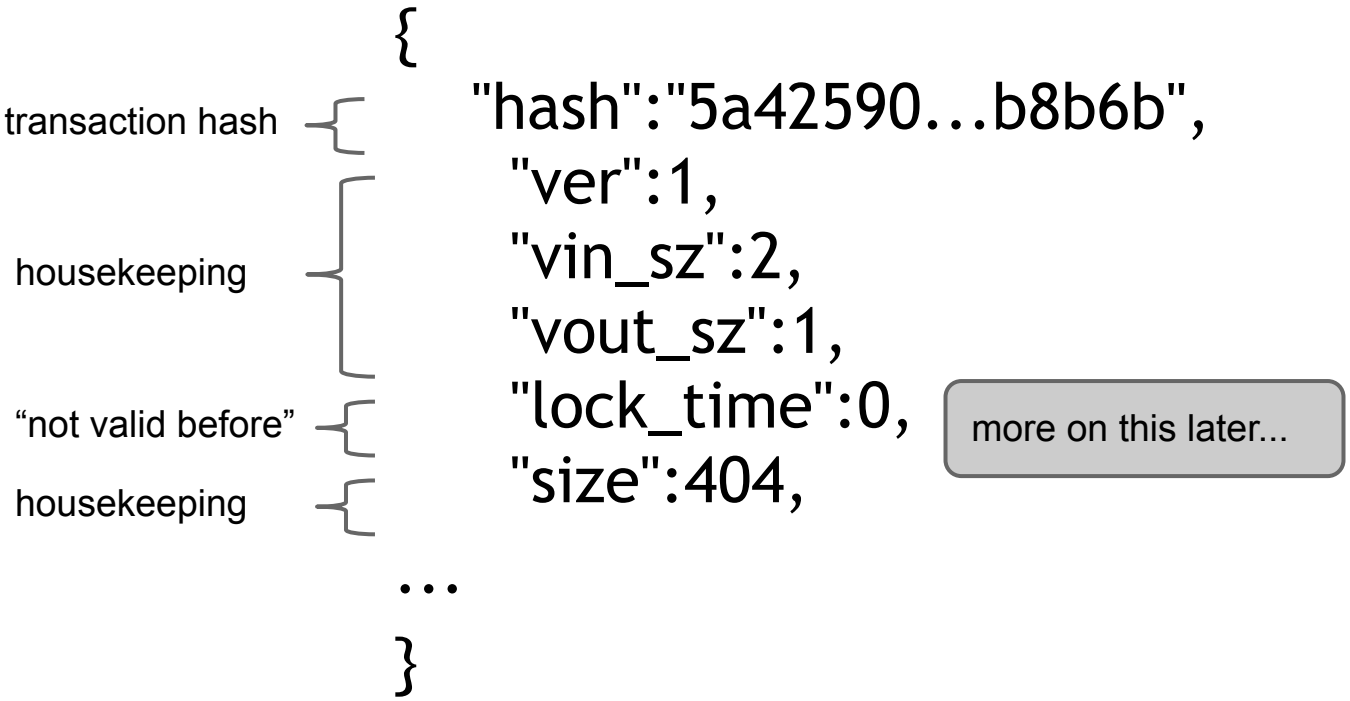
metadata

input(s)

output(s)


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



The real deal: transaction inputs

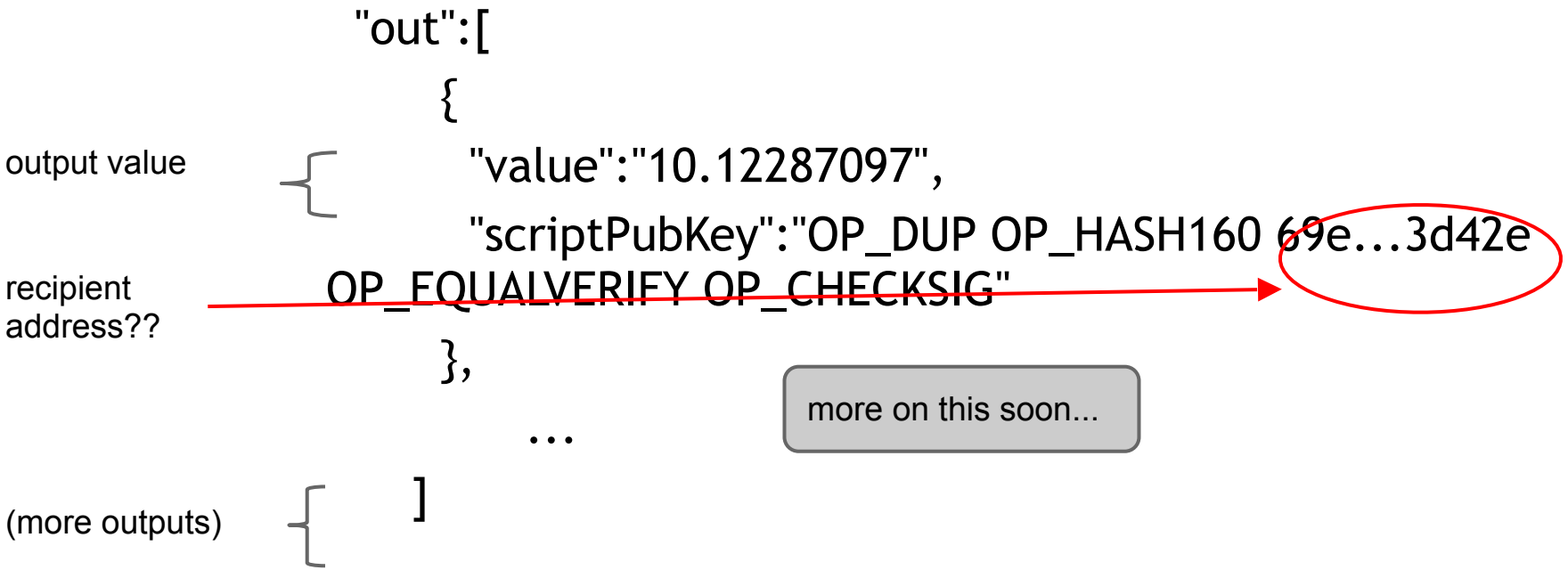
previous transaction

signature

(more inputs)

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

The real deal: transaction outputs

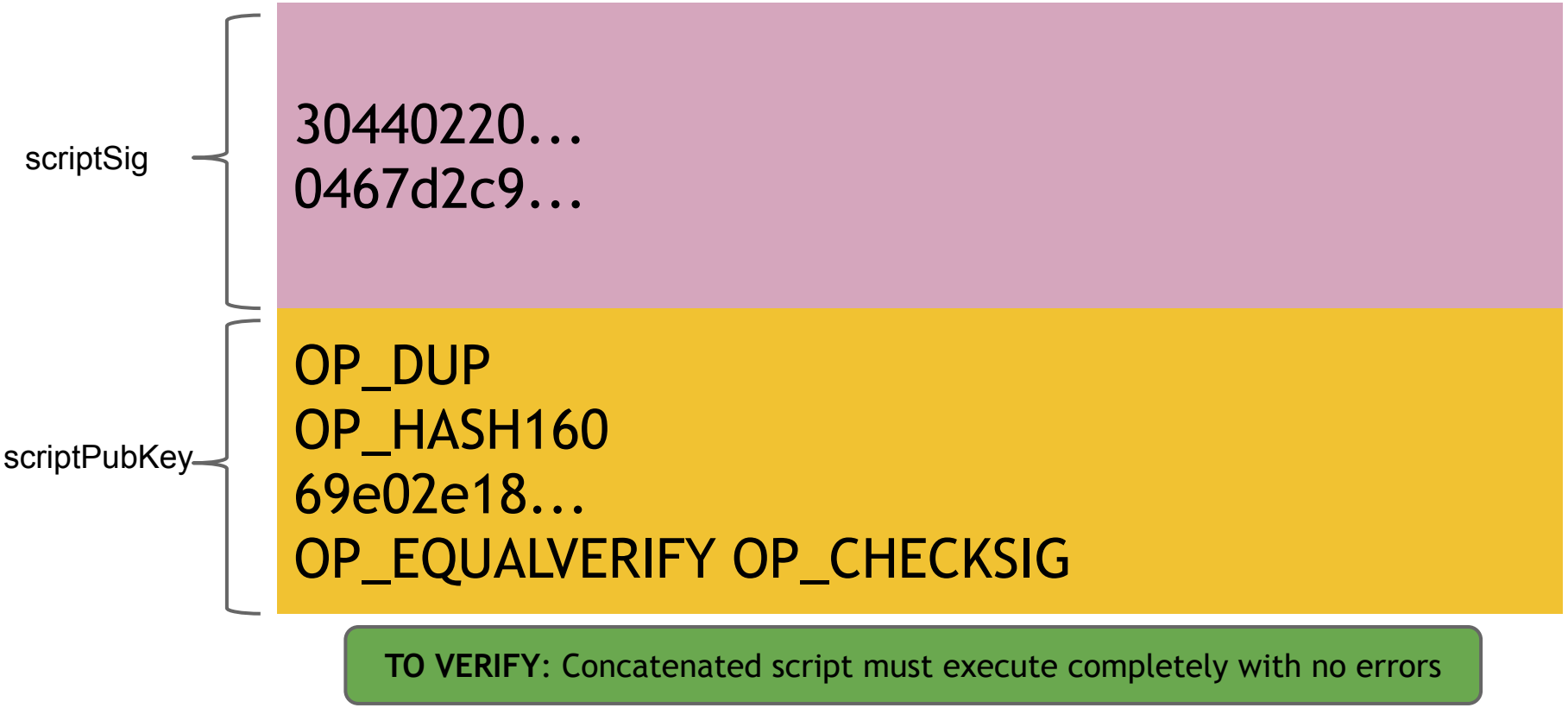


Bitcoin scripts

Output “addresses” are really *scripts*

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

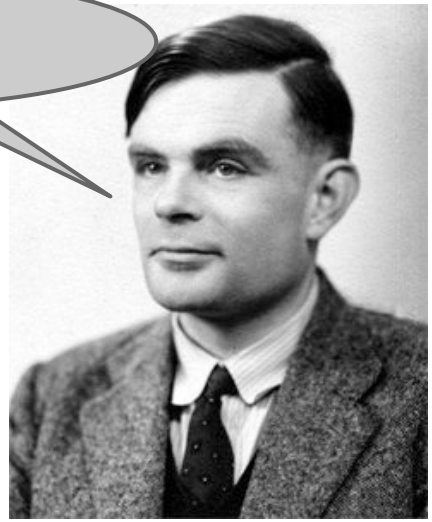
Input “addresses” are *also* scripts



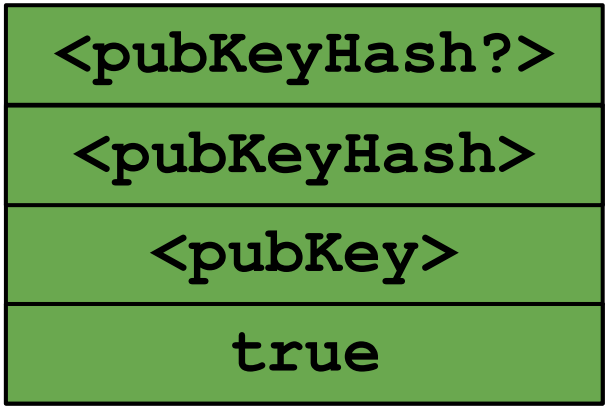
Bitcoin scripting language (“Script”)

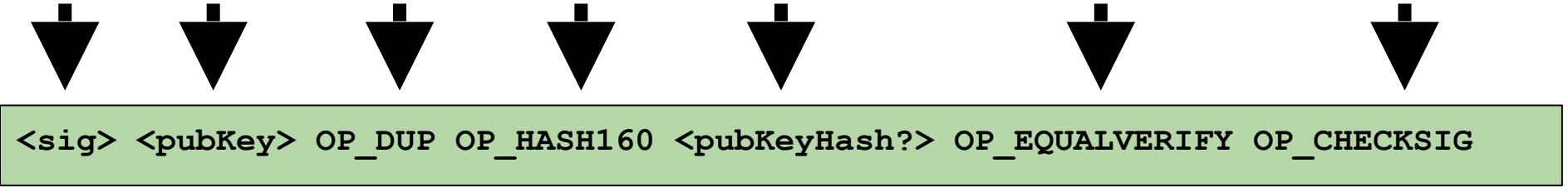
Design goals

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



Bitcoin script execution example





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

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

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-Ha
- Remainder are errors, proof-of-burn

More on this soon

* numbers from NBFMG and slightly out of date

Proof-of-burn

nothing’s going to redeem that ☹

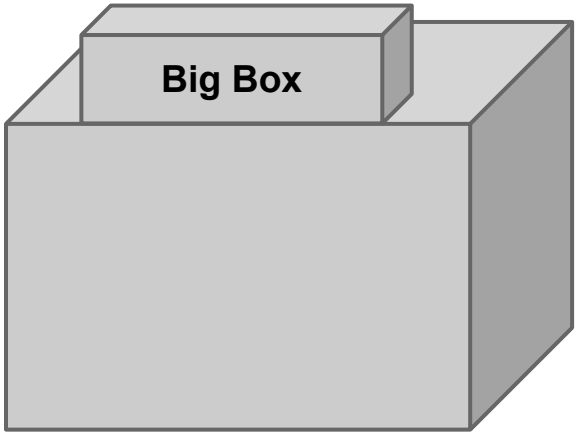
OP_RETURN
<arbitrary data>

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

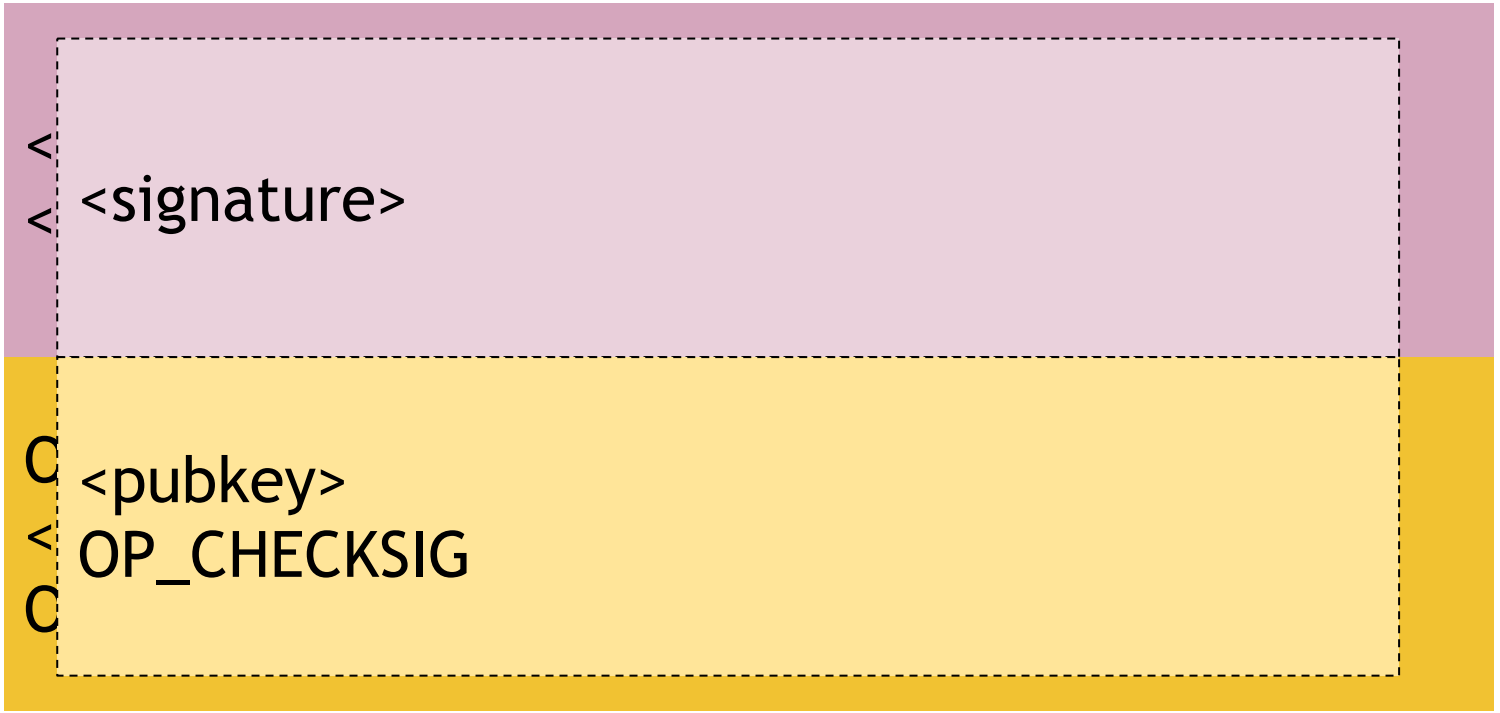
Should senders specify scripts?

I’m ready to pay for my purchases!

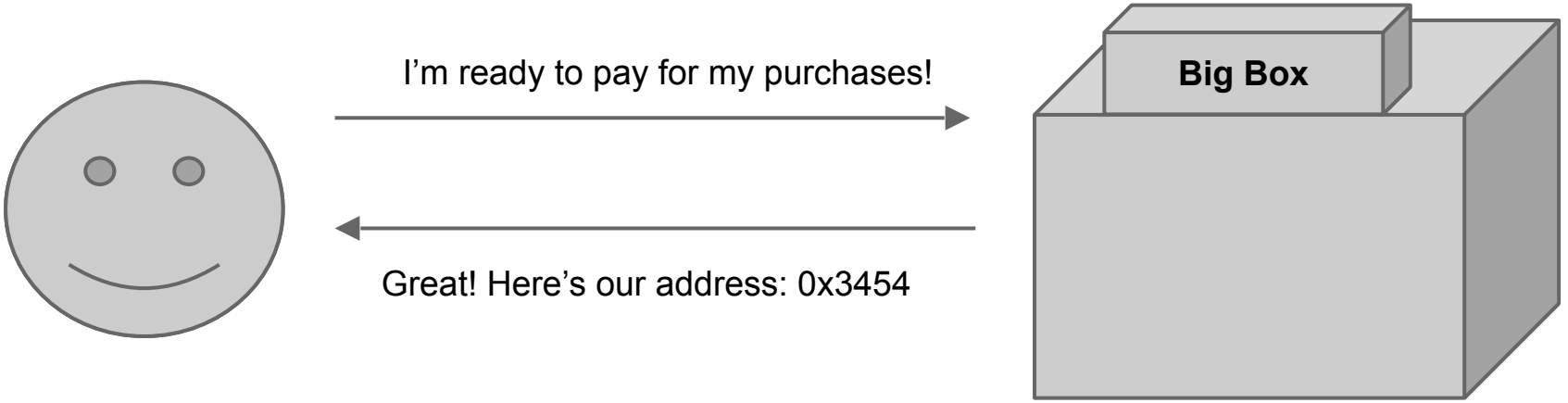


Cool! Well we’re using MULTISIG now, so include a script requiring 2 of our 3 account managers to approve. Don’t get any of those details wrong. Thanks for shopping at Big Box!

Idea: use the hash of redemption script



Pay to script hash



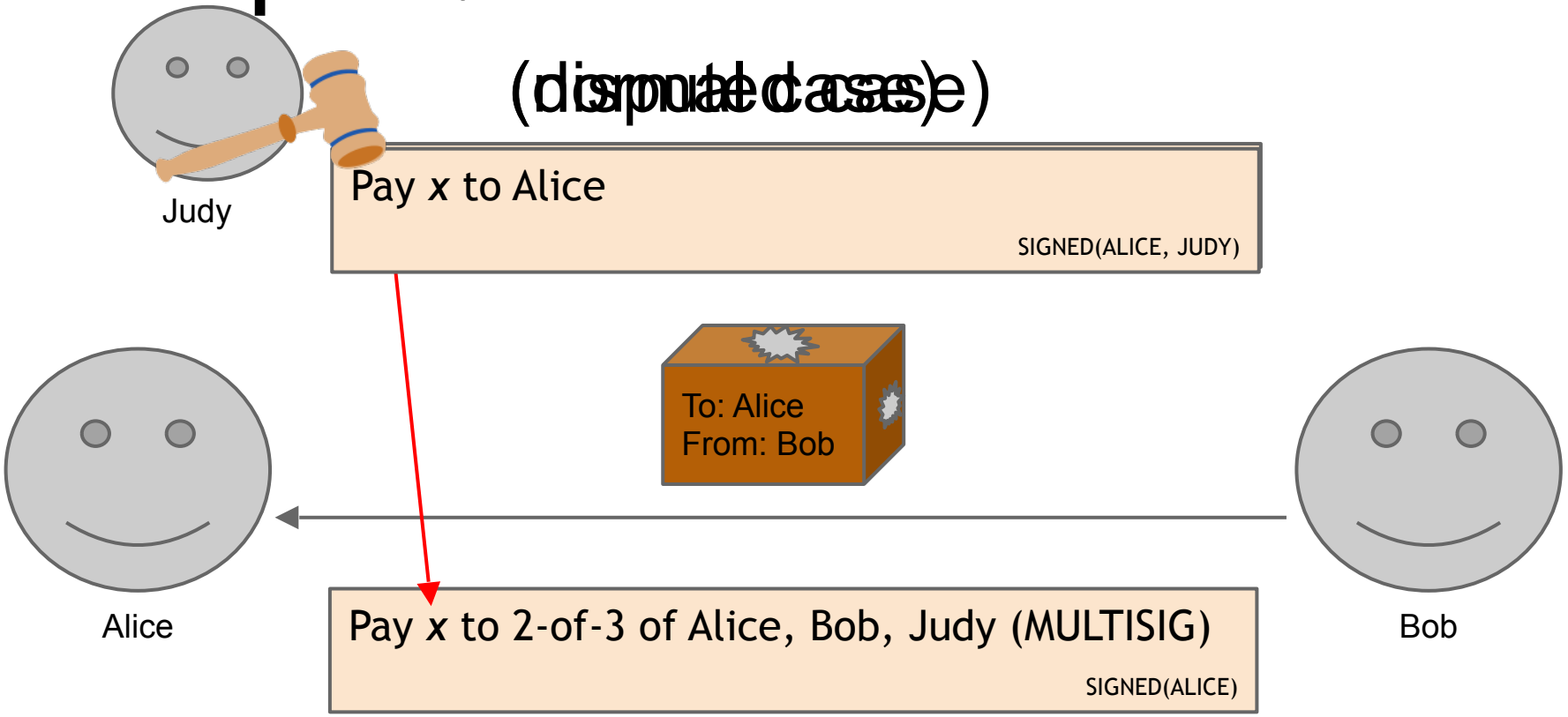
Block size limits and Segwit

Applications of Bitcoin scripts

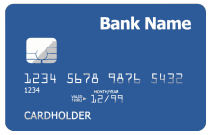
Example 1: “Fair” transactions

- Problem: Alice wants to buy a product from an online vendor Bob
- Alice doesn’t want to pay until after Bob ships
- Bob doesn’t want to ship until after Alice pays

Example 1: Fair transactions via Escrow



Example 2: Green addresses

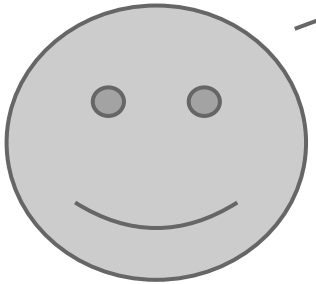


It's me, Alice! Could you make out a green payment to Bob?

Bank

004 days since last double spend!

Faraday cage



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.

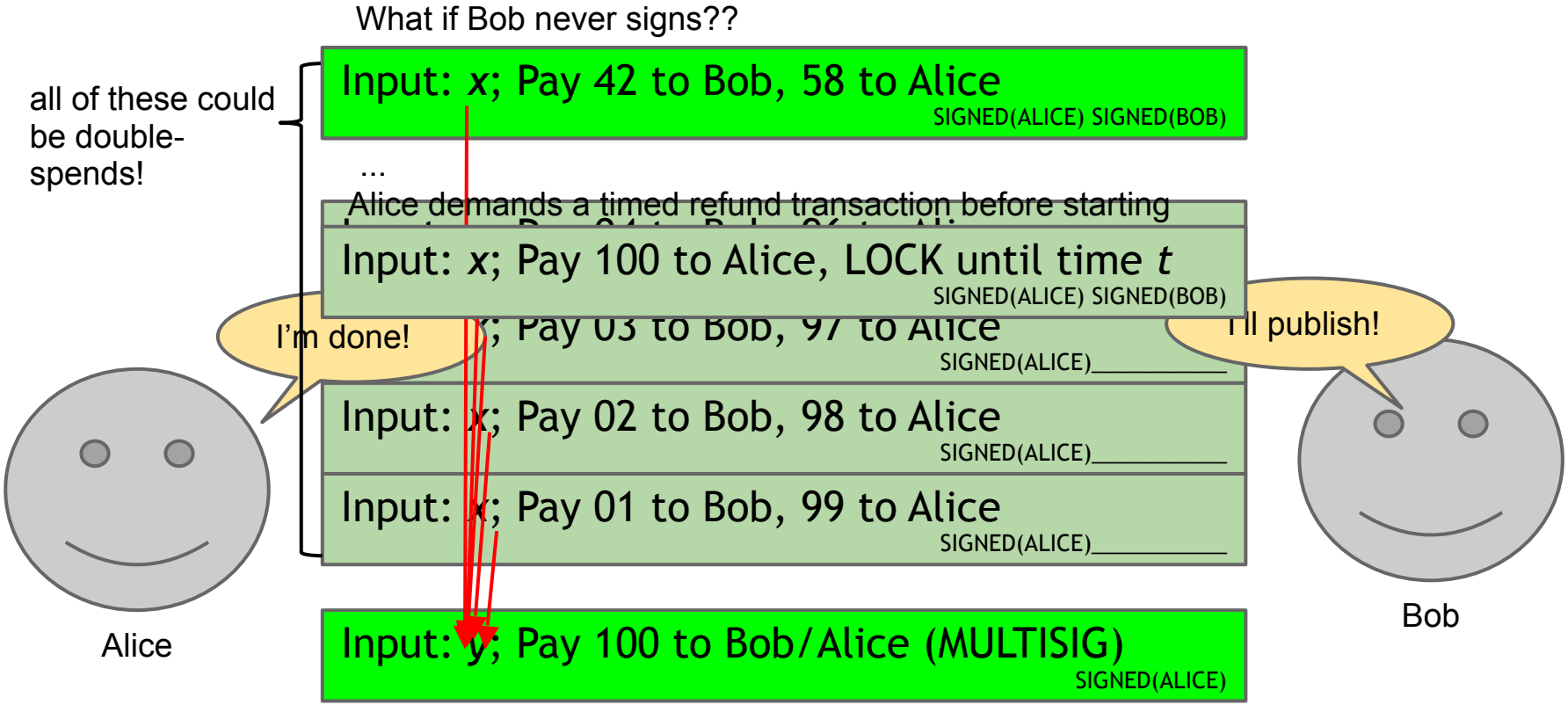
Example 2: Micro-payments

- Pay-as-you-go WIFI: Alice wants to pay WIFI provider (Bob) for each minute of WIFI service. But she doesn't want to incur a transaction fee for every minute
- Similarly, pay-as-you-go online subscriptions
- Ad-free websites

Example 2: Micro-payments with Bitcoin

- Main Idea: Instead of doing several transactions, do a single transaction for total payment (and thus incur only a single transaction fee)
- *How to implement it?*

Example 2: Micro-payments with Bitcoin



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

Micro-payments from Cryptocurrencies

More recent constructions, that achieve better properties

- Pass, shelat [CCS'16]
- Chiesa, Green, Liu, Miao, Miers, Mishra [EUROCRYPT'17]

More advanced scripts

- Fair multiplayer lotteries and fair multiparty computation [Andrychowichz-Dziembowski-Malinowski-Mazurek, S&P'14; Bentov-Kumaresan, CRYPTO'14]
- Hash pre-image challenges

"Smart contracts"

Later: More powerful smart contracts with Ethereum (Turing-complete scripting language)

Bitcoin blocks

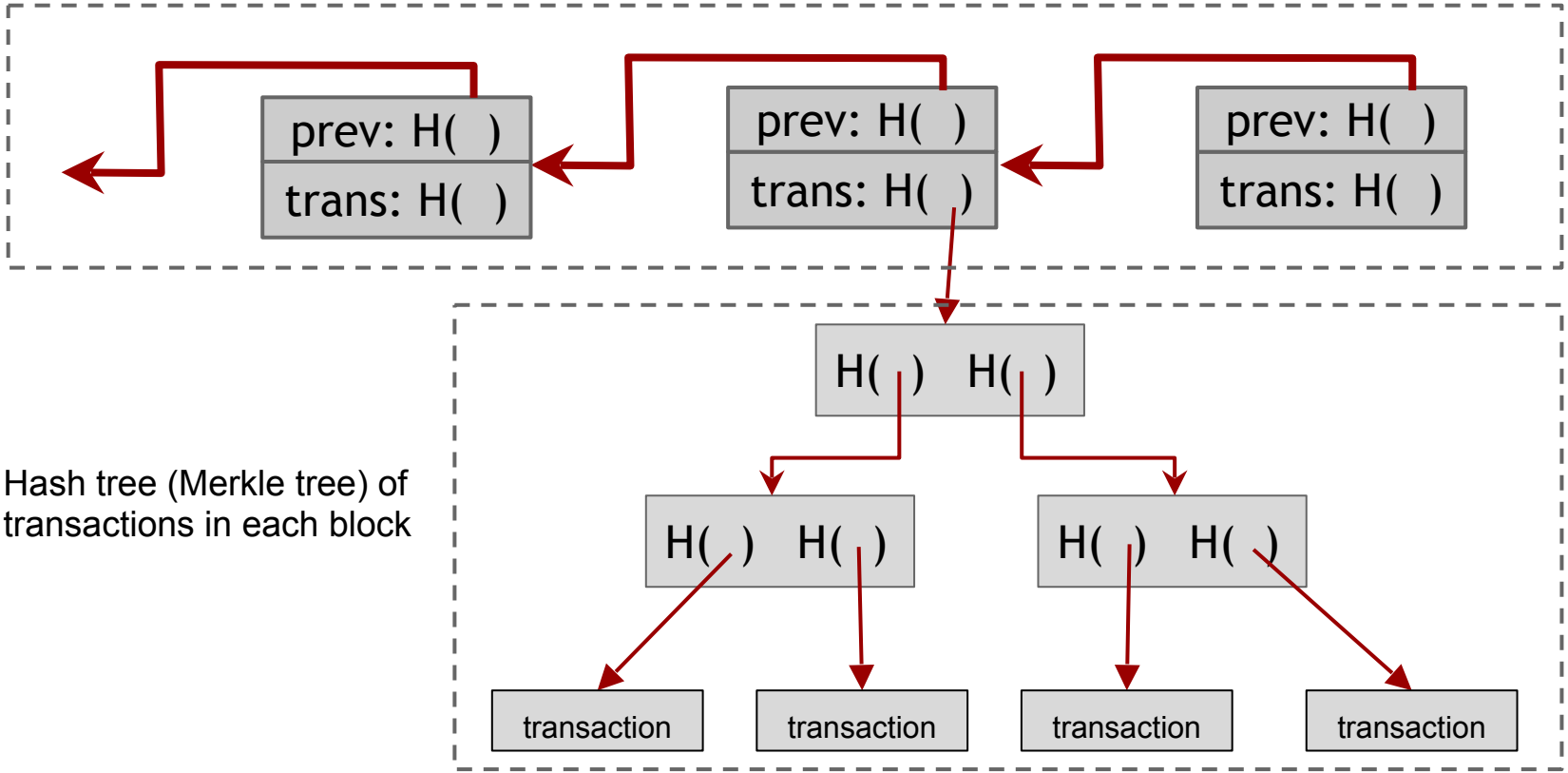
Bitcoin blocks

Why bundle transactions together?

- Single unit of work for miners
- Limit length of hash-chain of blocks
    - Faster to verify history

Bitcoin block structure

Hash chain of blocks



Hash tree (Merkle tree) of transactions in each block

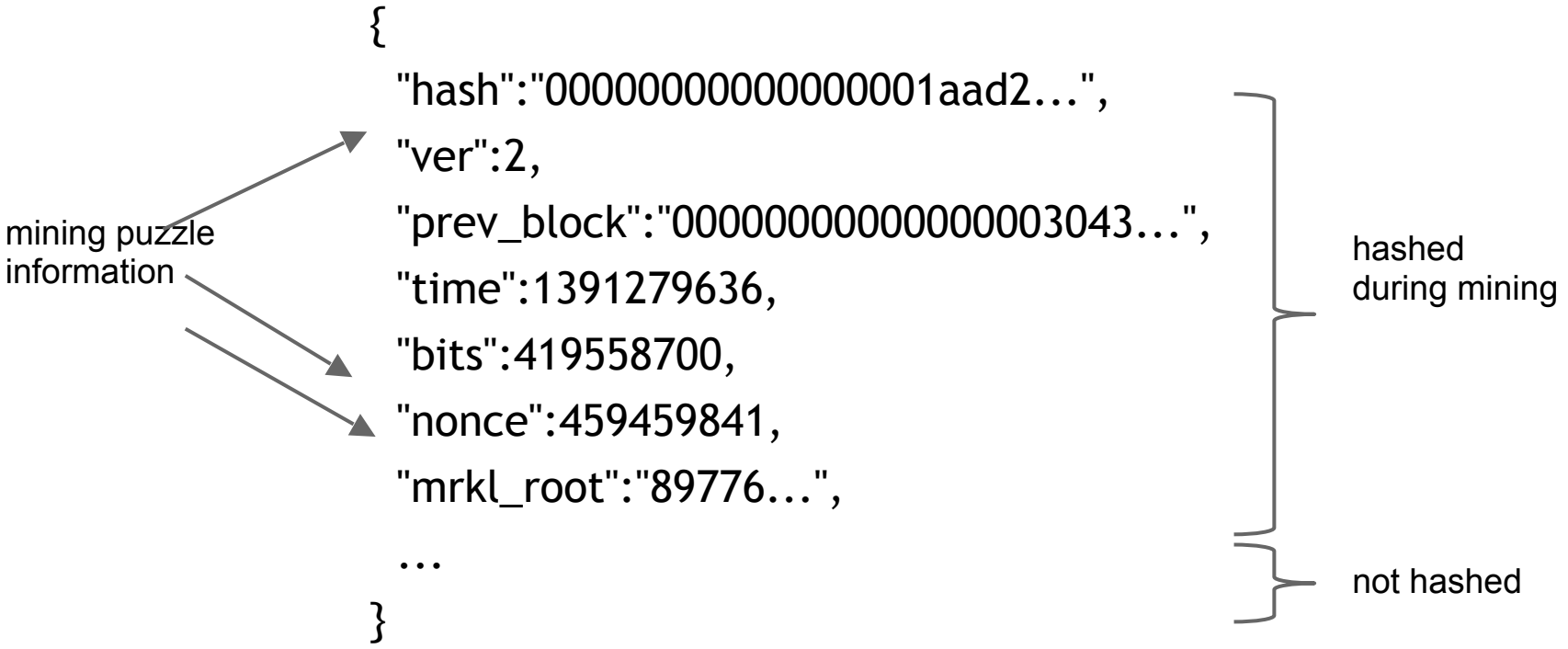
The real deal: a Bitcoin block

block header

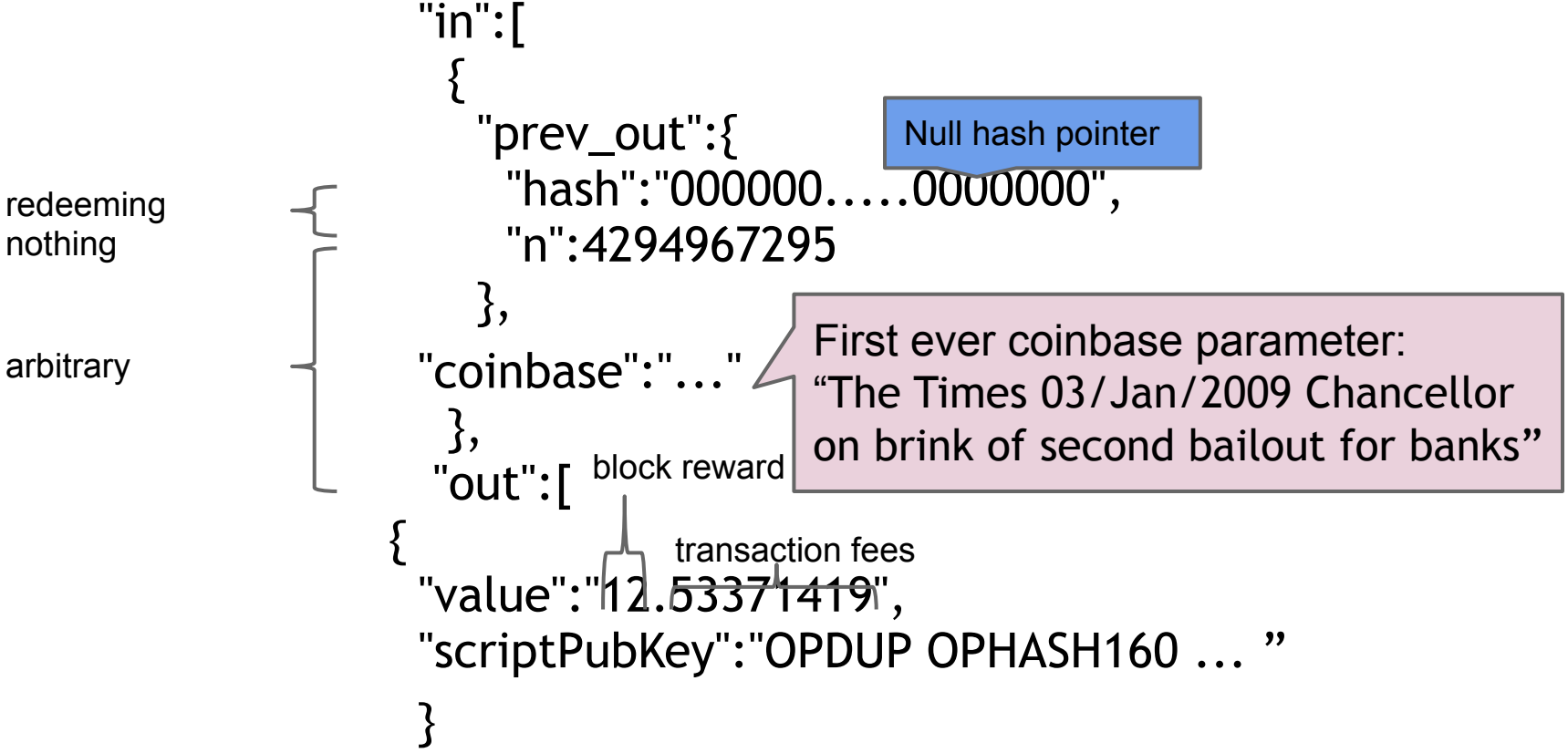
transaction data


```
{
  "hash":"00000000000000001aad2...",
  "ver":2,
  "prev_block":"00000000000000003043...",
  "time":1391279636,
  "bits":419558700,
  "nonce":459459841,
  "mrkl_root":"89776...",
  "n_tx":354,
  "size":181520,
  "tx":[
    ...
  ],
  "mrkl_tree":[
    "6bd5eb25...",
    ...
    "89776cdb..."
  ]
}
```

The real deal: a Bitcoin block header



The real deal: coinbase transaction



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

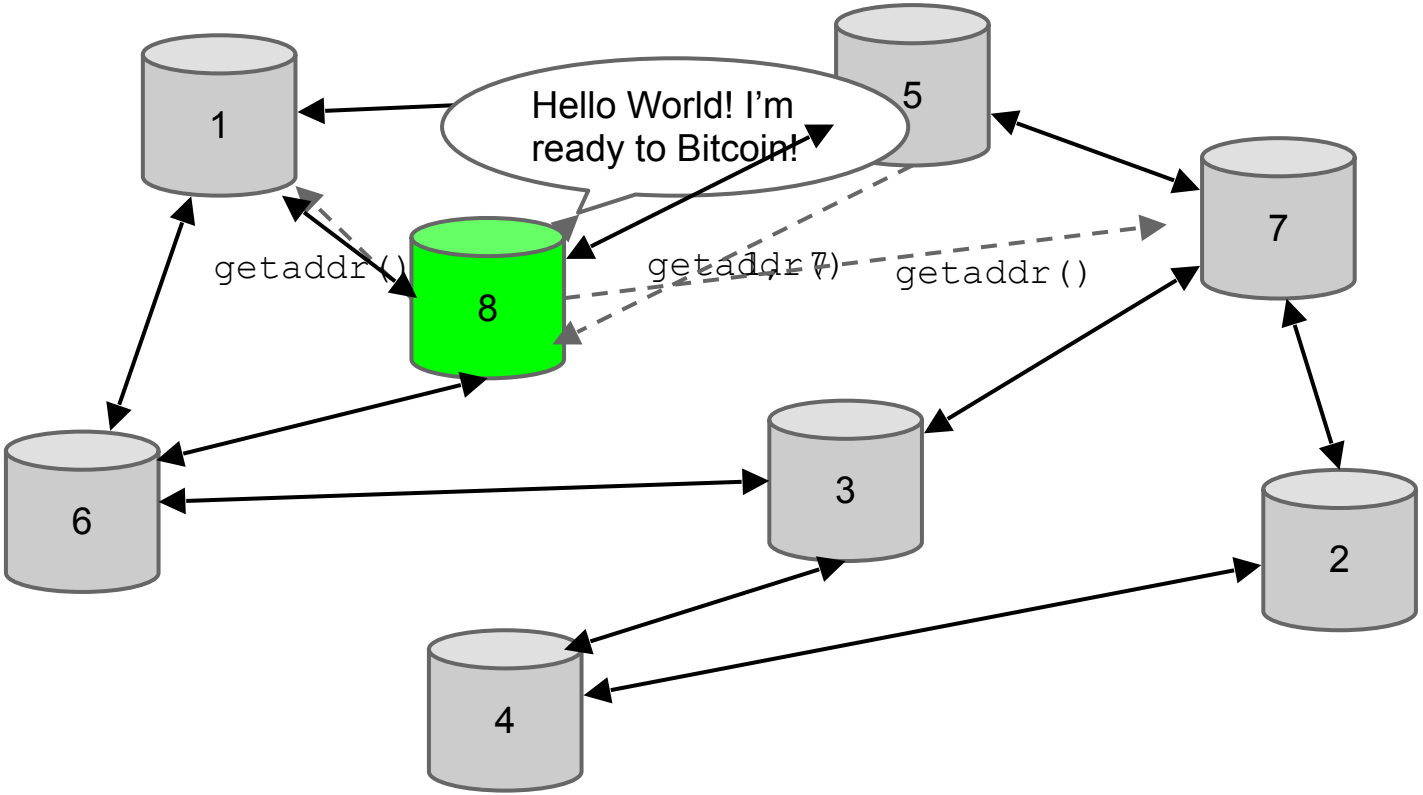
blockchain.info (and many other sites)

The Bitcoin network

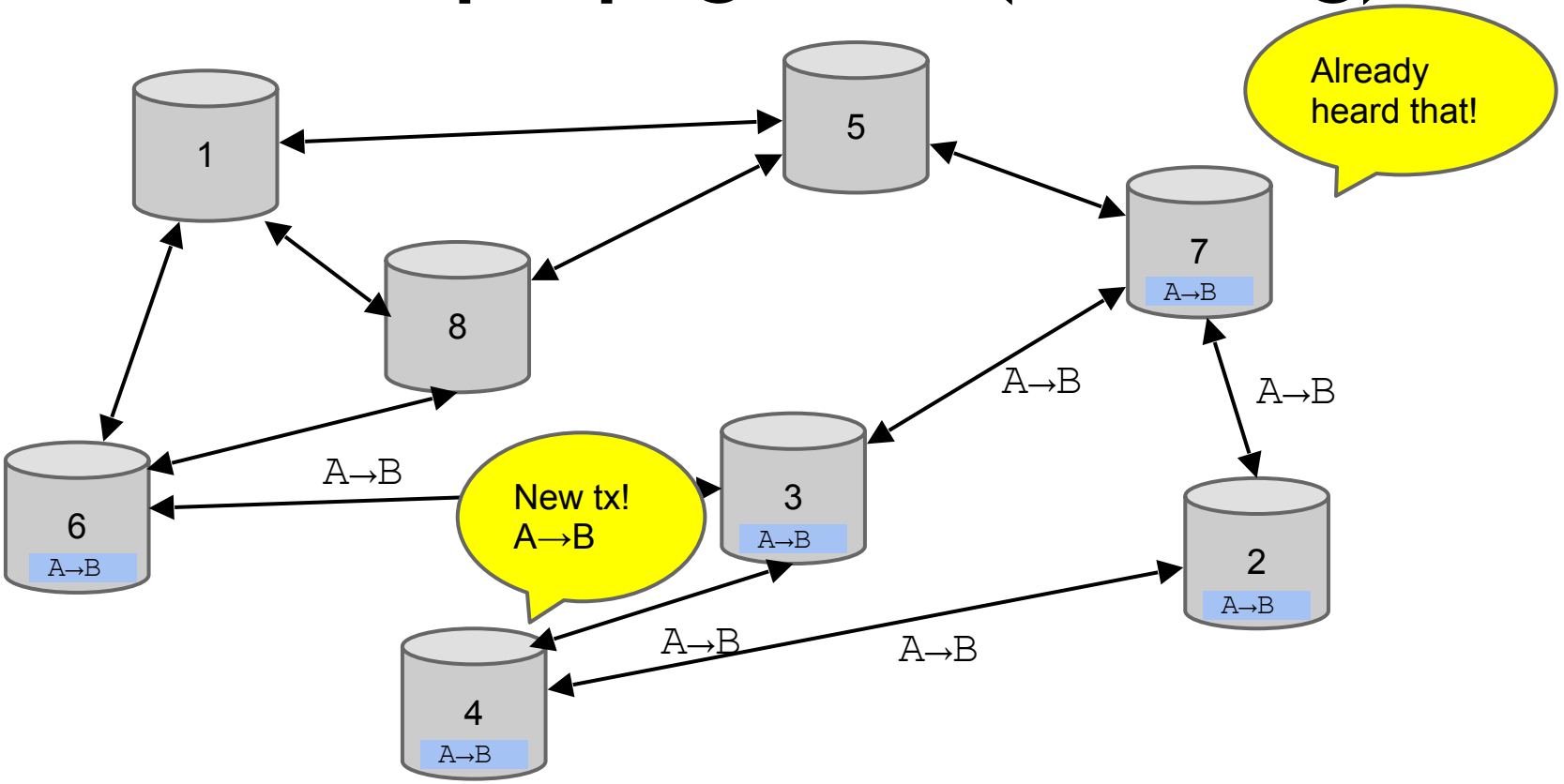
Bitcoin P2P network

- Ad-hoc protocol (runs on TCP port 8333)
- Ad-hoc network with random topology
- All nodes are equal
- New nodes can join at any time
- Forget non-responding nodes after 3 hr

Joining the Bitcoin P2P network

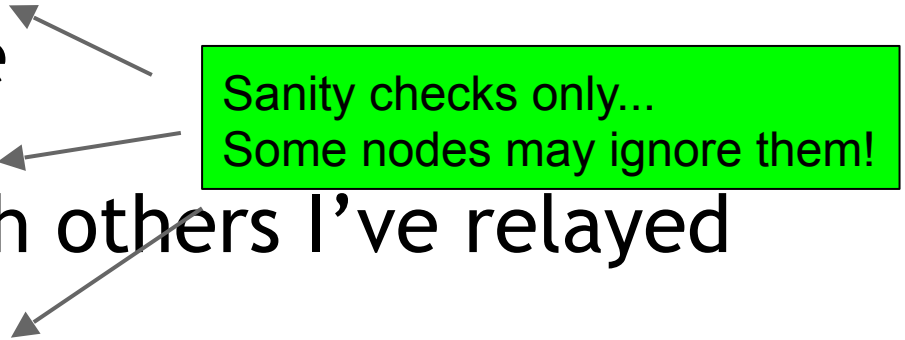


Transaction propagation (flooding)

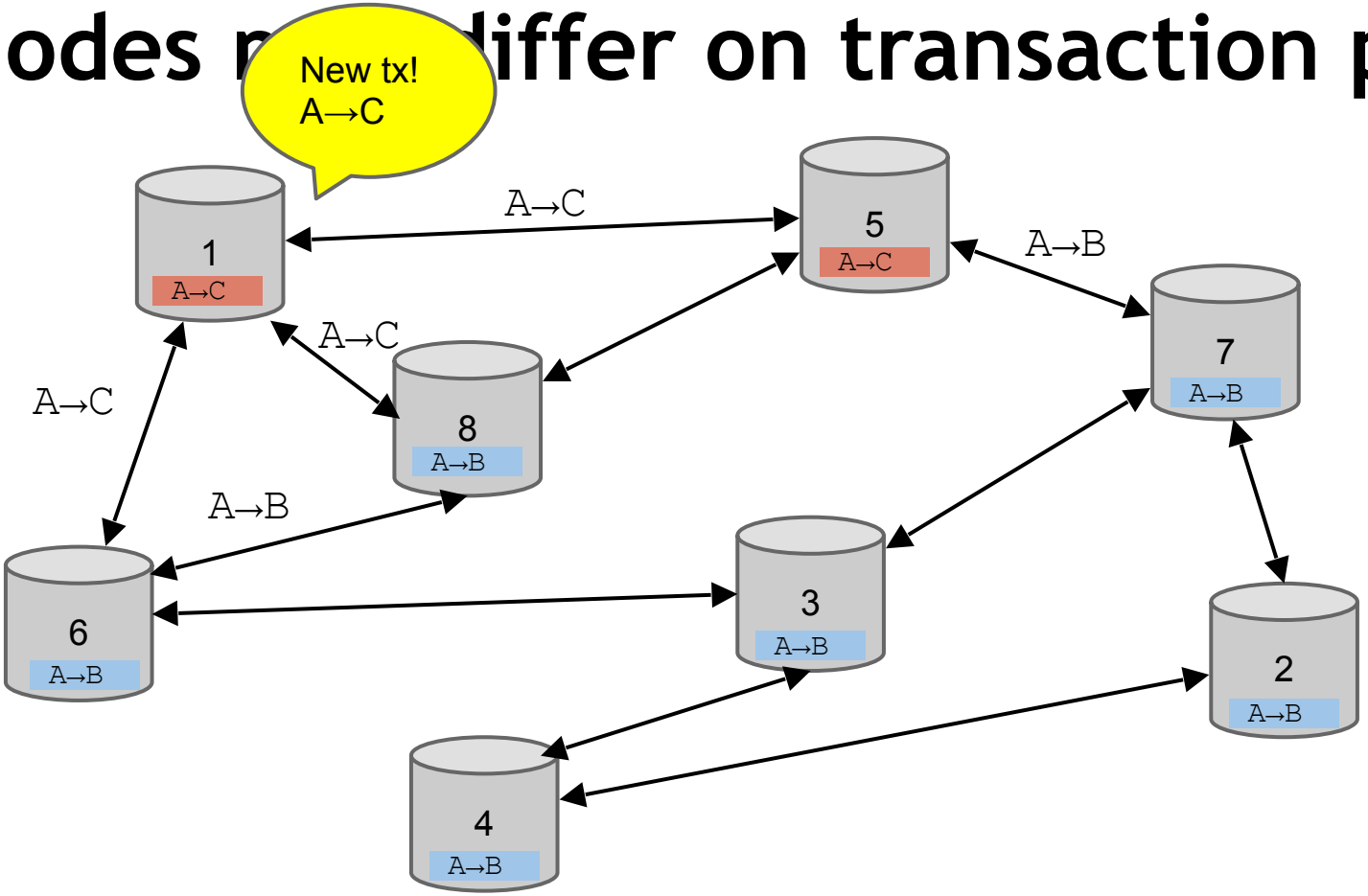


Should I relay a proposed transaction?

- Transaction valid with current block chain
- (default) script matches a whitelist
  - Avoid unusual scripts
- Haven’t seen before
  - Avoid infinite loops
- Doesn’t conflict with others I’ve relayed
  - Avoid double-spends



Nodes may differ on transaction pool



Race conditions

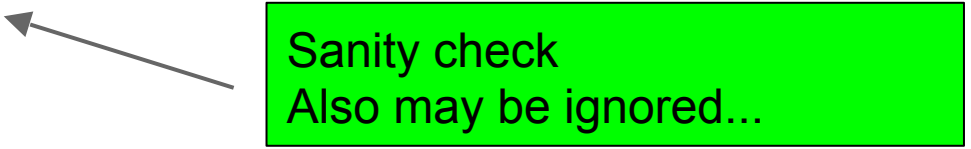
Transactions or blocks may *conflict*

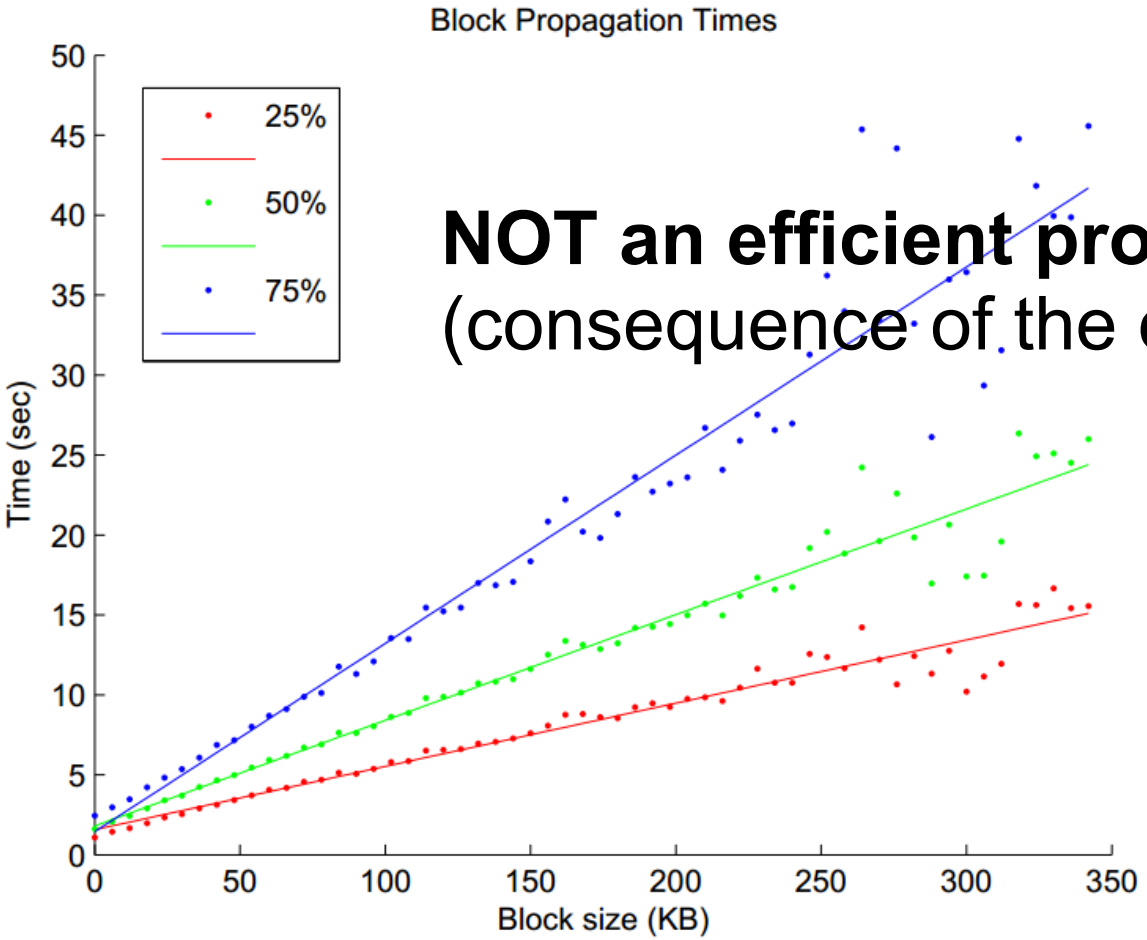
- Default behavior: accept what you hear first
- Network position matters
- Miners may implement other logic!

Block propagation nearly identical

Relay a new block when you hear it if:

- Block meets the hash target
- Block has all valid transactions
  - Run *all* scripts, even if you wouldn't relay
- Block builds on current longest chain
  - Avoid forks





NOT an efficient protocol
(consequence of the design)

Source: Yonatan Sompolinsky and Aviv Zohar: “Accelerating Bitcoin’s Transaction Processing” 2014

How big is the network?

- Unclear how to measure exactly
- Estimates-up to 1M IP addresses/month
- Only about 5-10k “full nodes”
    - Permanently connected
    - Fully-validate
- This number may be dropping!

Fully-validating nodes

- Permanently connected
- Store entire block chain
- Hear and forward every node/transaction

Thin/SPV clients (not fully-validating)

Idea: don’t store everything

- Store block headers only
- Request transactions as needed
  - To verify incoming payment
- Trust fully-validating nodes

Limitations & improvements

Hard-coded limits in Bitcoin

- 10 min. average creation time per block
- 1 M bytes in a block
- 20,000 signature operations per block
- 23M total bitcoins maximum
- 50,25,12.5... bitcoin mining reward

These affect economic balance of power too much to change now

Throughput limits in Bitcoin

- 1 M bytes/block (10 min)
- >250 bytes/transaction
- 7 tra

Improving throughput: A strong motivation for Altcoins

Compare

- VISA: 2,000-10,000 transactions/sec
- PayPal: 50-100 transaction/sec

Cryptographic limits in Bitcoin

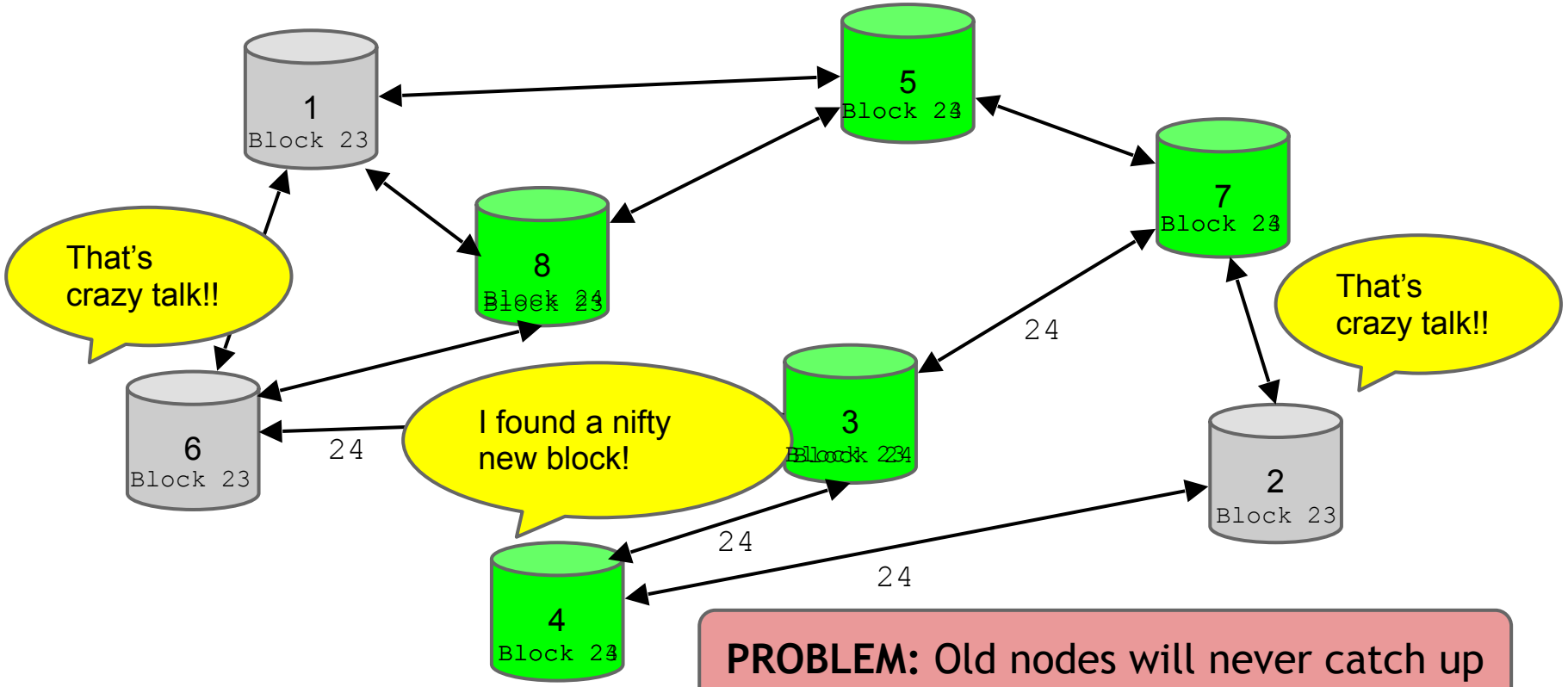
- Only 1 signature algorithm (ECDSA/P256)
- Hard-coded hash functions

Some of these crypto primitives used here might break by 2040 (e.g., collision-found in hash function, or powerful quantum computer breaks ECDSA)...

Why not update Bitcoin software to overcome these limitations?

- Many of these changes require “hard forks”, which are currently considered unacceptable

“Hard-forking” changes to Bitcoin



Soft forks

Observation: we can add new features which only *limit* the set of valid transactions

Need majority of nodes to enforce new rules

Old nodes will approve

RISK: Old nodes might mine now-invalid blocks

Soft fork example: pay to script hash

<signature>
<<pubkey> OP_CHECKSIG>

OP_HASH160
<hash of redemption script>
OP_EQUAL

Old nodes will just approve the hash, not run the embedded script

Soft fork possibilities

- New signature schemes
- Extra per-block metadata
  - Shove in the coinbase parameter
  - Commit to unspent transaction tree in each block

Hard forks

- New op codes
- Changes to size limits
- Changes to mining rate
- Many small bug fixes

Currently seem unlikely to happen

Many of these issues addressed by Altcoins