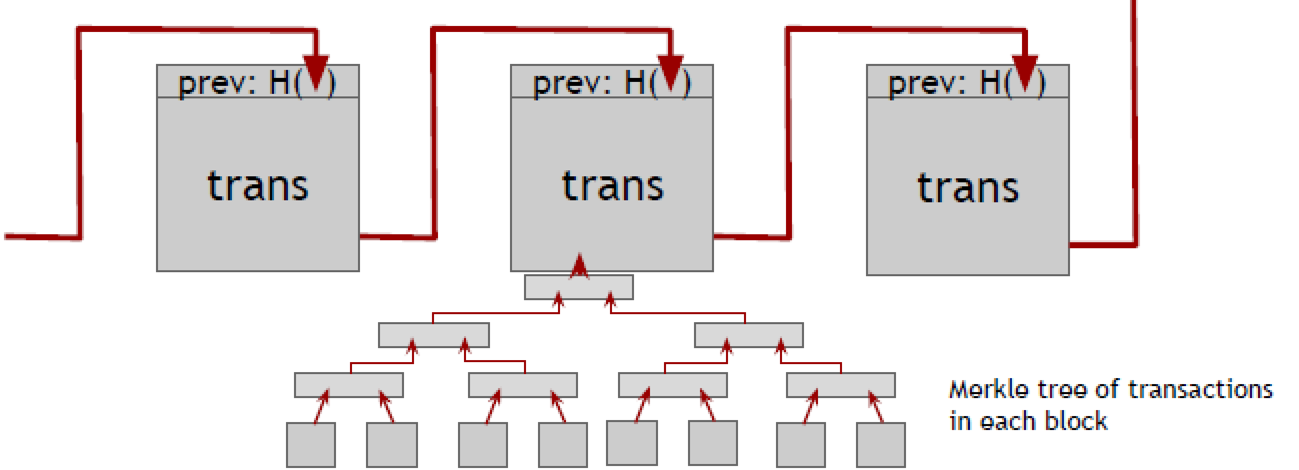
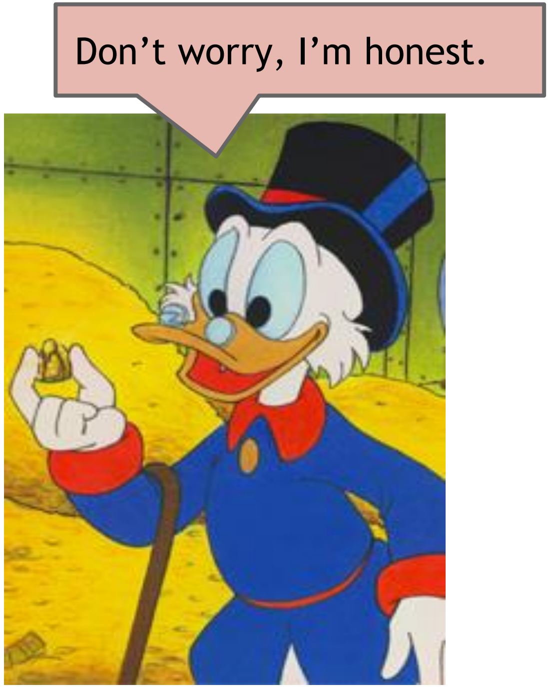
Intro to Crypto and Cryptocurrencies

Slides by Arvind Narayanan et al.

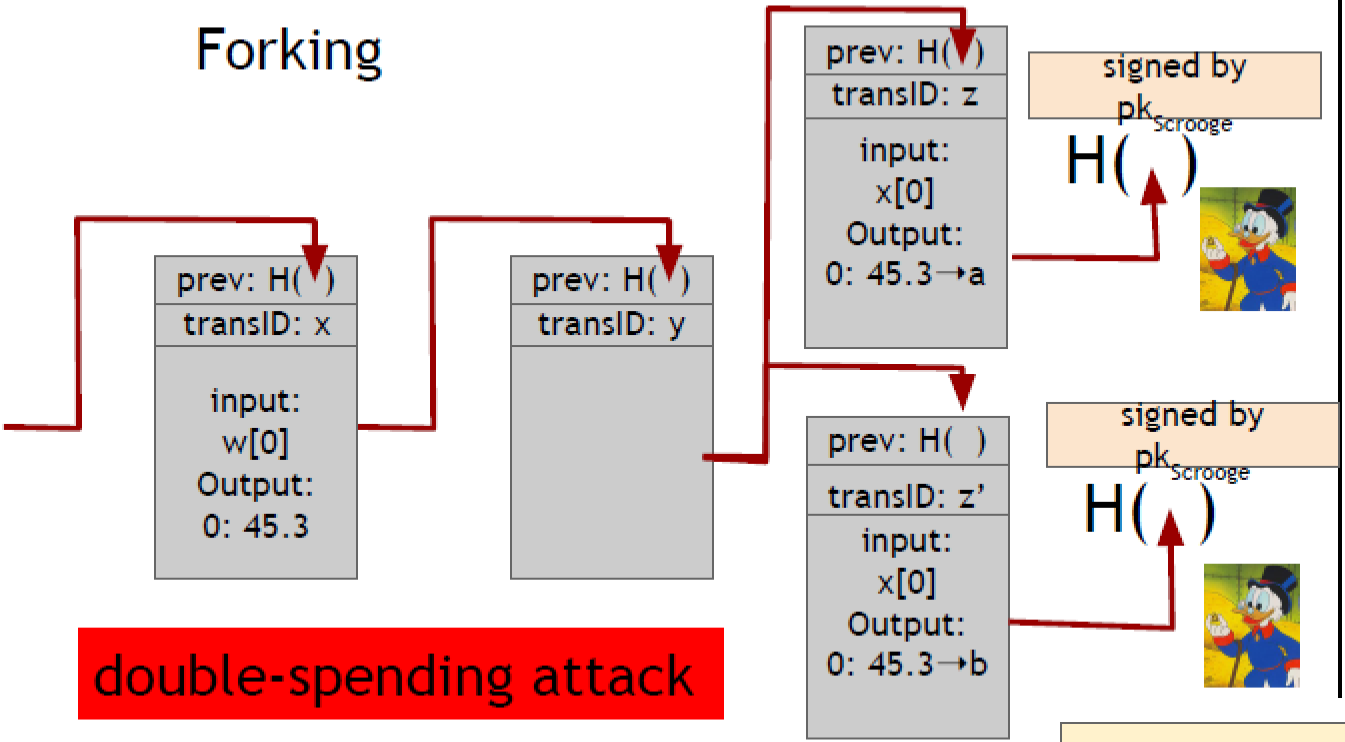
Scrooge publishes ledger of all transactions
(a blockchain, signed by Scrooge)



Merkle tree of transactions
in each block


Don’t worry, I’m honest.

Forking



What if Scrooge is malicious?

Don’t worry, I’m honest.

Crucial question:

Can we descroogify the currency, and operate without any central, trusted party?

Centralization vs. decentralization

Centralization vs. decentralization

Competing paradigms that underlie many digital technologies

Decentralization is not all-or-nothing

E-mail:

decentralized protocol, but dominated by centralized webmail services

Bitcoin Consensus

Consensus algorithm (simplified)

1. New transactions are broadcast to all nodes
2. Follow Flooding/Gossip Protocol to broadcast
3. Some nodes collect new transactions into a block
4. In each round a random node gets to broadcast its block
5. Other nodes accept the block only if all transactions in it are valid (unspent, valid signatures)
6. Nodes express their acceptance of the block by including its hash in the next block they create

Mining Process

Consensus Agreement

Why consensus is hard

Nodes may crash
Nodes may be malicious

Network is imperfect

- Not all pairs of nodes connected
- Faults in network
- Latency

No notion of global time

Many impossibility results

- Fischer-Lynch-Paterson (deterministic nodes): consensus impossible with a <u>single</u> faulty node

Some well-known protocols

Example: Paxos, Raft, ….

Makes certain compromises

Never produces inconsistent result, but can (rarely) get stuck

Bitcoin consensus: theory & practice

Bitcoin consensus works better in practice than in theory

Theory is still catching up

<u>BUT</u> theory is important, can help predict unforeseen attacks

Some things Bitcoin does differently

Introduces incentives

- Possible only because it's a currency!

Embraces randomness

- Does away with the notion of a specific start and end-point
- Consensus happens over long time scales — about 1 hour

Hashcash: Proof of work

- Based on the idea of HashCash, a Proof of Work concept invented by Adam Back in 1997 (http://www.hashcash.org/papers/hashcash.pdf)

- Originally proposed as an anti-spam throttling mechanism

- The core idea is that before accepting a transaction, the sender must first demonstrate a “cost” via a computationally “hard” problem that can simultaneously be easily verified.

- This generally referred to as a “Proof of Work”

Hashcash: Proof of work

- HashCash Cost Function: Interactive Vs. Non-interactive
- *s*: service name

$$\begin{cases} \mathcal{T} \leftarrow \mathsf{MINT}(s, w) & \text{mint token} \\ \mathcal{V} \leftarrow \mathsf{VALUE}(\mathcal{T}) & \text{token evaluation function} \end{cases}$$

$$\begin{cases} \mathsf{PUBLIC:} & \text{hash function } \mathcal{H}(\cdot) \text{ with output size } k \text{ bits} \\ & \\ \mathcal{T} \leftarrow \mathsf{MINT}(s, w) & \textbf{find } x \in_R \{0,1\}^\star \textbf{ st } \mathcal{H}(s\|x) \stackrel{\text{left}}{=}_w 0^k \\ & \textbf{return } (s, x) \\ \mathcal{V} \leftarrow \mathsf{VALUE}(\mathcal{T}) & \mathcal{H}(s\|x) \stackrel{\text{left}}{=}_v 0^k \\ & \textbf{return } v \end{cases}$$

Hashcash: Proof of work

- Hashcash Cost Function: Interactive Vs. Non-interactive

$$\begin{cases} \mathcal{C} \leftarrow \mathsf{CHAL}(s, w) & \text{server challenge function} \\ \mathcal{T} \leftarrow \mathsf{MINT}(\mathcal{C}) & \text{mint token based on challenge} \\ \mathcal{V} \leftarrow \mathsf{VALUE}(\mathcal{T}) & \text{token evaluation function} \end{cases}$$

$$\begin{cases} \mathcal{C} \leftarrow \mathsf{CHAL}(s, w) & \textbf{choose } c \in_R \{0,1\}^k \\ & \textbf{return } (s, w, c) \\ \mathcal{T} \leftarrow \mathsf{MINT}(C) & \textbf{find } x \in_R \{0,1\}^\star \textbf{ st } \mathcal{H}(s||c||x) \overset{\text{left}}{=}_w 0^k \\ & \textbf{return } (s, x) \\ \mathcal{V} \leftarrow \mathsf{VALUE}(T) & \mathcal{H}(s||c||x) \overset{\text{left}}{=}_v 0^k \\ & \textbf{return } v \end{cases}$$

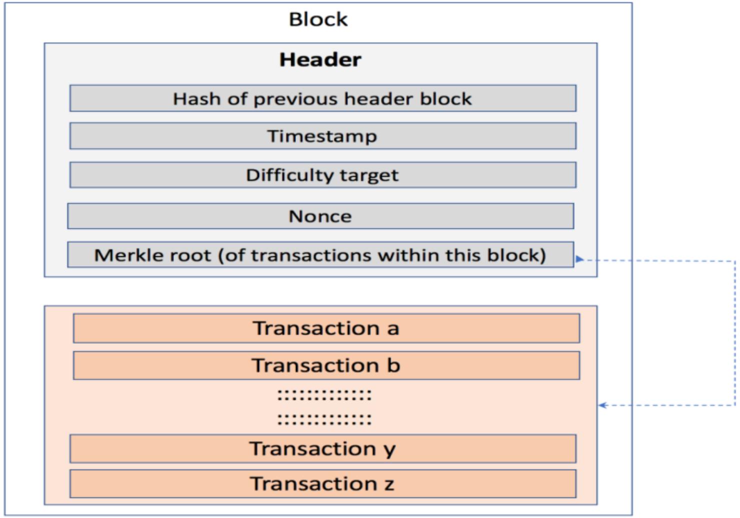
Prevents **DOS motivated attack or** pre-computation attacks

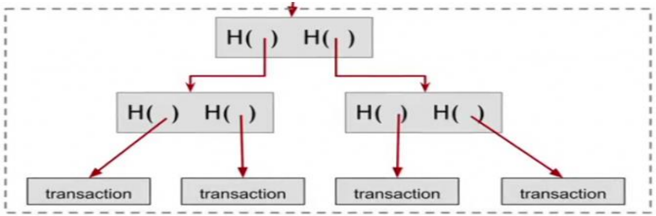
The real deal: a Bitcoin block

block header

transaction data

```
{
  "hash":"0000000000000001aad2...",
  "ver":2,
  "prev_block":"00000000000000003043...",
  "time":1391279636,
  "bits":419558700,
  "nonce":459459841,
  "mrkl_root":"89776...",
  "n_tx":354,
  "size":181520,
  "tx":[
    ...
  ],
  "mrkl_tree":[
    "6bd5eb25...",
    ...
    "89776cdb..."
  ]
}
```





Hash tree (Merkle tree)

Bitcoin blocks

Why bundle transactions together?

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

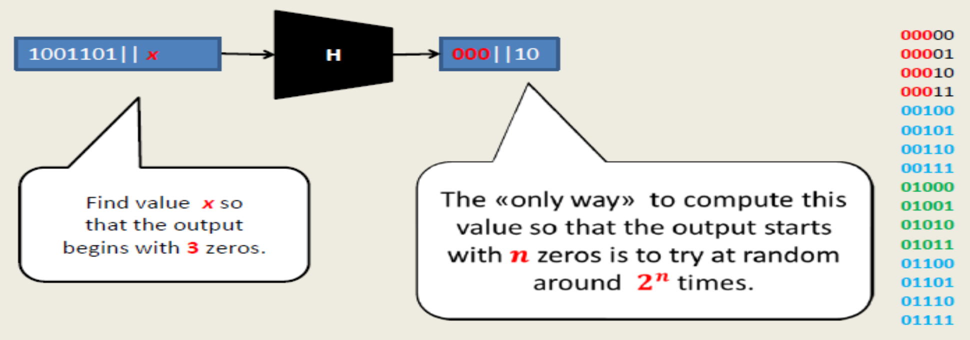
The real deal: a Bitcoin block

block header

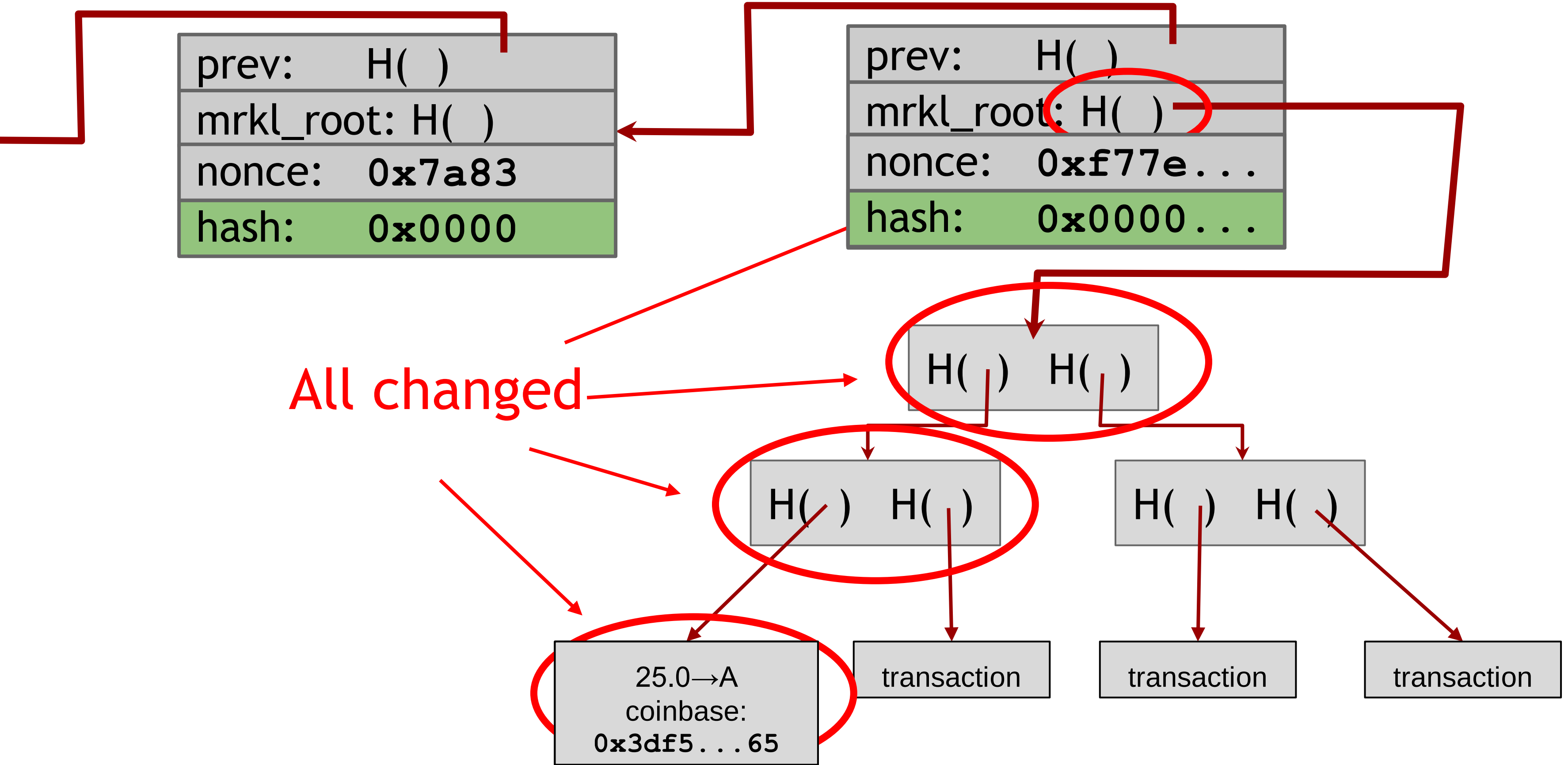
transaction data


```
{
  "hash":"0000000000000001aad2…",
  "ver":2,
  "prev_block":"00000000000000003043…",
  "time":1391279636,
  "bits":419558700,
  "nonce":459459841,
  "mrkl_root":"89776…",
  "n_tx":354,
  "size":181520,
  "tx":[
    …
  ],
  "mrkl_tree":[
    "6bd5eb25…",
    …
    "89776cdb…"
  ]
}
```


Proof of Work **[Back2002]**



Finding a valid block: Proof-of-Work



Coinbase

```
"in":[
  {
    "prev_out":{
      "hash":"000000.....0000000",
      "n":4294967295
    },
"coinbase":"..."
  },
 "out":[
{
"value":"25.03371419",
"scriptPubKey":"OPDUP OPHASH160 ... ”
}
```

Null hash pointer

redeeming nothing

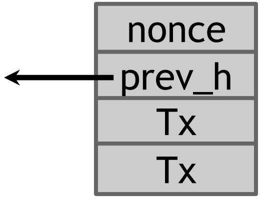
arbitrary

block reward

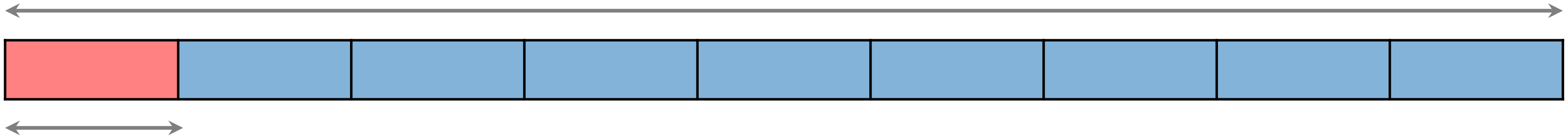
transaction fees

Hash puzzles: Bitcoin Proof-of-Work

To create block, find nonce s.t.
H(nonce ‖ prev_hash ‖ tx ‖ … ‖ tx) is very small



Output space of hash



Target
space

If hash function is secure:
only way to succeed is to try enough nonces until you get lucky

Target space is 1% of overall output space, You would have to try 100 nonces before you are likely to get valid result.

PoW property 1: difficult to compute

Other words, the size of the target space is less than $1/10^{20}$ of the size of the overall space

As of Aug 2014: about 10^{20} hashes/block

Only some nodes bother to compete – miners

PoW property 2: parameterizable cost

Nodes automatically re-calculate the target every two weeks

Goal: <u>average</u> time between blocks = 10 minutes

Prob (Alice wins next block) =
fraction of global hash power she controls

Alice with 0.1% of total hash power will find roughly one in every 1000 blocks.

Key security assumption

Attacks infeasible if majority of miners <u>weighted by hash power</u> follow the protocol

PoW property 3: trivial to verify

Nonce must be published as part of block

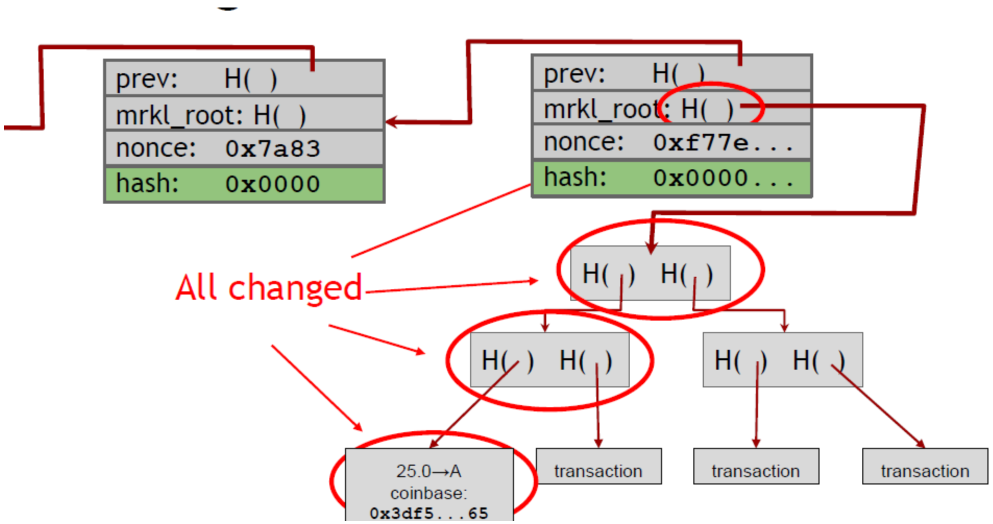
Other miners simply verify that
$H(\text{nonce} \parallel \text{prev\_hash} \parallel \text{tx} \parallel \ldots \parallel \text{tx}) < \text{target}$

The real deal: a Bitcoin block

block header

transaction data

```
{
  "hash":"0000000000000001aad2...",
  "ver":2,
  "prev_block":"00000000000000003043...",
  "time":1391279636,
  "bits":419558700,
  "nonce":459459841,
  "mrkl_root":"89776...",
  "n_tx":354,
  "size":181520,
  "tx":[
    ...
  ],
  "mrkl_tree":[
    "6bd5eb25...",
    ...
    "89776cdb..."
  ]
}
```



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

Bitcoin Mining

Sanity check
Also may be ignored...

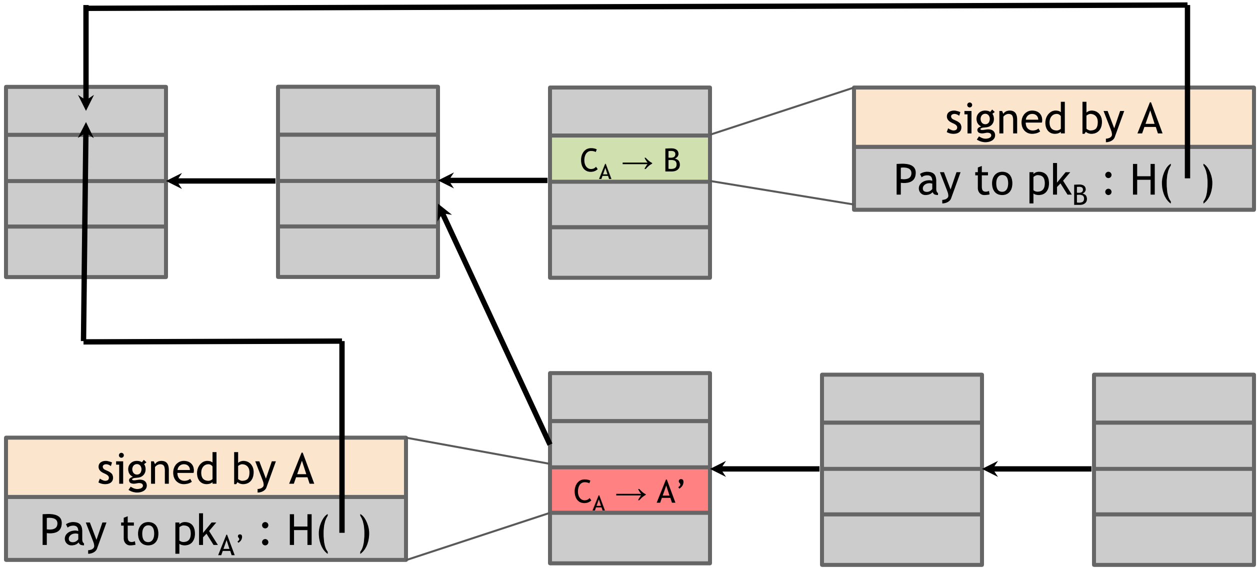
Mining Bitcoins in 6 easy steps

1. Join the network, listen for transactions
   a. Validate all proposed transactions
2. Listen for new blocks, maintain block chain
   a. When a new block is proposed, validate it
3. Assemble a new valid block
4. Find the nonce to make your block valid
5. Hope everybody accepts your new block
6. Profit!

Useful to Bitcoin network

Incentives in proof of work

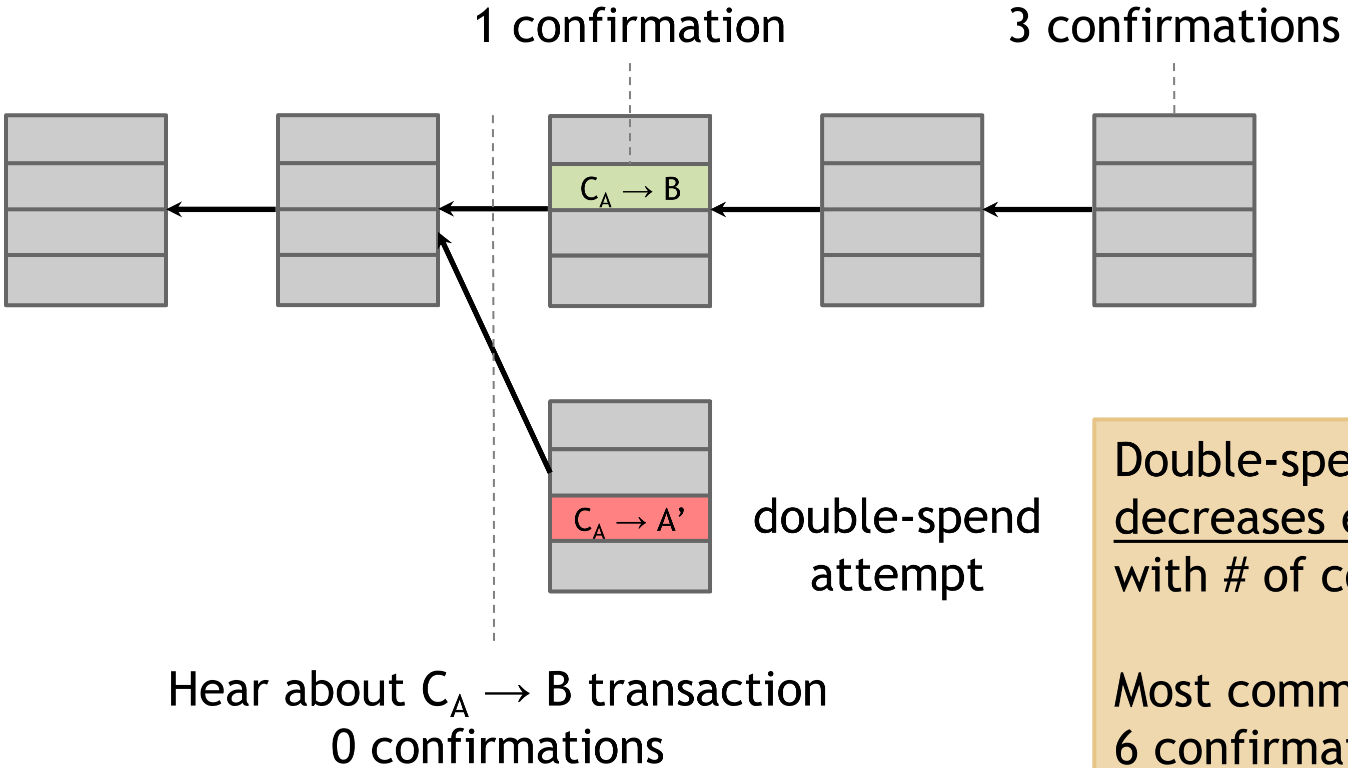
What can a malicious node do?



Double-spending attack

Honest nodes will extend the <u>longest valid branch</u>

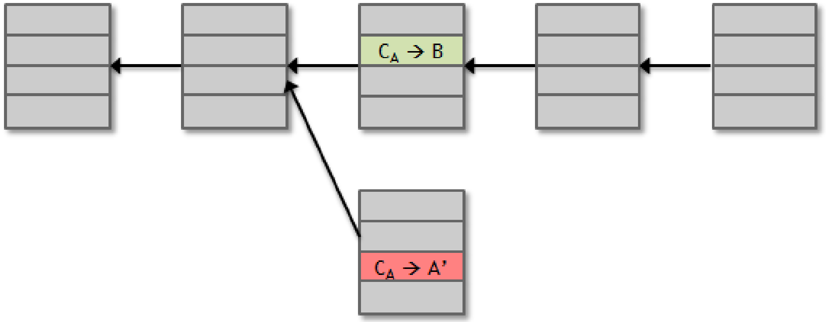
From Bob the merchant's point of view



Double-spend probability <u>decreases exponentially</u> with # of confirmations

Most common heuristic:
6 confirmations

Security



Protection against invalid transactions is cryptographic, but enforced by consensus

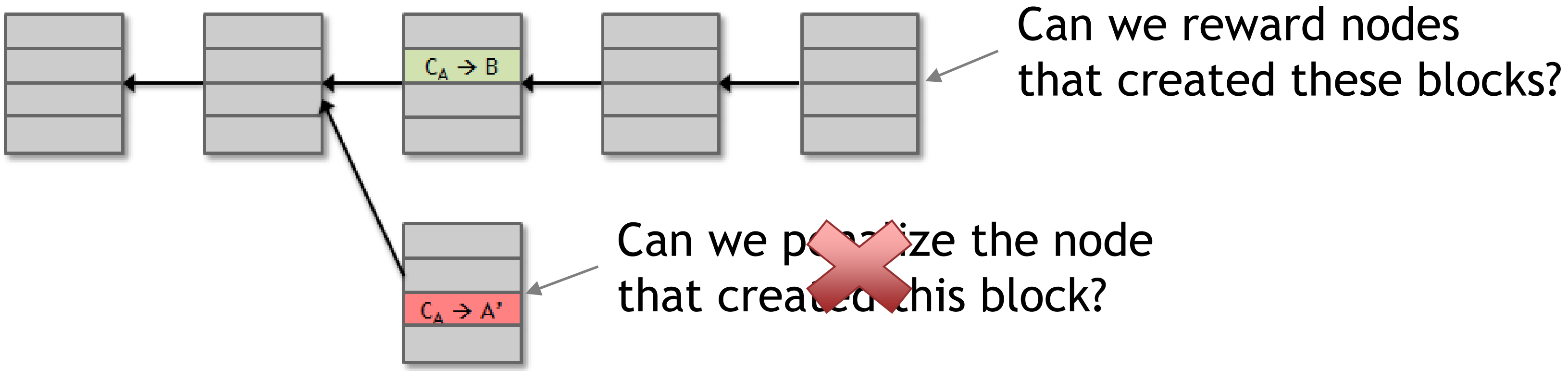
Denial of Service

Protection against double-spending is purely by consensus

You’re never 100% sure a transaction is in consensus branch. Guarantee is probabilistic

Assumption of honesty is problematic

Can we give nodes incentives for behaving honestly?



Everything so far is just a distributed consensus protocol
But now we utilize the fact that the currency has value

Incentive 1: block reward

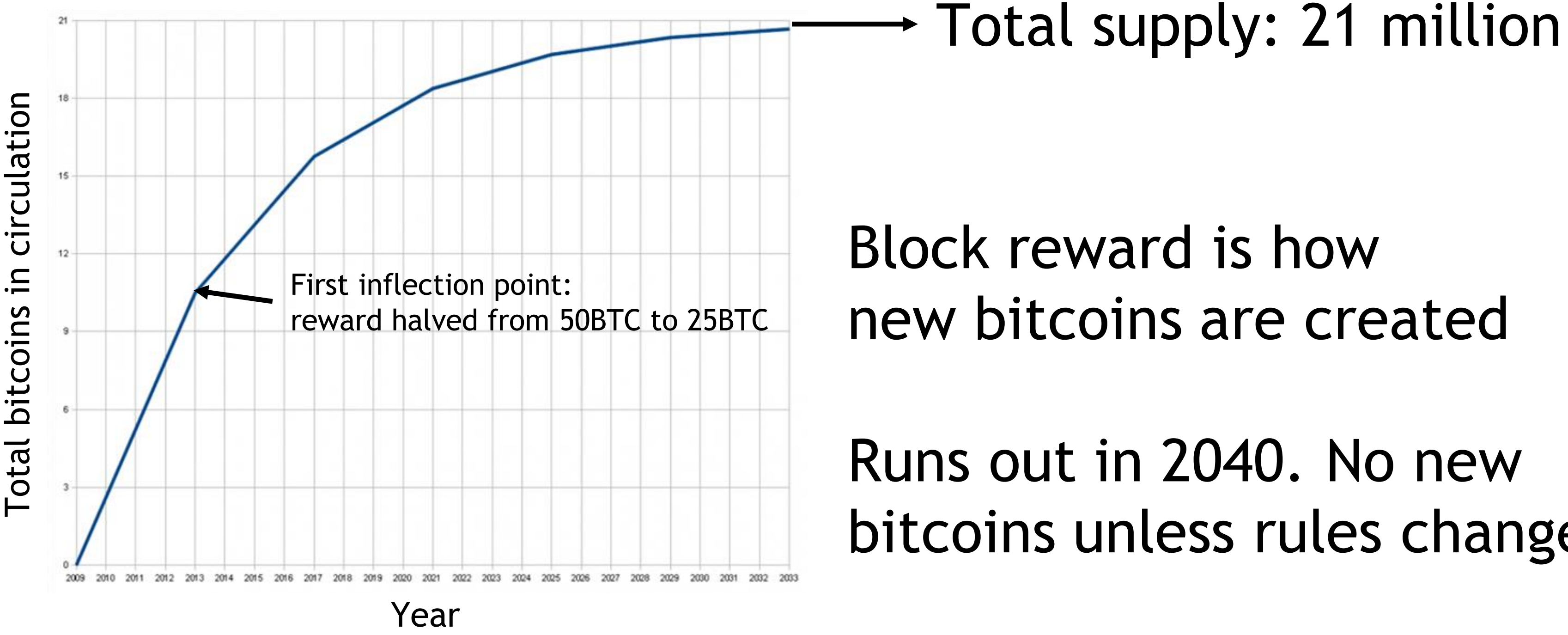
Creator of block gets to

- include special coin-creation transaction in the block
- choose recipient address of this transaction

Value is fixed: currently 12.5 BTC, halves every 4 years

Block creator gets to “collect” the reward only if the block ends up on long-term consensus branch!

There's a finite supply of bitcoins



Block reward is how new bitcoins are created

Runs out in 2040. No new bitcoins unless rules change

Incentive 2: transaction fees

Creator of transaction can choose to make output value less than input value

Remainder is a transaction fee and goes to block creator

Purely voluntary, like a tip

The real deal: coinbase transaction

```
"in":[
 {
  "prev_out":{
   "hash":"000000.....0000000",
   "n":4294967295
  },
"coinbase":"..."
 },
 "out":[
```

block reward

transaction fees

```
{
"value":"25.03371419",
"scriptPubKey":"OPDUP OPHASH160 ... ”
}
```

Mining economics

If mining reward (block reward + Tx fees)	>	hardware + electricity cost	→	Profit

Complications:

- fixed vs. variable costs
- reward depends on global hash rate

Putting it all together

Recap

Identities

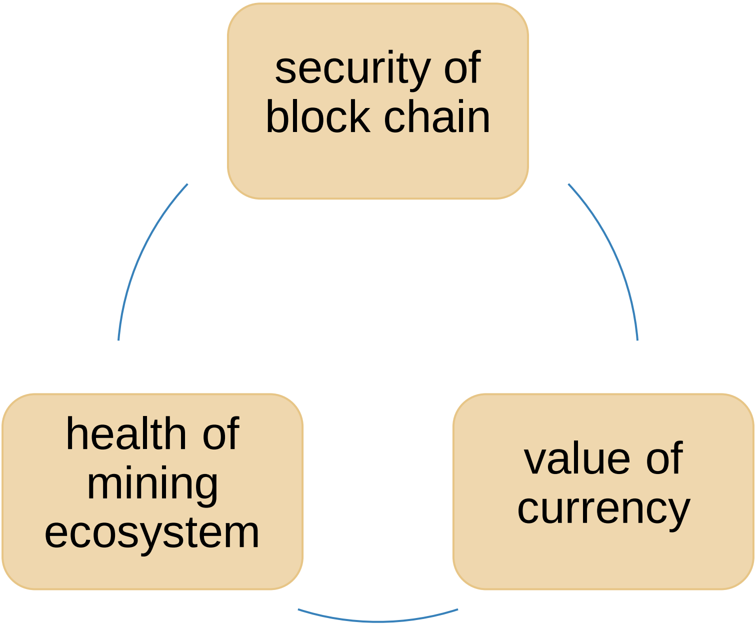
Transactions

P2P network

Block chain & consensus

Hash puzzles & mining

Bitcoin is bootstrapped



What can a “51% attacker” do?

Steal coins from existing address? ✗

Suppress some transactions?

- From the block chain ✓
- From the P2P network ✗

Change the block reward? ✗

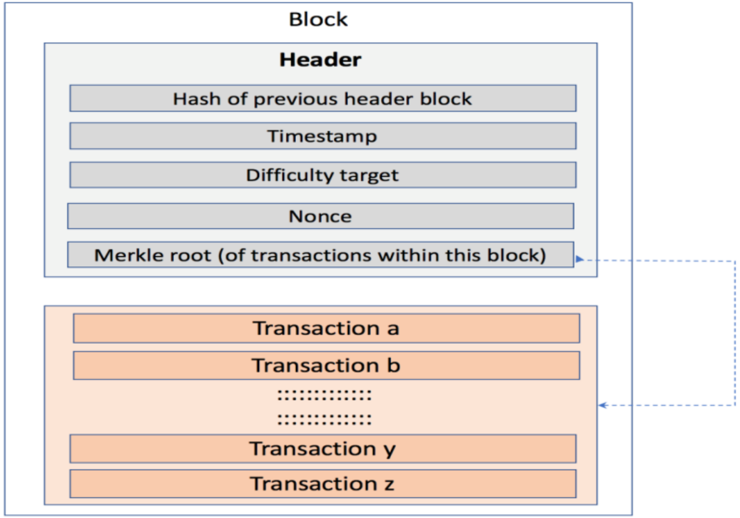
Destroy confidence in Bitcoin? ✓✓

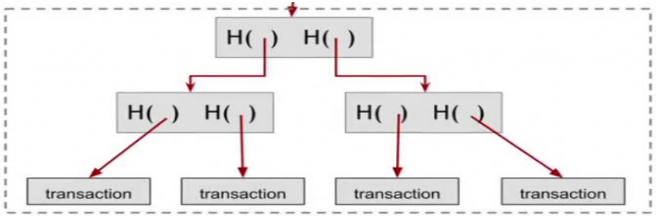
The real deal: a Bitcoin block

block header

transaction data

```
{
  "hash":"0000000000000001aad2...",
  "ver":2,
  "prev_block":"00000000000000003043...",
  "time":1391279636,
  "bits":419558700,
  "nonce":459459841,
  "mrkl_root":"89776...",
  "n_tx":354,
  "size":181520,
  "tx":[
    ...
  ],
  "mrkl_tree":[
    "6bd5eb25...",
    ...
    "89776cdb..."
  ]
}
```





Hash tree (Merkle tree)

Mining Difficulty (Max Target)

Bits: 486604799 = 1D00FFFF

target = coefficient * 2^(8 * (exponent—3))

```
target = 00FFFF * 2^(8 * (1D – 3))

target = 00FFFF * 2 ^ (8*1A)

target = 00FFFF * 2 ^ D0
```

Block #0

Summary	
Number Of Transactions	1
Output Total	50 BTC
Estimated Transaction Volume	0 BTC
Transaction Fees	0 BTC
Height	0 (Main Chain)
Timestamp	2009-01-03 18:15:05
Difficulty	1
Bits	486604799
Size	0.285 kB
Weight	0.896 kWU
Version	1
Nonce	2083236893
Block Reward	50 BTC

```
0x00ffff * 2**(8*(0x1d - 3)) = 0x00000000FFFF0000000000000000000000000000000000000000000000000000
```

Mining difficulty “target”

(as of March 2015)

256 bit hash output

0000000000000000172EC0000000000000000000000000000000000000000000

64+ leading zeroes required

Huge Number!
Greater than the Human Population of earth squared

Less than 1 in about 2^{64} nonecs that you try will work

Mining Pseudocode

More Info: https://en.bitcoin.it/wiki/Difficulty

```
TARGET = (65535 << 208) / DIFFICULTY;
coinbase_nonce = 0;
while (1) {
      header = makeBlockHeader(transactions, coinbase_nonce);
      for (header_nonce = 0; header_nonce < (1 << 32); header_nonce++){
            if (SHA256(SHA256(makeBlock(header, header_nonce))) <
      TARGET)
                  break; //block found!
      }
      coinbase_nonce++;
}
```

Max Target: An application-defined constant, which sets the target hash corresponding to the lowest possible difficulty, 1

CPU mining

```
while (1){
    HDR[kNoncePos]++;
    IF (SHA256(SHA256(HDR)) < (65535 << 208)/ DIFFICULTY)
        return;
}
```

two hashes

Throughput on a high-end PC = 10-**20 MHz ≈ 2**^{24} Hashes/Sec

139,461 years to find a block today!

Global Hash Rate & Difficulty

```
while (1){
    HDR[kNoncePos]++;
    IF (SHA256(SHA256(HDR)) < (65535 << 208)/ DIFFICULTY)
        return;
}
```

- **Hashes are 256-bit integers. So, TOTAL output size: 2^256.**
- **The current TARGET is "max_target/difficulty", where max_target is (65535 * 2^208).**
- **Therefore, fraction of output space is is TARGET/TOTAL. Therefore, TOTAL/TARGET=(2^256*difficulty/max_target) no. of hashes are needed on average to find a block.**
- **This is done over 600 sec, considering previous 2016 blocks.**
- **Global Hashrate:**

  (2^256 * difficulty/max_target)/600

  = (2^256 * difficulty/65535 * 2^208)/600

  = (2^48 * difficulty/65535)/600

  = difficulty * 7158388.055

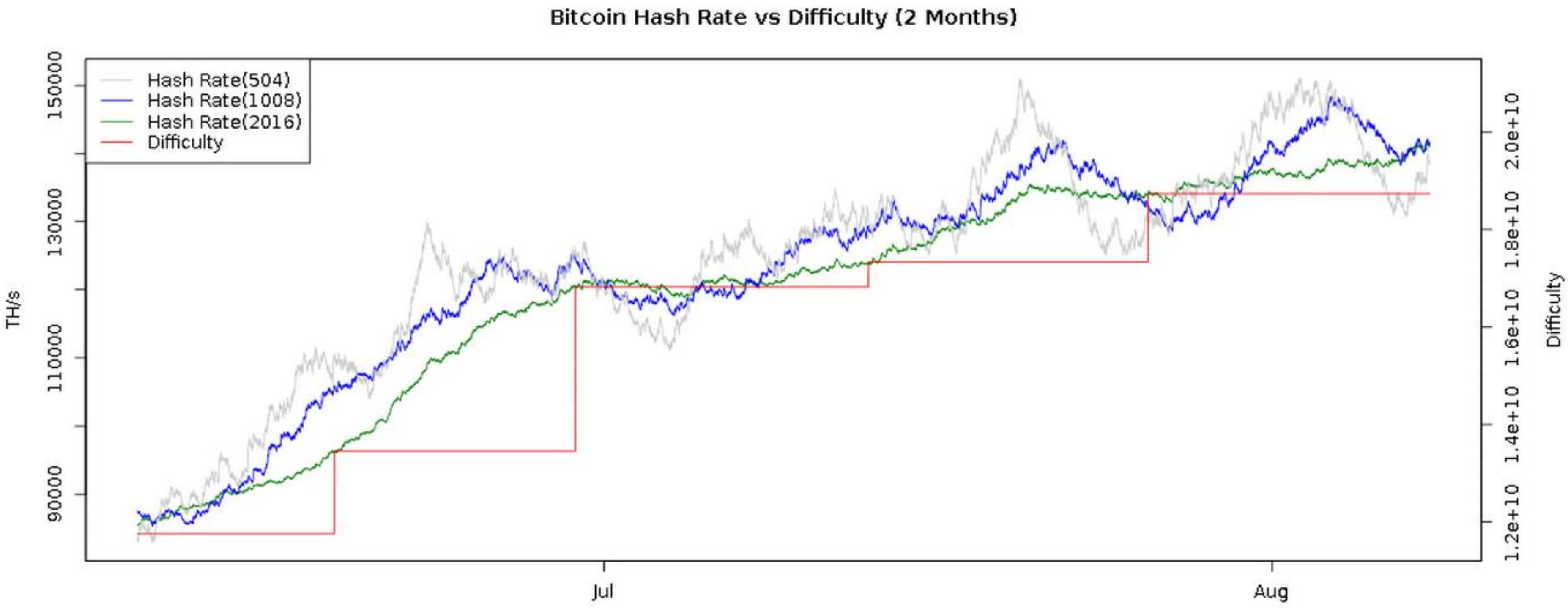
Setting the mining difficulty

Every two weeks, compute:

```
next_difficulty= currennt_difficulty *
                (2 weeks)/(time to mine last 2016 blocks)
```

Expected number of blocks in 2 weeks at 10 minutes/block

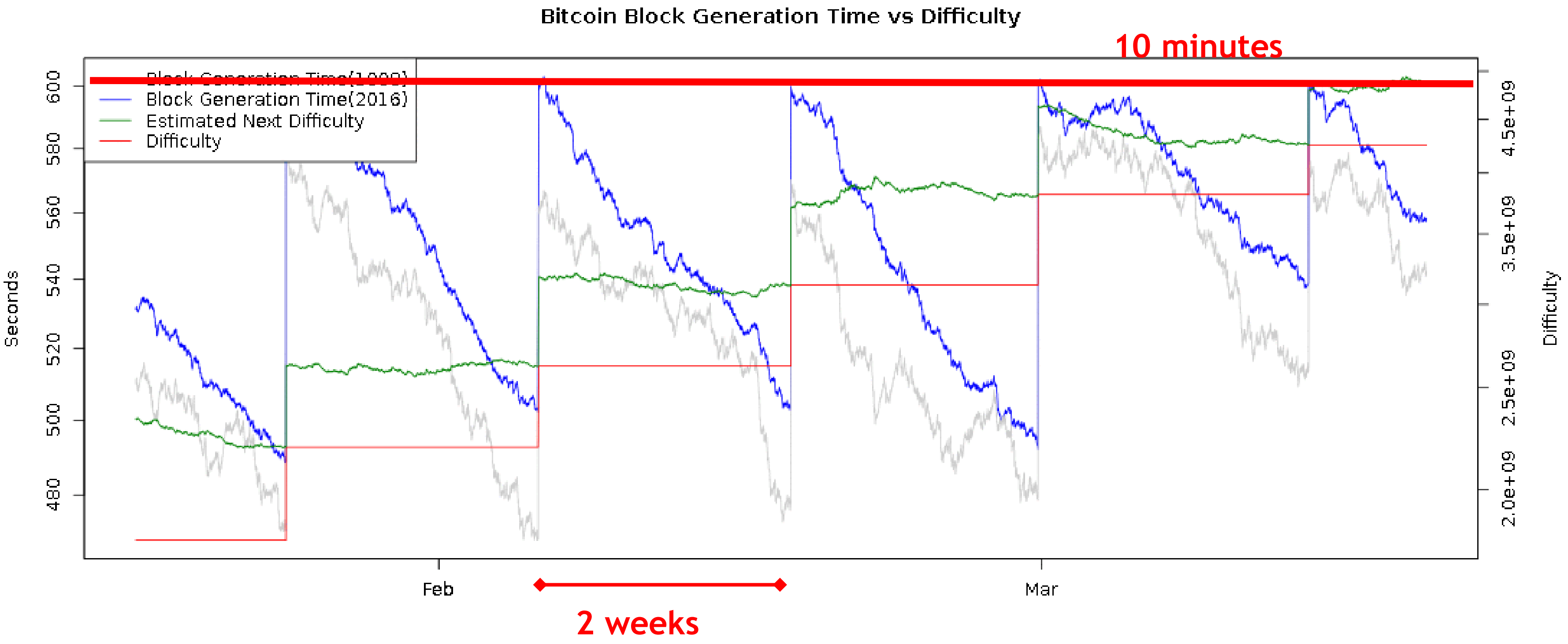
Mining difficulty over time



Note that the y-axis begins at 80,000 TeraHashes/s. The hash rate is averaged over 2016/1008/504 blocks

bitcoinwisdom.com

Time to find a block



bitcoinwisdom.com

Bitcoin Difficulty and Hashrate Cl ×

https://bitcoinwisdom.com/bitcoin/difficulty

Apps | Mail Login | Finance-1 | Finance-2 | Utility | Travel | Entertainment | Research | Book Online

Bitcoin Difficulty:	6,068,891,541,676
Estimated Next Difficulty:	5,933,343,057,673 (-2.23%)
Adjust time:	After 1529 Blocks, About 11.0 days
Hashrate(?):	40,575,440,550 GH/s
Block Generation Time(?):	1 block: 10.3 minutes 3 blocks: 31.0 minutes 6 blocks: 1.0 hours
Updated:	13:10 (2.6 hours ago)

Bitcoin Hash Rate (2 Months)

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

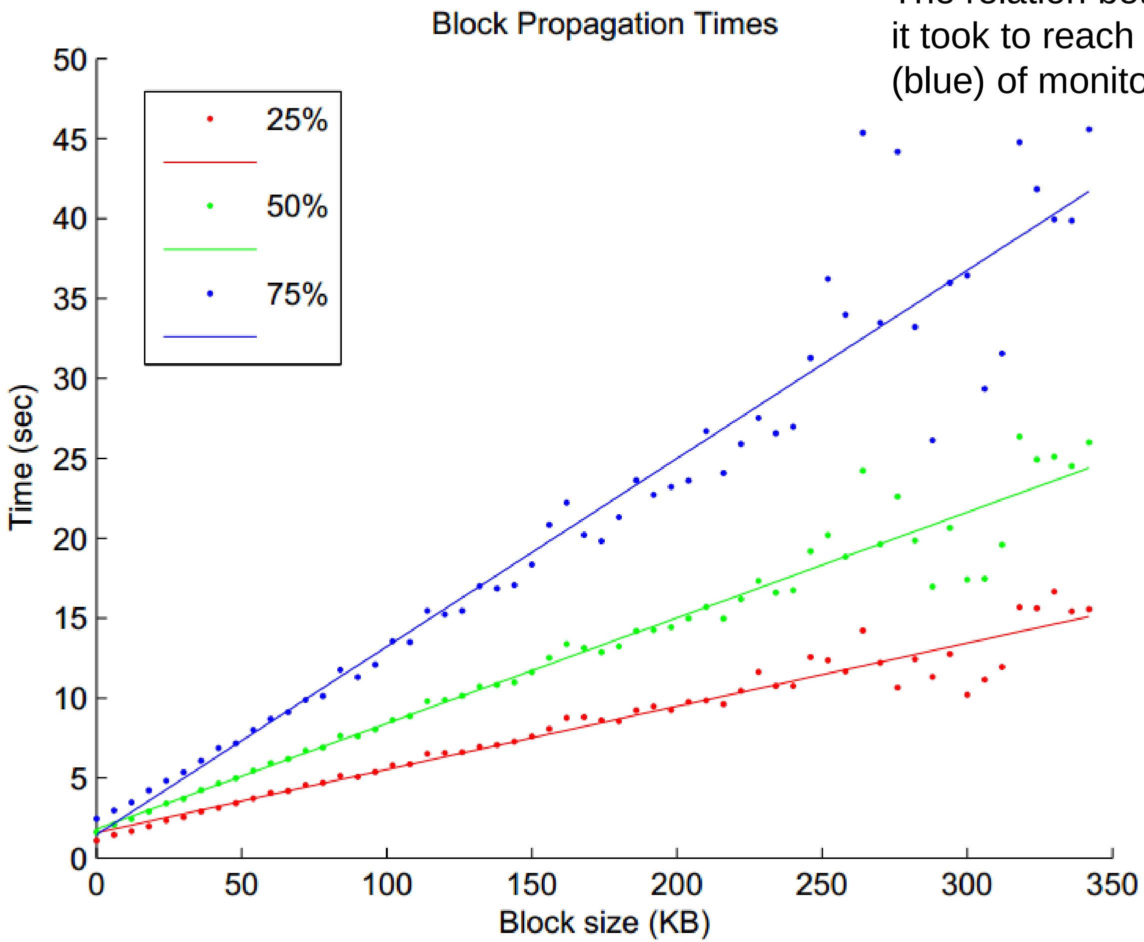
Block Information:

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

The relation between the block size and the time it took to reach 25% (red), 50% (green), and 75% (blue) of monitored nodes



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