BLOCKS AND THE BLOCKCHAIN

JOHN NEWBERY

@jfnewbery

github.com/jnewbery

ABOUT ME

Live in New York

Work for Chaincode Labs

Contribute to Bitcoin Core

github.com/jnewbery

BLOCKS AND THE BLOCKCHAIN

- Why do we need a blockchain?
- What is proof-of-work? What is mining?
- What is difficulty? How do difficulty re-adjustments happen?
- How are new Bitcoin?
- What does a block look like? What's in a block header?
- How are transactions included in a block?
- How do we agree on what the current blockchain is?
- How have blocks changed with Segregated Witness (SegWit)?

WHY DO WE NEED A BLOCKCHAIN?

THE DOUBLE SPEND PROBLEM

- Bitcoin transactions are self-validating
- Everyone can verify that a Bitcoin transaction is valid
- Alice pays Bob by:
  - using some of her unspent coins
  - signing with her private key
- Alice can create a second transaction paying Carol with *the same* unspent coins. That's also a valid transaction!
- This is called the 'double spend' problem

THE DOUBLE SPEND PROBLEM (PART 2)

- If Alice has the private keys for her unspent coins, she can sign as many times as she wants
- If there's no way to know which coins have already been spent, there is no way to prevent double spends
- We need a way for everyone to agree which coins have been spent already
- We need to agree on the *ordering* of transactions

THE DOUBLE SPEND PROBLEM (PART 3)

- Ordering transactions is easy in a centralized system: trust a third party to do it!
- Banks, credit card companies, etc are third parties
- Nobody knew how to create a shared ledger without a trusted third party until…

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

Commerce on the Internet has come to rely almost exclusively on financial institutions serving as

"...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..."

A SOLUTION TO THE DOUBLE SPEND PROBLEM!

- Distribute the ledger amongst everyone on the network
- Nodes take it in turn to add a new ‘page’ to the ledger.
- In Bitcoin we call this page of transactions a *block*
- Who gets to add the next block is determined by a hash-based proof-of-work contest.
- This is described in the whitepaper as ‘one-CPU-one-vote’

MAKING A CHAIN OF BLOCKS

- The proof-of-work over the blocks *commits* the block to the transactions and to the previous block
- A block can’t be changed without redoing the work of that block
- A buried block can’t be changed without redoing the combined work for that block and all the blocks after it

PROOF OF WORK AND MINING

PROOF-OF-WORK

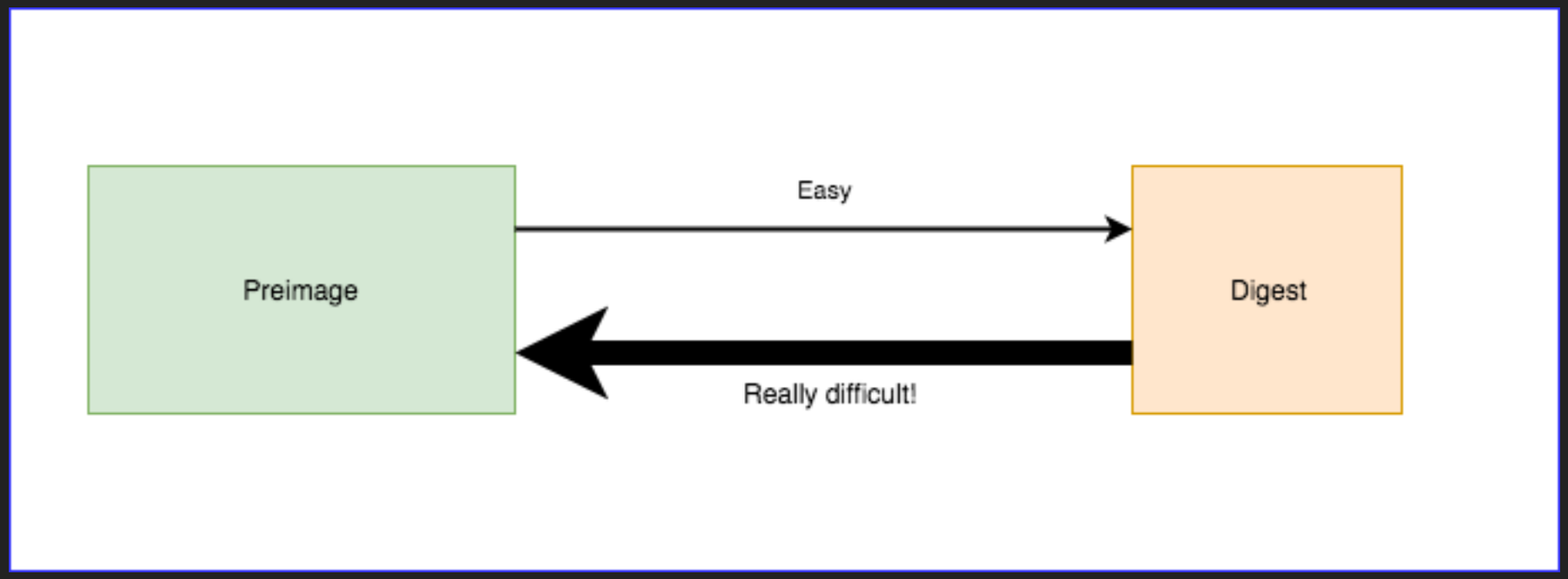
- Satoshi's solution to the double spend problem
- Based on Adam Back's *hashcash* and other earlier proof-of-work schemes
- Requires the miner to do computational work in order to *discover* a new block

CRYPTOGRAPHIC HASH FUNCTIONS

- A hash function is a function that takes an arbitrary-length input message and outputs a fixed-length *digest*
- A cryptographic hash function has additional properties:
  - it is infeasible to generate a message from its hash value (preimage resistance)
  - a small change to a message results in a completely different digest (avalanche effect)
  - it is infeasible to find two different messages with the same hash value (collision resistance)
- A cryptographic hash function is a one-way function. To an observer, the outputs of the hash function look like random numbers

CRYPTOGRAPHIC HASH FUNCTIONS

- A cryptographic hash function is a one-way function. To an observer, the outputs of the hash function look like random numbers



- Try it now: find the digest of “devplusplus”

SHA 256

- SHA256 is a cryptographic hash function that maps inputs to 256 bit outputs
- Those outputs are essentially randomly distributed:
  - Half of all possible messages will hash to 0b0... and half of all possible messages will hash to 0b1...
  - One fourth of all messages will hash to 0b00...
  - One eighth of all messages will hash to 0b000...
  - ...
- In general, 1 out of 2^x messages will hash to a digest with x leading zeroes

PROOF-OF-WORK OVER A MESSAGE (1)

- To do proof-of-work *over* a message:

    1. Append some random bits to the end of the message. We call those bits a *nonce* (a **n**umber used **once**). For now, let's call <message|nonce> a *block*
    2. Hash the block using SHA256
    3. If the digest starts with the *target* number of zeroes, the block is valid. If not, the block is invalid - go to (1) and try with a different nonce

PROOF-OF-WORK OVER A MESSAGE (2)

- If the difficulty target is 4 zeroes, then *on average* we'll need to try 16 different nonces to find a valid block
- An observer only needs to do one hash to verify that the block is valid
- Try it now:
  - Find a valid block for the message "devplusplus" with 4 bits of difficulty
  - Validate your neighbor's block

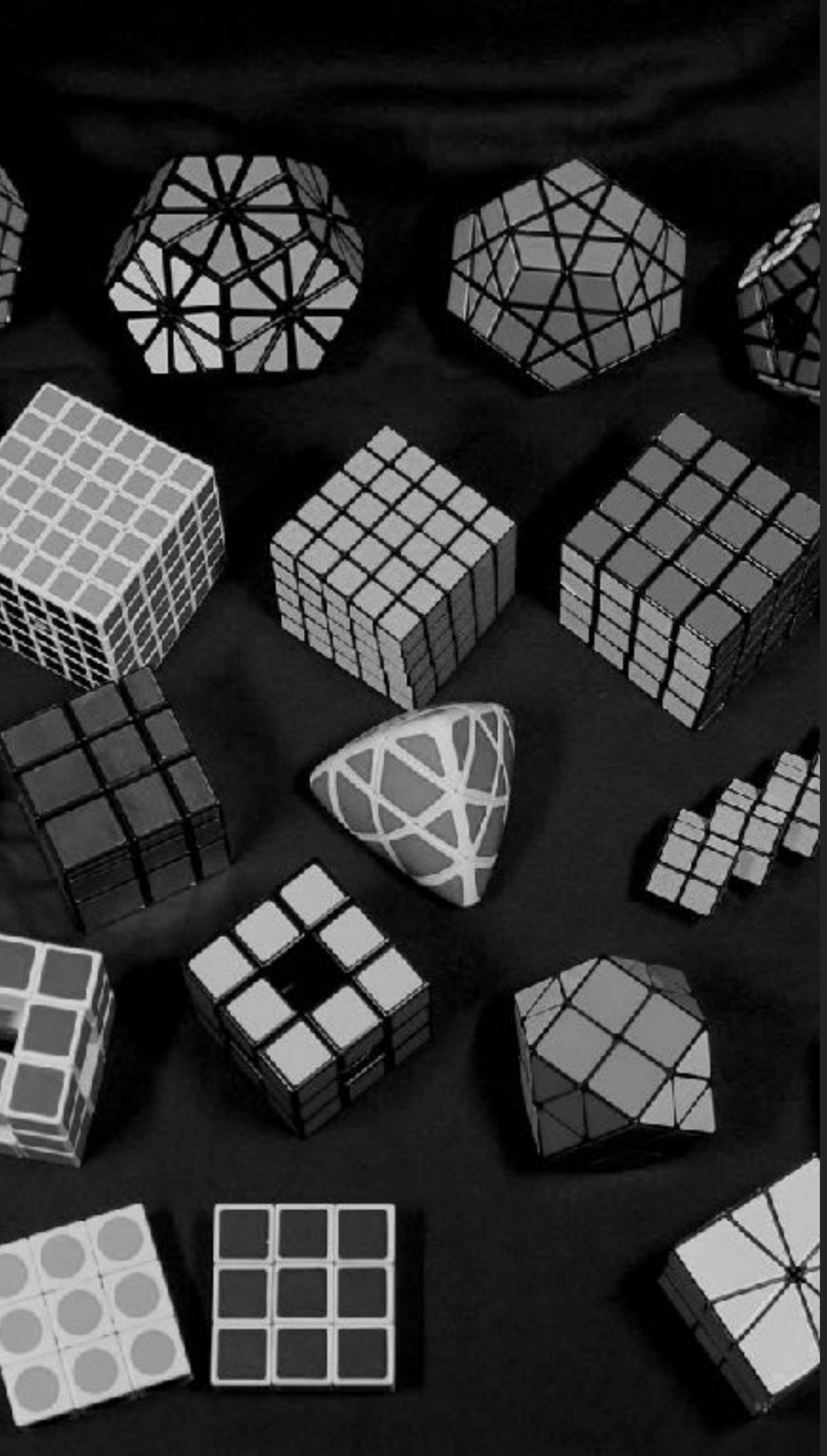
BITCOIN MINING

- Bitcoin mining uses the exact mechanism. Miners try lots of different nonces until they discover a valid block
- Miners do work over the Bitcoin block header. The nonce is the final 4 bytes of that header*
- The current difficulty on the bitcoin network requires ~70 leading zeroes

** note that this isn't enough nonce space, so they use part of the coinbase transaction for additional entropy.*

MINING AND THE BLOCKCHAIN

- The block header includes the hash of the previous block
- By mining a new block, the miner is doing work *over the entire chain*
- Mining is a race to extend the chain. When a miner discovers a block, he/she transmits it to the network and other miners start trying to build a block on top of it



DIFFICULTY

WHY DOES DIFFICULTY CHANGE?

- Satoshi designed the Bitcoin system to produce blocks on average every ten minutes
- As more miners start mining Bitcoin and technology advances, the *network hash rate* increases
- If difficulty remained the same, blocks would be discovered more and more quickly
- At today's network hash rate, blocks of difficulty 1 would be discovered every 0.0000000004 seconds

HOW IS DIFFICULTY VALIDATED?

- The block header contains a 4-byte *difficulty bits* field.
- The double-sha256 hash of the header is checked against the difficulty bits.
  - This is a *non-contextual* check
- This difficulty bits field is checked against the blockchain timestamps
  - This is a *contextual* check

DIFFICULTY BITS EXPLAINED

- The difficulty bits field is 4-bytes, little-endian. e.g. **0xe93c0118**
- The first byte **0x18** (24 in decimal) is the exponent
- The next three bytes **0x013ce9** is the coefficient
- The target is given by the formula: coefficient x $2^{(8 \times (\text{exponent} - 3))}$
- For our example, the target is:
  0x013ce9 x $2^{(8 \times (24 - 3))}$ = 0x00000000000000013ce9000000000000000000000000000000000000
- What is the current target on the Bitcoin mainnet?

DIFFICULTY (1)

- Block explorers sometimes express difficulty as a multiple of the lowest possible difficulty, e.g.:

Block #491529

Summary	
Number Of Transactions	677
Output Total	4,711.88379637 BTC
Estimated Transaction Volume	1,153.66097689 BTC
Transaction Fees	1.44170578 BTC
Height	491529 (Main Chain)
Timestamp	2017-10-24 19:55:09
Received Time	2017-10-24 19:55:09
Relayed By	Unknown
Difficulty	1,196,792,694,098.79

DIFFICULTY (2)

- The lowest allowed difficulty corresponds to difficulty bits **0xffff001d**, which corresponds to target:

  65535 x $2^{(8 \times (29-3))}$ = 0x00000000ffff0000000000000000000000000000000000000000000000000000

- Divide the lowest allowed difficulty by the block's "difficulty" to get the target:

  65535 x $2^{(8 \times (29-3))}$ / 888,171,856,257.3 ~=
  0x00000000000000013ce9000000000000000000000000000000000000000000

- Note that 'Difficulty' is rounded so this won't give you the exact target.

- What is the current difficulty on the Bitcoin main net?

HOW DOES DIFFICULTY CHANGE? (1)

- To keep blocks at ten minute intervals, the Bitcoin network *retargets* its difficulty every 2016 blocks
- 2016 blocks should take 20160 minutes
  - If the previous 2016 blocks took longer than 20160 minutes, make the target easier
  - If the previous 2016 blocks took shorter than 20160 minutes, make the target harder

HOW DOES DIFFICULTY CHANGE? (2)

- Retargeting done automatically by the Bitcoin network.
- The timestamps are taken from block 0 and block 2015 in the previous retarget window:
  - There's an off-by-one bug! *Why don't we fix that bug?*
  - The miner who discovers block 2015 has the chance to slightly change the difficulty of the next window.
- Try it now: calculate the difficulty for block 491904

HOW DOES DIFFICULTY CHANGE? (3)

- The difficulty adjustment algorithm was set in place by Satoshi
- There's a maximum difficulty change of ±4x for each retarget
- The algorithm isn't tolerant to large changes in network hash rate. For example, if network hash rate drops by 90%:
  - Blocks will be discovered every 100 minutes
  - It will take 20 weeks to reach the next retarget
  - At the next retarget, difficulty will drop to ¼ , so blocks will be discovered every 25 minutes
- Attempts to 'fix' this in other coins have often caused their own problems (eg Bitcoin Cash's Emergency Difficulty Adjustment)

HOW ARE NEW BITCOINS CREATED?

WHY DO MINERS MINE?

- Mining is very expensive:
  - Mining equipment (ASICs) cost thousands of dollars
  - Mining requires a lot of electricity (and cooling)
  - Labor costs can be high
- So why do miners mine?

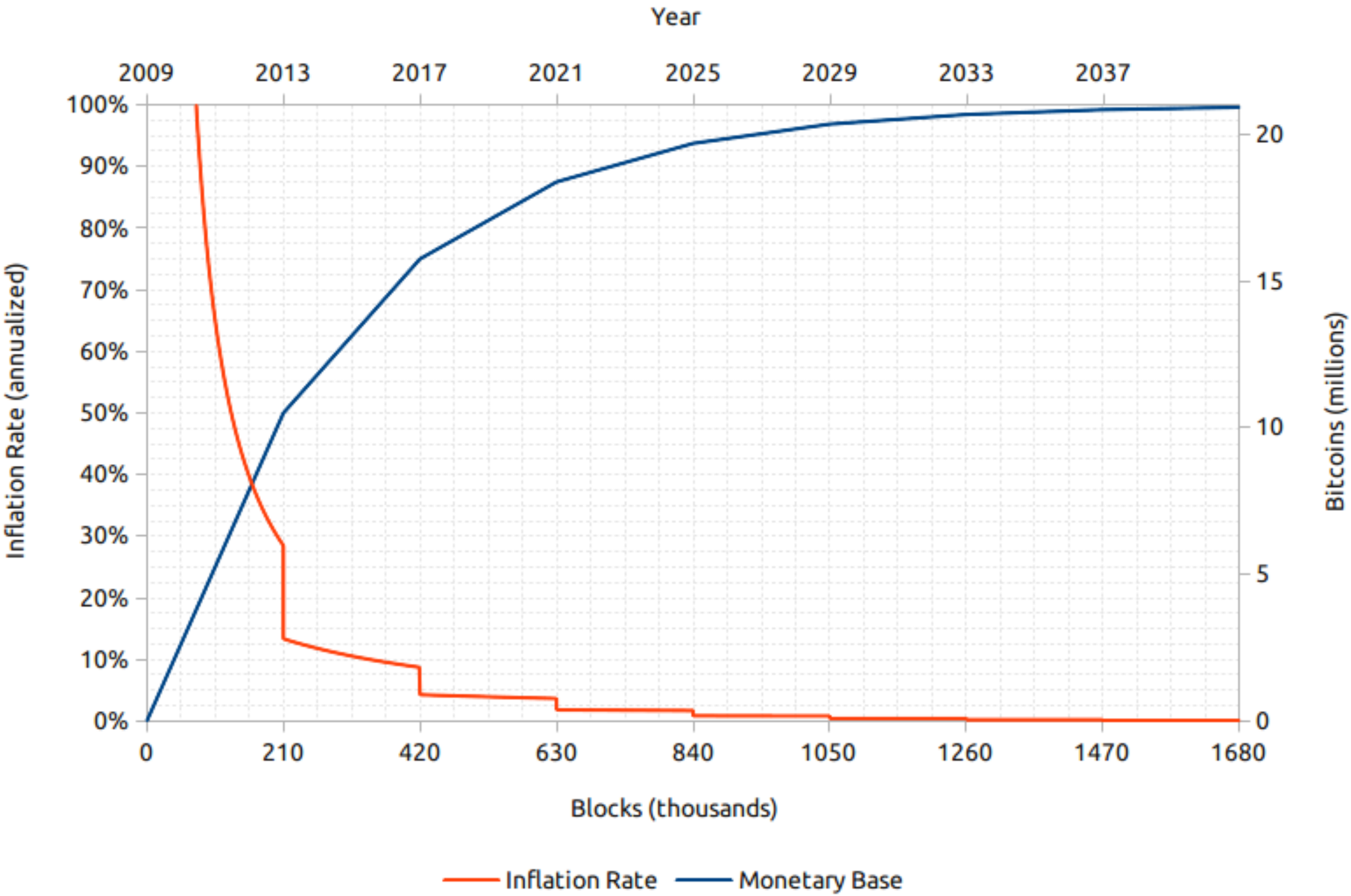
Hint: it's not from the benevolence of the miner that we expect our blocks …

. . . BUT FROM THEIR REGARD TO THEIR OWN INTEREST*

- Miners are rewarded for their efforts
- Each block contains a *reward* or *block subsidy*, which is paid out to the miner in the *coinbase transaction*.
- Initially, the reward was 50 Bitcoin per block.
- After 210,000 blocks (approx 4 years), the reward was halved** to 25 Bitcoin per block.
- At height 420,000 the reward was halved again to 12.5 Bitcoin per block.
- After 33 halvings (6,930,000 blocks) the reward is zero.
- What is the total supply of Bitcoin?

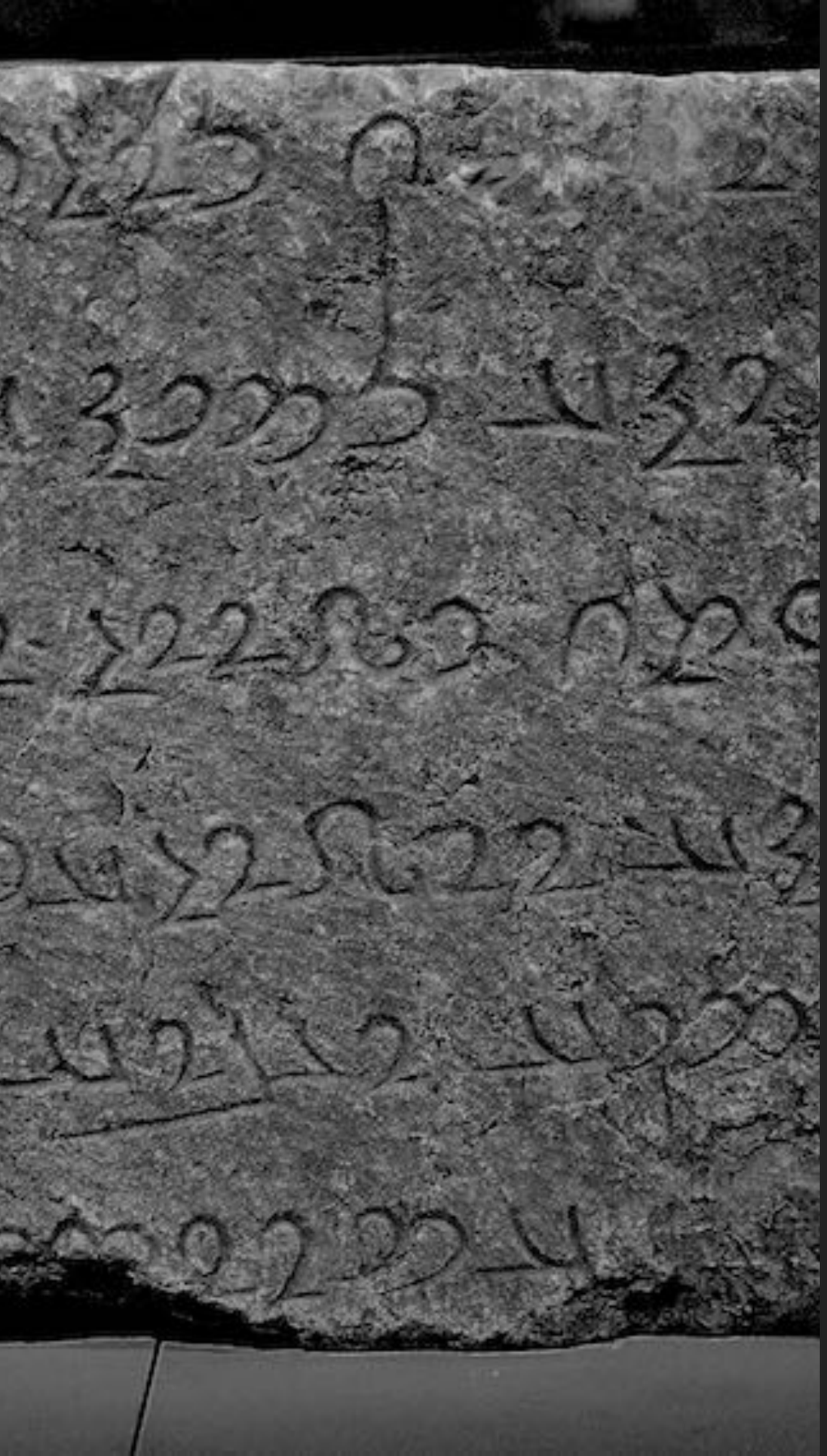
** with apologies to Adam Smith*
*** unofficially called a 'halvening'*

Bitcoin Inflation vs. Time



WHAT HAPPENS WHEN THE BITCOIN RUN OUT?

- Miners are also rewarded with the *transaction fees* of all the transactions they include in their block
- Miners choose transactions to include in a block based on their *fee rate* (amount of fee divided by transaction size)
- When blocks aren't full, these fees are very low, since there isn't competition for block space
- As demand for block space increases, fees rise
- What happens to the system as the subsidy decreases is an open question!



BLOCK STRUCTURE

BLOCKS AND BLOCK HEADERS

- Blocks consist of:
  - *A block header*: information about the block (80 bytes)
  - *The transactions*: the serialized transactions in the block (up to 1MB before SegWit, up to 4MB after SegWit)

BLOCKS HEADERS

- Blocks headers contain:
  - The block version (4 bytes)
  - The hash of the previous block (32 bytes)
  - The Merkle root of the transactions in the block (32 bytes)
  - The difficulty bits (4 bytes)
  - The timestamp of the block (4 bytes)
  - The nonce (4 bytes)

THE COINBASE TRANSACTION

THE COINBASE TRANSACTION

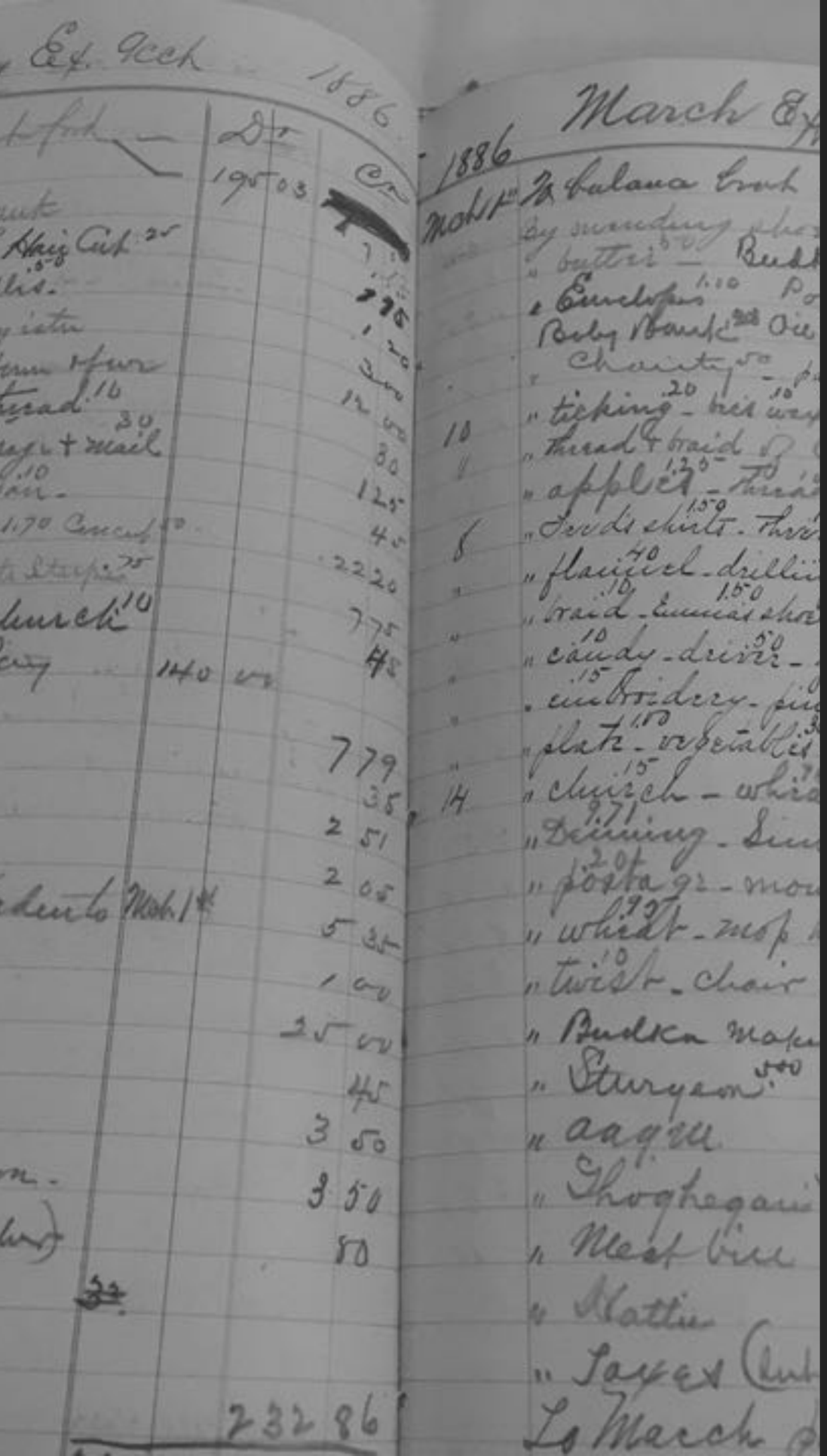
- The first transaction in every block is the *coinbase* transaction
- A block must have exactly one coinbase transaction
- The coinbase transaction spends no UTXOs
- The coinbase transaction produces new coins
- The coinbase transaction also collects the transaction fees for all the transactions in the block

THE COINBASE TXIN

- A coinbase transaction has only one input
- The outpoint is null (prev transaction is 0, index is -1)
- The scriptSig starts with the block height (BIP34) and can be up to 100 bytes.

THE COINBASE TXOUTS

- A coinbase can have many outputs
- The transaction outputs add up to the block reward plus the sum of all transaction fees in the block
- Since SegWit activation, one of the outputs must *commit* to the witness root (full details later)



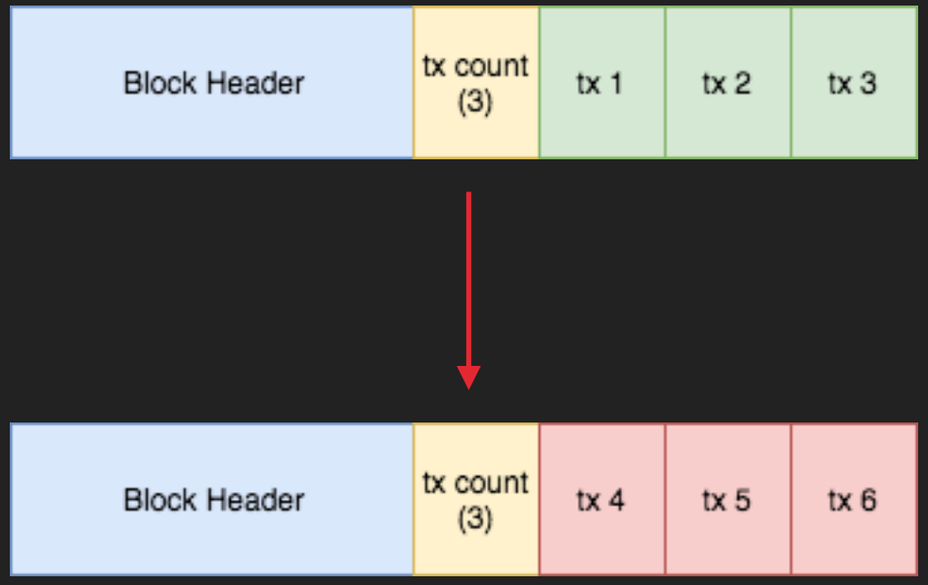
HOW ARE TRANSACTIONS INCLUDED IN A BLOCK?

SERIALIZED BLOCK (1)

- Blocks are transmitted over the P2P network in **MSG_BLOCK** or **MSG_WITNESS_BLOCK** messages.
- The message contains a ‘serialized block’:
  - block header (80 bytes)
  - transaction count (compact size int)
  - serialized transactions

SERIALIZED BLOCK (2)

- Question 1: what stops a malicious node from changing the transaction in the block?

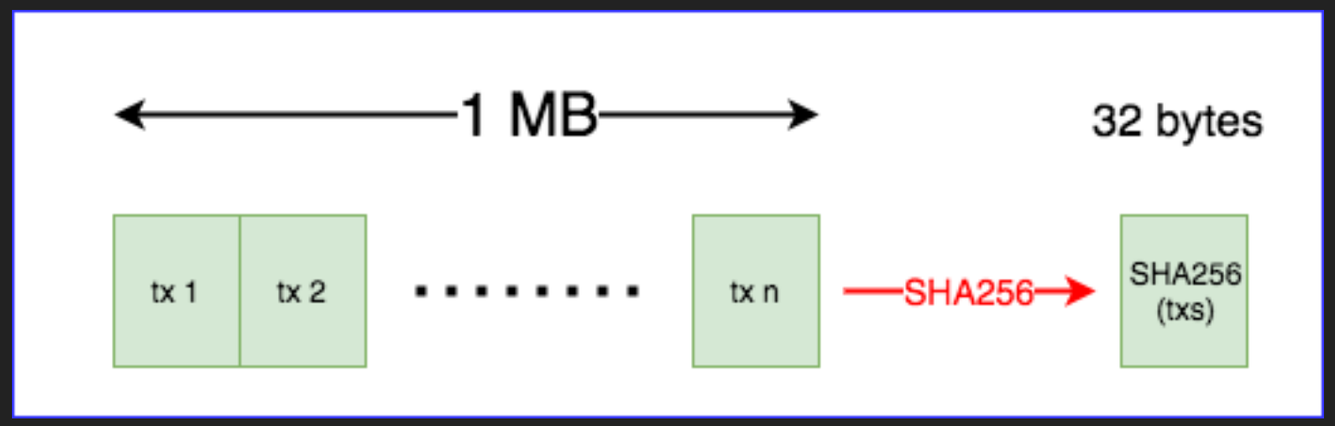


TRANSACTION COMMITMENT (2)

- Answer: The block header *commits* to the transactions
- It's not possible to change the transactions in the block without changing the block header
- Changing just one bit of the block header requires redoing the proof-of-work
- Question 2: how do we fit 1MB of transactions into a 80 byte header?

TRANSACTION COMMITMENT (2)

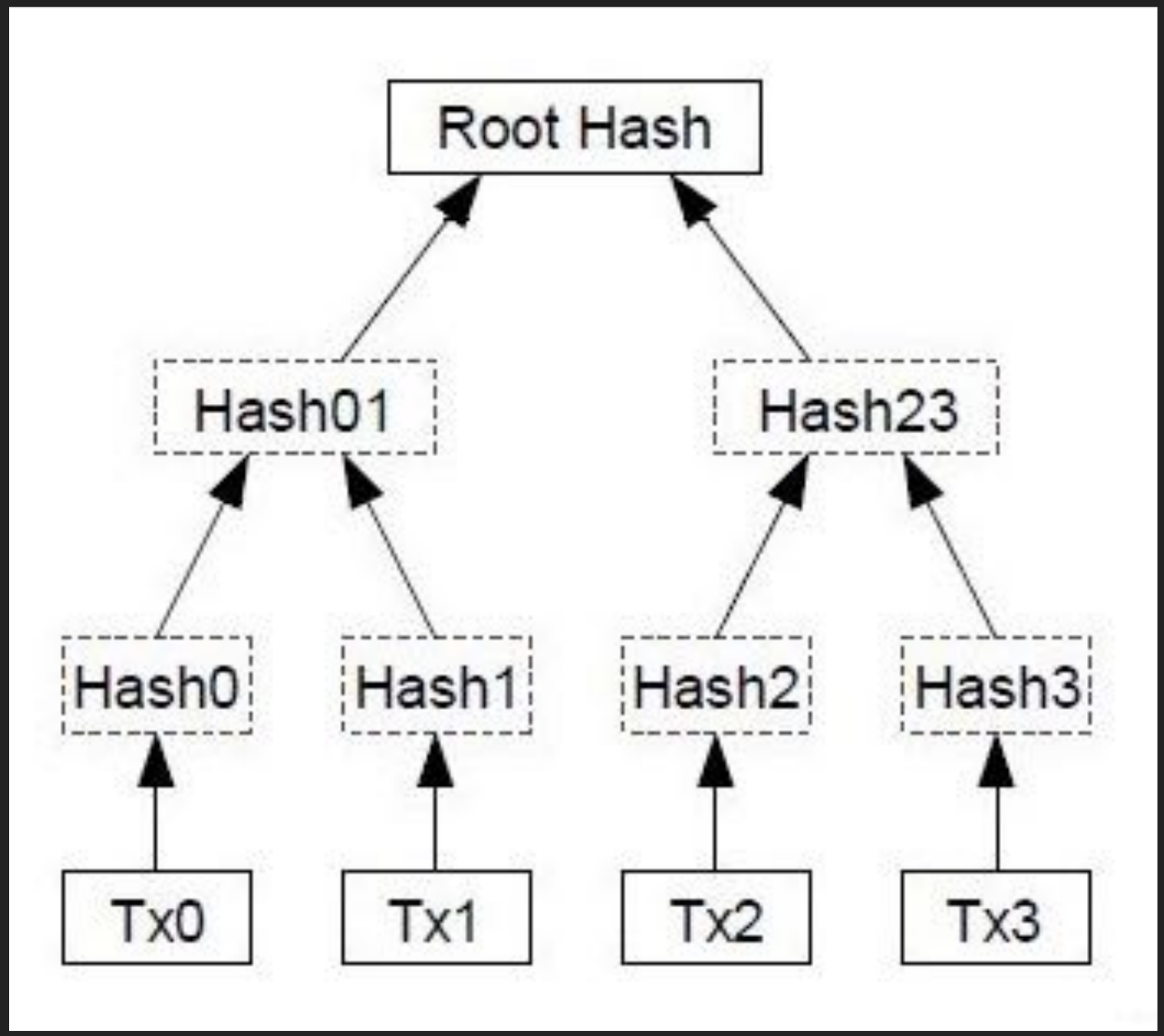
- Answer: the transactions are hashed into a 32 byte digest.
- One commitment scheme would be to put the entire 1MB of serialized transactions through a round of SHA256



TRANSACTION MERKLE ROOT

- A more useful commitment scheme is to use a *Merkle tree*
- Each tx is individually hashed to produce a digest
- The digests are paired and hashed again to produce a second level digest
- This step is repeated until only one digest remains, called the Merkle root
- The Merkle root identifies the whole set in just a 32 bytes

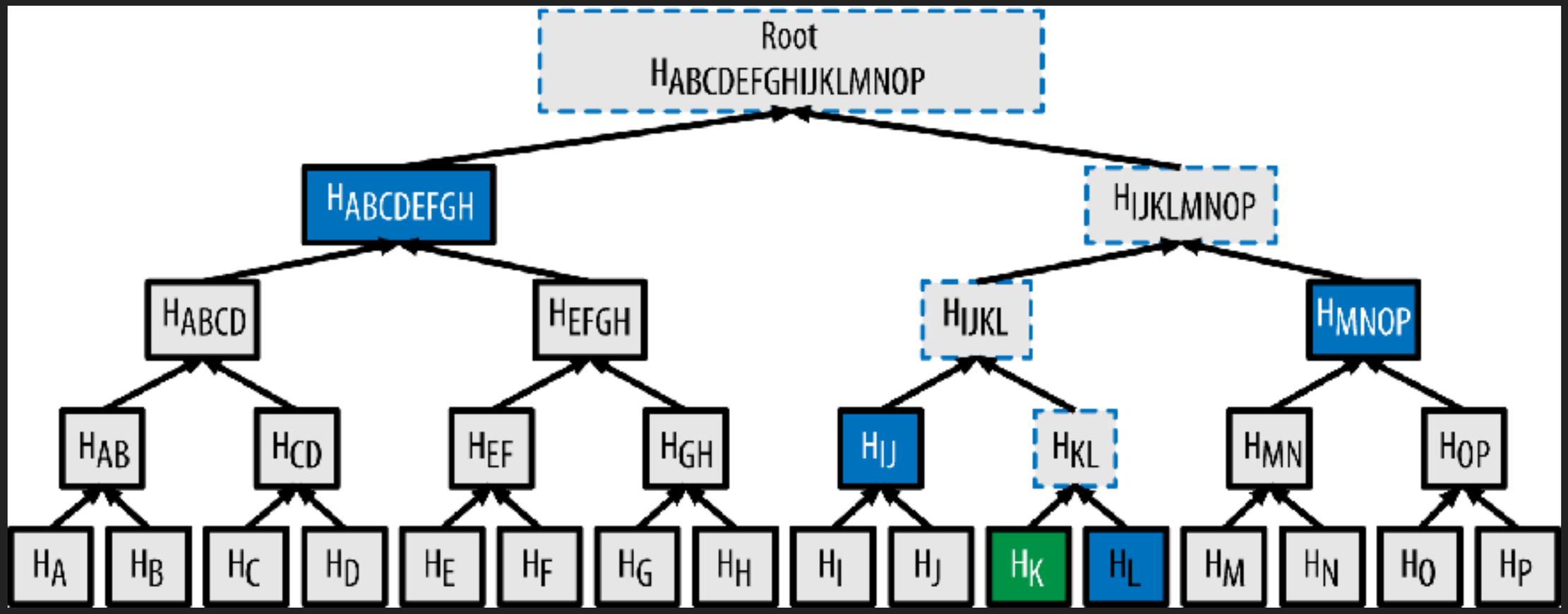
MERKLE TREE


Root Hash
Hash01
Hash23
Hash0
Hash1
Hash2
Hash3
Tx0
Tx1
Tx2
Tx3

PROOFS OF INCLUSION

- A Merkle tree permits a compact proof-of-inclusion
- A prover can show that a transaction is included in the block with a *Merkle proof*
- The Merkle proof contains the path through the Merkle tree to the transaction
- Note that the transaction Merkle root does not permit a *proof-of-exclusion*

MERKLE PROOF

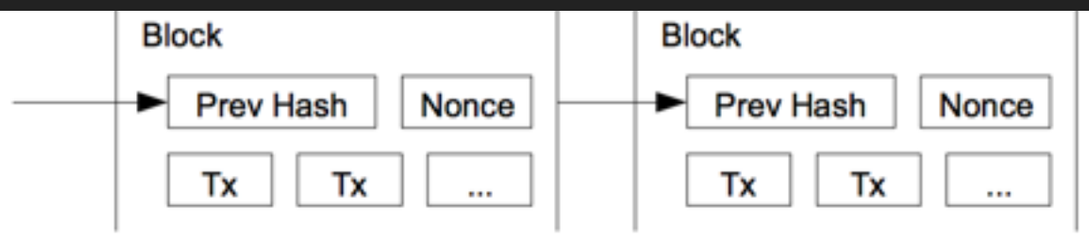


SIMPLE PAYMENT VERIFICATION

- *Simple Payment Verification* (SPV) wallets make use of Merkle proofs
- The SPV wallet asks a full node for proof that a transaction is in the blockchain
- A prover provides a headers chain and a Merkle proof to show that a transaction exists
- Since there is no proof-of-exclusion, a prover can ‘lie by omission’ and not reveal the presence of a transaction

FORKS AND RE-ORGS

HOW IS CONSENSUS REACHED?



The proof-of-work also solves the problem of determining representation in majority decision making. If the majority were based on one-IP-address-one-vote, it could be subverted by anyone able to allocate many IPs. Proof-of-work is essentially one-CPU-one-vote. The majority decision is represented by the longest chain, which has the greatest proof-of-work effort invested in it. If a majority of CPU power is controlled by honest nodes, the honest chain will grow the fastest and outpace any competing chains. To modify a past block, an attacker would have to redo the proof-of-work of the block and all blocks after it and then catch up with and surpass the

“The majority decision is represented by the longest chain, which has the greatest proof-of-work effort invested in it. If a majority of CPU power is controlled by honest nodes, the honest chain will grow the fastest and outpace any competing chains…”

Question: where’s the obvious bug?

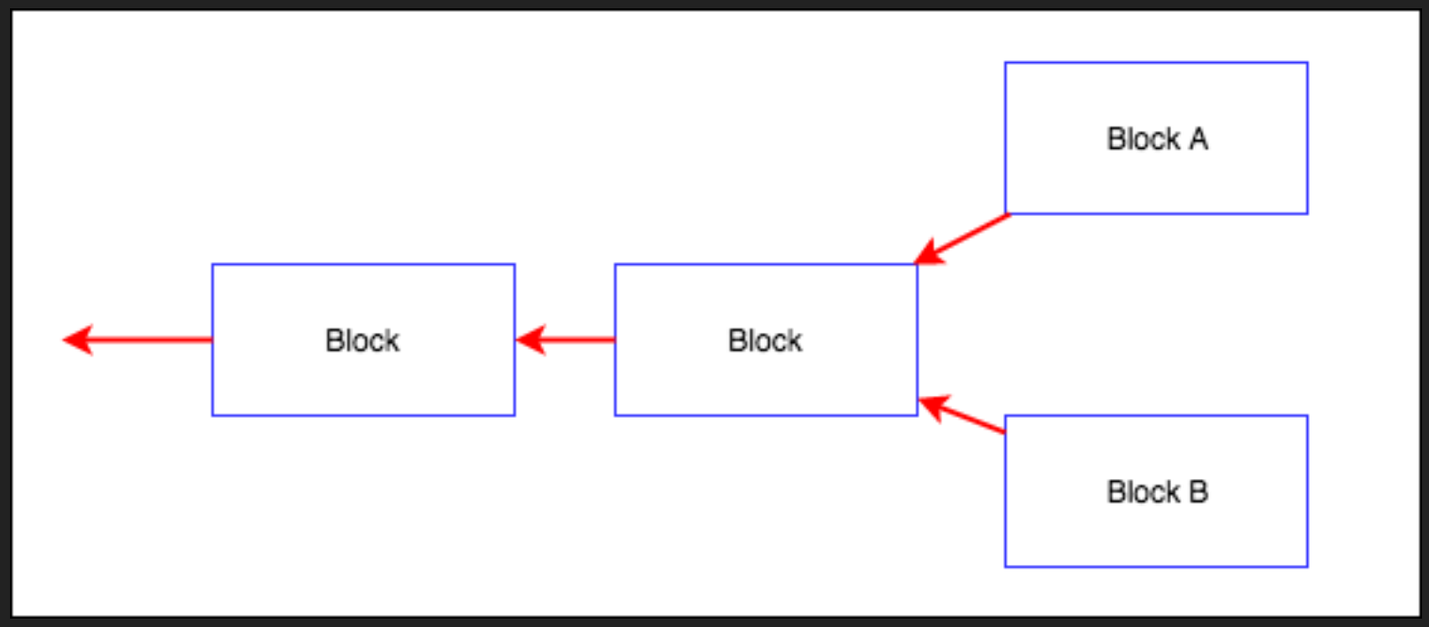
WHAT IS THE CORRECT CHAIN?

- The **valid** chain with the most accumulated work is considered the correct chain
- If there are two chains with the same accumulated work, the tie-break is which chain was seen first

CHAIN FORK (1)

- Temporary hain forks are an expected and regular occurrence on Bitcoin
- Block discovery is a random process
- Occasionally two miners will discover blocks at the same time
- Some nodes see block A first and other nodes see block B first
- One-block chain forks happen once or twice a month on the network

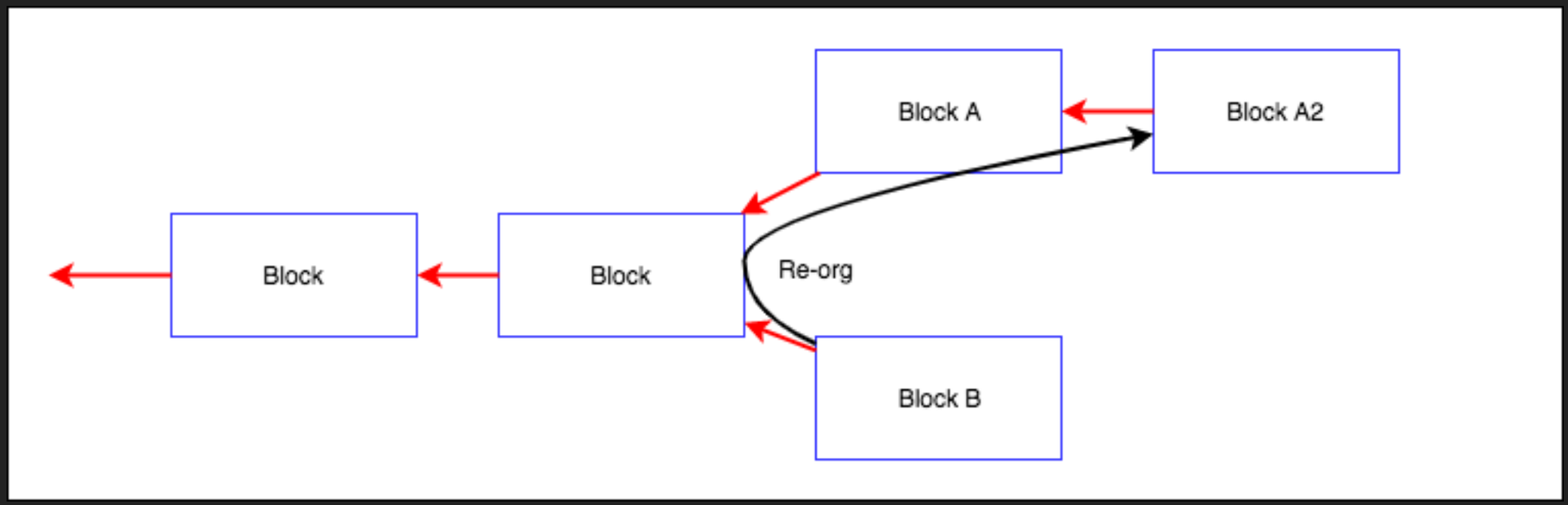
CHAIN FORK (2)



RESOLVING A CHAIN FORK (1)

- When the chain forks, miners will start building on top of one of the two tips
- Once a miner finds a block on one of the tips, that chain becomes the most-work chain
- Nodes that were on the other tip will *re-org* to the new most-work chain

RESOLVING A CHAIN FORK (2)



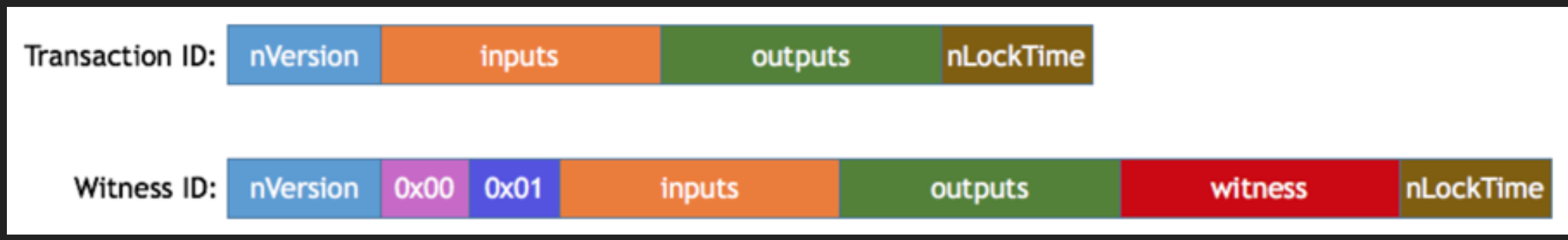
RE-ORGS

- A re-org involves rewinding one or more blocks, and then applying the blocks on the other side of the fork
- Nodes must store information about how to rewind their most recent blocks
- Bitcoin Core stores this rewind information in `revxxxxx.dat` files

SEGREGATED WITNESS

SEGWIT TRANSACTION SERIALIZATION

- SegWit adds two new P2P messages - **MSG_WITNESS_TX** and **MSG_WITNESS_BLOCK**
- **MSG_WITNESS_TX** serializes the transaction with the witness



- **MSG_WITNESS_BLOCK** is the same as **MSG_BLOCK**, but the transactions with witnesses are serialized with their witnesses.
- See BIP 144 for full details

THE WITNESS COMMITMENT

- Since SegWit activation, one of the outputs of the coinbase transaction must *commit* to the witness root
- The commitment is an OP_RETURN, followed by 0x24aa21a9ed, followed by the Merkle root of the witnesses in the block
- This commitment is placed in the scriptPubKey of one of the coinbase transaction outputs (if two scriptPubKeys match this pattern, use the one from the highest index output)
- See BIP 141 for full details