Incentives & Proof of Work

Alexander Schönhuth

UNIVERSITÄT BIELEFELD
Faculty of Technology

Bielefeld University
May 25, 2022

RECAP LECTURE 4

INTRODUCTION

- *Decentralization*
  - Introduction
  - Decentralization in Bitcoin
- *Distributed Consensus*
  - Motivation
  - Challenges
  - Impossibility Results
  - General Algorithm
- *Blockchain Consensus: Issues*
  - Concurrency
  - No Global Time
  - Node Failure
- *Blockchain Consensus: Algorithm*
  - Input Values
  - Elect, Vote and Decide
  - Disturbing Attacks

Incentives: Rewards

Preventing Sybil Attacks: Proof of Work

Hash Puzzles Properties

Mining Cost Bootstrapping 51-Percent Attack

Overview

Introduction

- *Incentives*
  - Block Rewards
  - Transaction Fees
- *Proof of Work*
  - Sybil attack: Recap
  - Proof of work: Key idea
  - Hash puzzles
- *Hash Puzzles: Properties*
  - Difficult to compute
  - Parameterizable cost
  - Trivial to verify
- *Mining Cost, Bootstrapping, 51-Percent Attack*
  - Mining cost: Basic calculation & complications
  - Bootstrapping: Feedback loop & recruiting miners
  - 51-Percent-Attack: Issues to consider

Bitcoin Consensus Algorithm: Recap I

Algorithm [Bitcoin Consensus]:

1. *Generating Input:*
    - New transactions are broadcast to all nodes
    - Each node collects new transactions into a block
2. *Elect:* A *random* node is selected to propose its block
3. *Vote:* Each node votes for block if found to
    - Nodes accept block if valid
        - Coins in transactions are unspent
        - Signatures successfully verified
    - Nodes vote for acceptance/rejection of block:
        - *Acceptance:* including its hash in the next block they propose
        - *Rejection:* hash of previous block included in next block
4. *Decide:* Block gets confirmed if accepted by majority of nodes
    - Initially, blockchain may branch; later, branches differ in length
    - Confirm only blocks in the longest branch

Bitcoin Consensus Algorithm: Recap II

Algorithm [Bitcoin Consensus]:

1. *Generating Input:* Input = blocks of transactions ✔
2. *Elect:* A *random* node is selected to propose its block
   - Probability proportional to value that resists Sybil attack
   - *Proof of Work (PoW):* in proportion to compute power
   - *Proof of Stake (PoS):* in proportion to bitcoins owned
3. *Vote:* Each node include hash of block in next block if valid ✔
4. *Decide:*
   - Only blocks of longest branch get confirmed
   - Confirm blocks at a delay, such that situation is clear
   - Pay reward to block creator only when confirming block

Bitcoin Consensus Algorithm: Recap II

Algorithm [Bitcoin Consensus]:

1. *Generating Input:* Input = blocks of transactions ✔
2. *Elect:* A *random* node is selected to propose its block
   - Probability proportional to value that resists Sybil attack
   - *Proof of Work (PoW):* in proportion to compute power
   - *Proof of Stake (PoS):* in proportion to bitcoins owned
3. *Vote:* Each node include hash of block in next block if valid ✔
4. *Decide:*
   - Only blocks of longest branch get confirmed
   - Confirm blocks at a delay, such that situation is clear
   - Pay reward to block creator only when confirming block

Incentives:
Rewards

Preventing
Sybil Attacks:
Proof of Work

Hash Puzzles
Properties

Mining Cost
Bootstrapping
51-Percent Attack

Bitcoin Consensus Algorithm: Recap II

Algorithm [Bitcoin Consensus]:

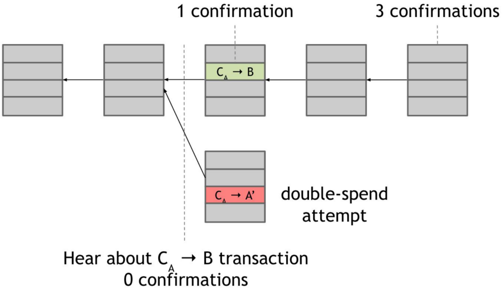
1. *Generating Input:* Input = blocks of transactions ✔
2. *Elect:* A *random* node is selected to propose its block
   - Probability proportional to value that resists Sybil attack
   - *Proof of Work (PoW):* in proportion to compute power
   - *Proof of Stake (PoS):* in proportion to bitcoins owned
3. *Vote:* Each node include hash of block in next block if valid ✔
4. *Decide:*
   - Only blocks of longest branch get confirmed
   - Confirm blocks at a delay, such that situation is clear
   - Pay reward to block creator only when confirming block

Decide: Recap III

- *Motivation:* With probability ≥ 0.5, pick honest block creator node
- Notion of honesty is problematic:
  - Majority of nodes acts in their own interest
  - Some nodes outright malicious
- Penalizing dishonesty impossible:
  - Morally illegitimate transaction cannot be spotted per se
  - Nodes have abstract identities, escapes generally applicable jurisdiction
- *Remedy:* Incentives for honest behaviour
  - However, abstract identities prevent conventional rewards

Solution: Pay bitcoins for honesty!

Rewards: Motivation



$k = 3$: only creator of block with $C_A \to B$ is rewarded

From `bitcoinbook.cs.princeton.edu`

Rule "Extend the Longest Branch": Incentive

- Consider paying *rewards* to block creators
- Pay only when block has $\geq k$ confirmations

If not following rule, reward at risk!

TYPES OF REWARDS

Block Rewards

- Block creator includes **coin creation transaction** into block
- Block creator can choose recipient of coin
  - ☞ Typically, recipient is himself
- *Interpretation:* Payment for block creation service
  - ☞ Block creation is expensive ... we will see

Reminder: *Coin creation transaction realized only when block on consensus chain*

Transaction Fees

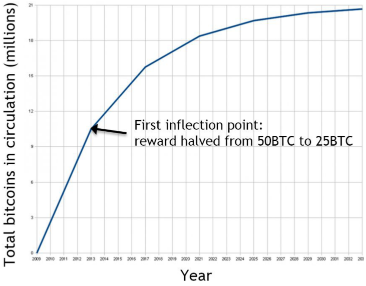
- Nodes broadcasting transactions can include transaction fees, to be paid to block creator
- *Interpretation:* Payment for realizing individual transactions

Reminder: *Transaction fees redeemed only when block on consensus chain*

Block Reward I

Block Reward: Mechanisms

- At the beginning (2009), block reward was 50 bitcoins
- Block reward halves every 210,000 blocks
- On average, blocks are created every 10 minutes

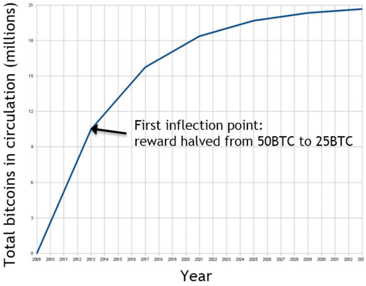


From `bitcoinbook.cs.princeton.edu`

Block Reward II

Block Reward: Properties I

- *Consequence:* Reward halves every 4 years (approximately)
- Current block reward: 6.25 Bitcoins (BTC)
- Last halving: May 11, 2020; next halving April 27, 2024



From `bitcoinbook.cs.princeton.edu`

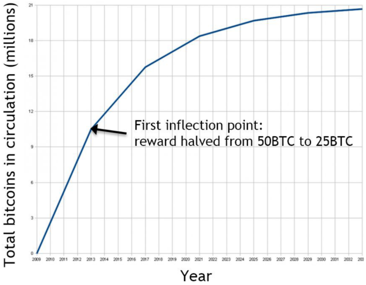
Block Reward III

c (1 + 1/2 + 1/4 + 1/8 + ...) -> 2c

c = 210000 * 50

Block Reward: Properties II

- *Geometric Series:* Final amount of 21 000 000 bitcoins
- Block reward *only* coin generation mechanism
- Block reward to run out in 2140: alternative reward necessary



From `bitcoinbook.cs.princeton.edu`

TRANSACTION FEES I

- A transaction consists of metadata, inputs and outputs
- Output specifies amount of Bitcoins to be sent to recipient
- Let total outputs $<$ total inputs: difference is *transaction fee*
- Creator of block including transactions with fees collects them
- *Note:* Transaction fees are voluntary
- *Incentive* for including them:
  - Transactions processed preferably
  - Supporting honest behaviour in general

Incentives:
Rewards

Preventing
Sybil Attacks:
Proof of Work

Hash Puzzles
Properties

Mining Cost
Bootstrapping
51-Percent Attack

Bitcoin Consensus Algorithm: Elect

Algorithm [Bitcoin Consensus]:

1. *Generating Input:* Input = blocks of transactions ✔
2. *Elect:* A *random* node is selected to propose its block
   - Probability proportional to value that resists Sybil attack
   - *Proof of Work (PoW):* in proportion to compute power
   - *Proof of Stake (PoS):* in proportion to bitcoins owned
3. *Vote:* Each node include hash of block in next block if valid ✔
4. *Decide:*
   - Only blocks of longest branch get confirmed
   - Confirm blocks at a delay, such that situation is clear
   - Pay reward to block creator only when confirming block

Sybil Attack: Recap

- *Sybil Attack:* Malicious participants can create large numbers of nodes by creating identities

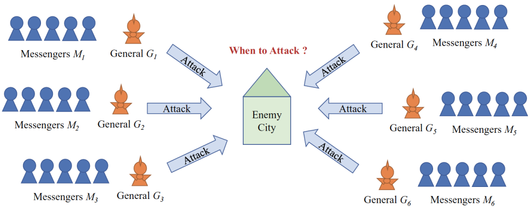
  ☞ increases probability to pick one of their nodes

- *Solution:* Probability for a node picked proportional to value malicious nodes cannot manipulate:
  - *Proof of Work (PoW):* In proportion to compute power of a node
  - *Proof of Stake (PoS):* In proportion to amount of bitcoins owned

- *Note:* A sybil attacker requires to distribute his compute power or coins across his identities

  ☞ No increase in probability to be elected

Proof of Work: Illustration I

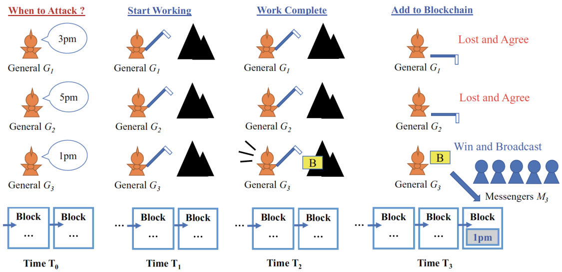


Generals need to agree on when to attack by exchanging messengers

From Kuo et al., 2019

- Byzantine army separated into divisions, each lead by a general
- Generals communicate by messenger when to launch attack
- *Goal:* Loyal generals should agree on good time without traitorous generals preventing this

Proof of Work: Illustration II

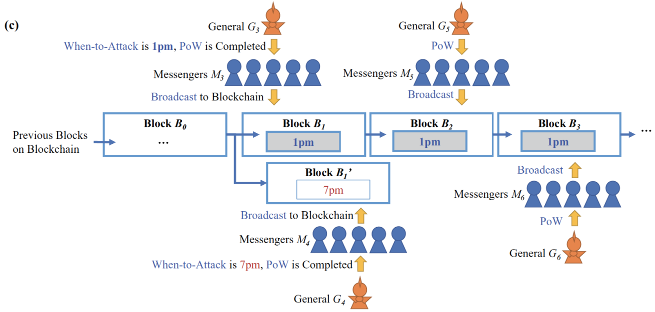


Generals have to "work hard" to suggest time

From Kuo et al., 2019

- The "hardest working general" wins, suggests time to attack
- Suggested time appended to block chain

Proof of Work: Illustration III



Mining process is repeated until consensus time materializes

From Kuo et al., 2019

- Times suggested by malicious generals yield orphan blocks
- Majority of generals honest ☞ reach greement on good time

Proof of Work: Key Idea I

- *Proof of Work (PoW):* Select leader in proportion to compute power owned
- *Proof of Stake (PoS):* Select leader in proportion to currency owned
- Sybil attack does not work:
  - A malicious participant cannot multiply compute power / coins by creating additional identies
  - So, malicious participant selected at equal probability, with or without additional identities
- In use at Bitcoin: Proof of Work

Proof of Work: Key Idea II

- *Issue:* Compute power cannot be determined explicitly
- *Solution:* Determine it implicitly

  ☞ Launch a *competition*

  - Everyone allowed to participate

    ☞ Everyone allowed to become elected leader
  - Chances proportional to amount of compute power invested
- *Competition:* Solve a hash puzzle
  - Use a *puzzle-friendly* hash function
  - This warrants a tough puzzle
  - Solving it requires considerable compute power

Incentives:
Rewards

Preventing
Sybil Attacks:
Proof of Work

Hash Puzzles
Properties

Mining Cost
Bootstrapping
51-Percent Attack

Proof of Work: Hash Puzzle I

Definition [Hash Puzzle]: Let

- Let H be a hash function
- B be a block; that is

$$B = prev\_hash \,||\, tx_1 \,||\, ... \,||\, tx_n$$

  where *prev_hash* is the hash of the block to be extended and $tx_i, i = 1, ..., n$ are strings spelling out transactions

- Let *target* be a particular value
- Let *nonce* be a number (”used only once”)

A *hash puzzle* consists in finding a *nonce* such that

$$H(nonce \,||\, B) < target$$

Proof of Work: Hash Puzzle II

- *Reminder:* H was defined to be *puzzle-friendly* if it was infeasible to find x in significantly less than 2^n trials such that

$$H(k||x) \in Y$$

  where k was a given string, and Y was a target set of possible hash values, defined by fixing n bits
- *Analogy:* Replace
  - k with the given block B
  - x with the nonce
  - Y with all bit strings that are smaller than target
  - Commonly, Y reflects all bit strings whose first n bits are zeroes

Conclusion: When using *puzzle-friendly* H, solving the hash puzzle requires brute force compute power.

Proof of Work: Hash Puzzle III

- *Conclusion:* Using *puzzle-friendly H* requires brute force compute power to solve puzzle
- In other words, the only way to solve the puzzle is to evaluate $H(nonce \,||\, B)$ using plenty of nonces
- Average time needed depends on size of target space
- That is, commonly, on how many zeroes the resulting value is to begin with
- Upon having found *nonce*, broadcast *nonce* and block *B*

  ☞ You are the winner! (Applause)

Hash Puzzles: Properties

Hash puzzles should be

1. Difficult to compute
2. Allow for parameterizable costs
3. Trivial to verify

Hash Puzzles: Difficult to Compute I

- Difficulty essentially warranted by puzzle-friendly hash function
- Further, target space required to be sufficiently small

  ☞ In other words, hashed values to start with sufficiently many zeroes
- As of 2014, the difficulty level was about 10^{20} hashes per block

  ☞ Amounts to fixing more than 60 initial bits to zero
- Size of the target space only $1/10^{20}$ of size of output space overall
- See e.g. `https://www.coinwarz.com/mining/bitcoin/difficulty-chart` for current status

Hash Puzzles: Difficult to Compute II

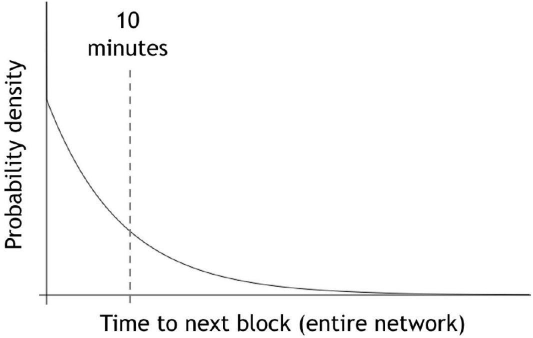
Bitcoin Mining

- *Bitcoin mining*: process of trying to solve hash puzzles
- *Miners:* nodes partaking in mining process
- *"Mining"* because successful participation creates new coins

Hash Puzzles: Parameterizable Cost I

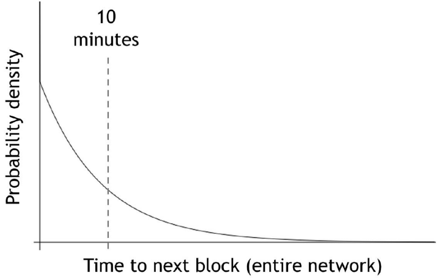
- Cost of mining is not fixed
- Every 2016 blocks, all nodes recalculate the target
- In other words, recalculate size of target space in relation to size of output space
- *Goal:* Average time between successive blocks approximately 10 minutes
- *Consequence:* Recalculation to happen roughly every 2 weeks
- *Motivation I:* Sufficiently *much* time to prevent (e.g. latency-induced) interference
- *Motivation II:* Sufficiently *little* time to realize transactions

HASH PUZZLES: PARAMETERIZABLE COST II



- Each trying of a nonce is a *Bernoulli trial*
- The probability p to succeed is very small (above: $p = 10^{-20}$)
- There are plenty if trials
- Low probability + plenty of trials ☞ *Poisson distribution*

HASH PUZZLES: PARAMETERIZABLE COST II

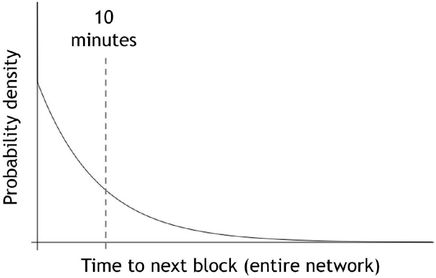


Waiting time to next block: exponential distribution

From `bitcoinbook.cs.princeton.edu`

- Each trying of a nonce is a *Bernoulli trial*
- The probability p to succeed is very small (above: $p = 10^{-20}$)
- There are plenty if trials
- Low probability + plenty of trials: ☞ *Poisson distribution*

Hash Puzzles: Parameterizable Cost III



Waiting time to next block: exponential distribution

From `bitcoinbook.cs.princeton.edu`

- The Poisson distribution governs number of occurrences per time
- The corresponding waiting time to next block is an *exponential distribution*
- That means that the average time to success is independent of what happened before

HASH PUZZLES: TRIVIAL TO VERIFY

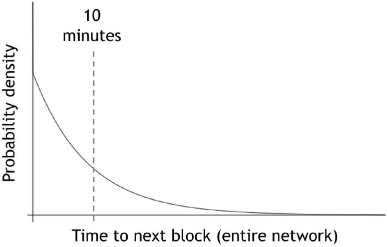
Verification

- Hash the nonce published together with the block suggested
- Resulting hash has sufficiently many initial zeroes ☞ verified!

Consequences

- Instant verification by any miner / node
- No central ”election authority” necessary

MINING: INDIVIDUAL WAITING TIME



From `bitcoinbook.cs.princeton.edu`

- Let $0 < q_M < 1$ be the fraction of hash power owned by miner M
- Then T_M, the mean time for M to find a block evaluates as

$$T_M = \frac{10\,min}{q_M}$$

- $q_M = 0.001$ (= 0.1%) yields $T_M = 10\,000$ minutes ☞ about a week

Incentives:
Rewards

Preventing
Sybil Attacks:
Proof of Work

Hash Puzzles
Properties

Mining Cost
Bootstrapping
51-Percent Attack

Mining Cost: Basic Calculation

- *Mining cost (MC):* Expenses for hardware and electricity to do the mining
- *Mining rewards (MR):*
  - Block reward BR
  - Transaction fees TF
  - So, $MR = BR + TF$
- *Calculation:* If $MR > MC$, miner profits

  ☞ only under that condition, miner is willing to mine

Mining Cost: Complications

- *Electricity cost* varies over time

  ☞ hard to control

- *Rewards MR* depend on rate of hash power owned
  - Rate depends on hash power brought in by other miners
  - The more hash power overall, the smaller *MR*

- *MR* measured in bitcoins (BTC), while *MC* measured in terms of national currency ("NC")

  ☞ Profit depends on BTC-to-NC exchange rate

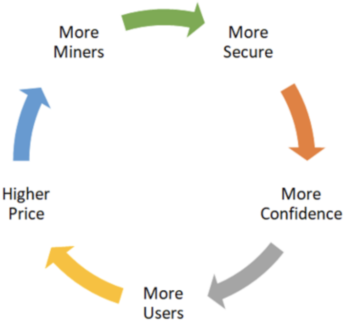
- Miner may invest in trying strategies alternative to brute force

- *Summary:* Complicated game theory problem without conclusive answer so far

Bootstrapping I

Getting Bitcoin to Work

- Tricky interplay between
  - *Security* of the blockchain
  - *"Healthiness"* of the mining "ecosystem"
  - *Value* of the currency
- Feedback loop:
  - Healthiness implies security
  - Security implies value
  - Value implies healthiness



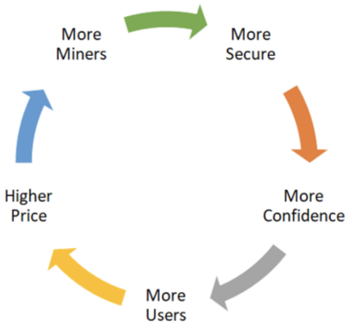
Bitcoin Feedback Loop

From `seekingalpha.com`

Bootstrapping II

Initial Situation

- Initially Satoshi Nakamoto only miner/user
- *Priority*, because only lever
  - Recruiting miners
  - Recruiting users
- Recruiting users:
  - Create media attention
- Recruiting miners by rewarding them



Bitcoin Feedback Loop

From `seekingalpha.com`

UNIVERSITÄT BIELEFELD

Bootstrapping III

Recruiting Miners

- Pay miners *block reward*
- *Reward decreases over time:*
  - Initial miners: many BTC
  - Early miners: particular incentive for honesty
  - Majority of honest miners establishes
  - Value of BTC increases
  - Need for honest miners decreases over time
- Explains block reward halving

Bitcoin Feedback Loop

From seekingalpha.com

51 PERCENT ATTACK I

DEFINITION [51-PERCENT-ATTACKER]:
A *51-percent attacker* is a group of malicious nodes that controls 51% of the hashing power.

51 Percent Attack: Things to Consider

- Stealing bitcoins
- Suppressing transactions
- Changing block reward
- Destroying confidence

51 PERCENT ATTACK II

Stealing Bitcoins

- Attacker steals bitcoin through invalid transaction
  - Signature invalid, but ...
  - ... attacker prevents validity by extending block containing transaction
  - Over time, transaction realizes *stealing coin is possible!*
- *Spending stolen coin:*
  - Node to receive stolen coin notices invalid signature
  - Refuses service
  - *Spending stolen coin impossible!*
- *Summary:* Stealing coins requires to subvert cryptography, and not only consensus

51 Percent Attack II

Suppressing Transactions

- Particular transaction not included in attacker's blocks
  - Transactions never get on longest chain
  - So they never realize
- Nevertheless, transaction is broadcast to network
  - Majority of nodes sees that transaction does not materialize
  - 51 percent attack becomes apparent
- *Summary:* Suppressing transactions reveals 51-percent attack

51 Percent Attack IV

Changing Block Reward

- Attacker needs to get in control of bitcoin software
- *Summary:* Changing block reward impossible

Destroying Confidence

- 51-percent attacks becoming apparent make users loose trust
- Users disappear ☞ value decreases
- Loss of confidence in fact inevitable
- *Summary: Destoying confidence possible, but improfitable for attacker*

MATERIALS / OUTLOOK

- See *Bitcoin and Cryptocurrency Technologies*, 2.4 – 2.5
- See `https://bitcoinbook.cs.princeton.edu/` for further resources
- Next lecture: “Bitcoin: Technical Mechanics”; Healthcare Application I
    - See *Bitcoin and Cryptocurrency Technologies* 3
    - See [Chen et al., Journal of Medical Systems 43(5), 2019]; `https://doi.org/10.1007/s10916-018-1121-4`