Blockhains and Distributed Ledgers Lecture 04

Aggelos Kiayias

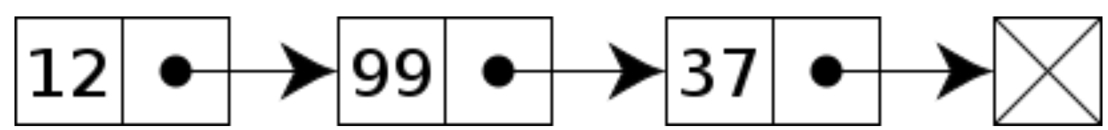
THE UNIVERSITY of EDINBURGH

Lecture 04

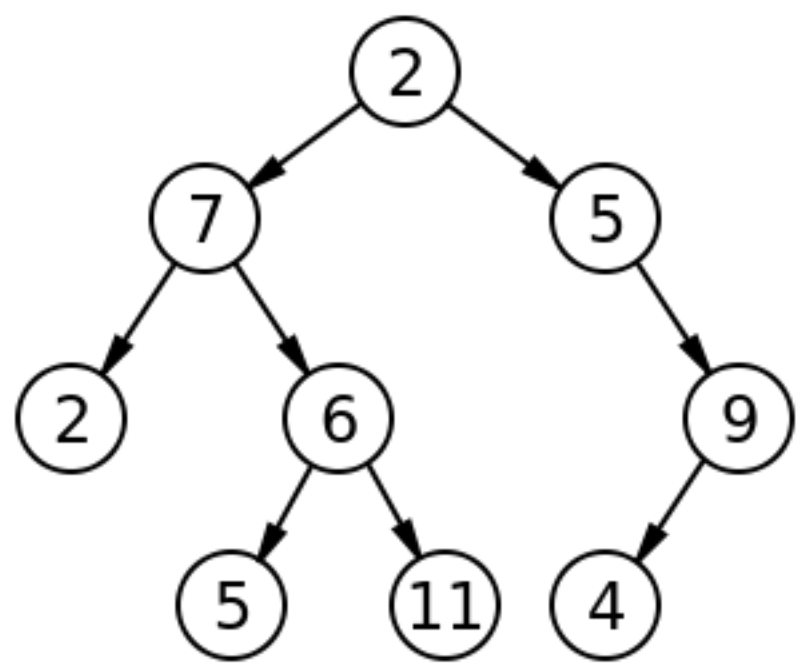
- Blockchain Protocol Specifics
  - Data structures for blockchain protocols
  - Blockchains & Variable Difficulty
  - Blockchain Protocol Variants

Recursive Data Structures, I

- Many data structures are defined recursively:
    - linked lists



    - trees



Recursive Data Structures, II

- A new data structure instance *D* can be defined given one or more instances of the data structure (as well as additional data).

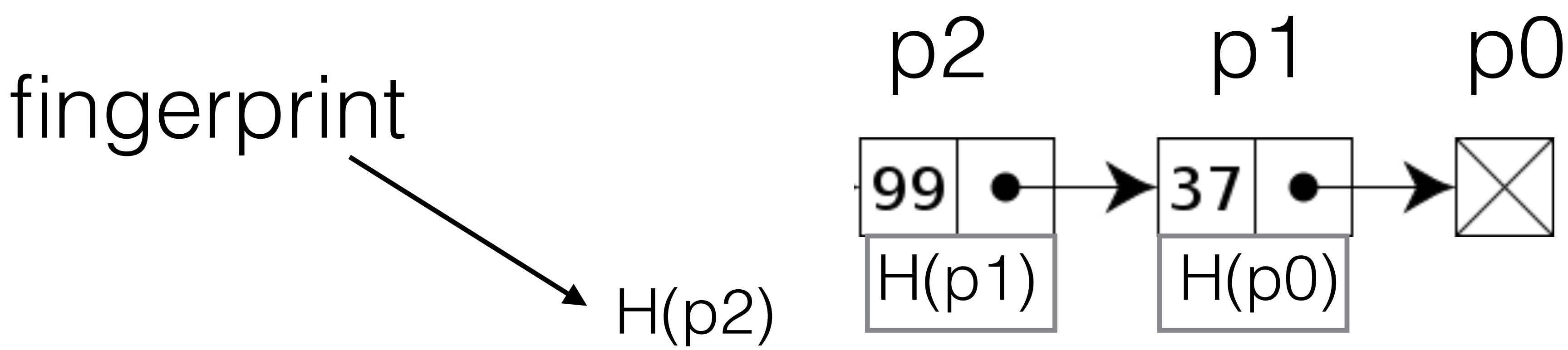
- Typical operations of interest: membership, append, insert, delete.

Authenticated Data Structures

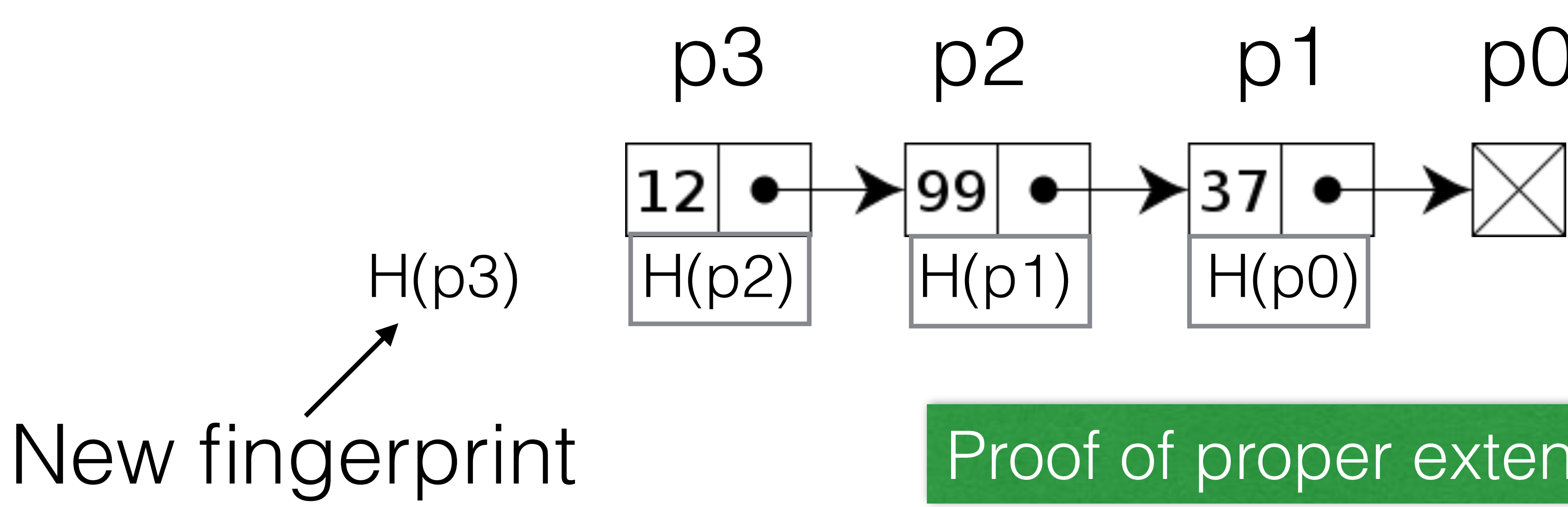
- A data structure instance has a short **fingerprint**, and operations on it can be outsourced to an **untrusted prover** who updates the representation as well as proves the update is correct.

Hash Chain

(authenticated linked list)



Append operation

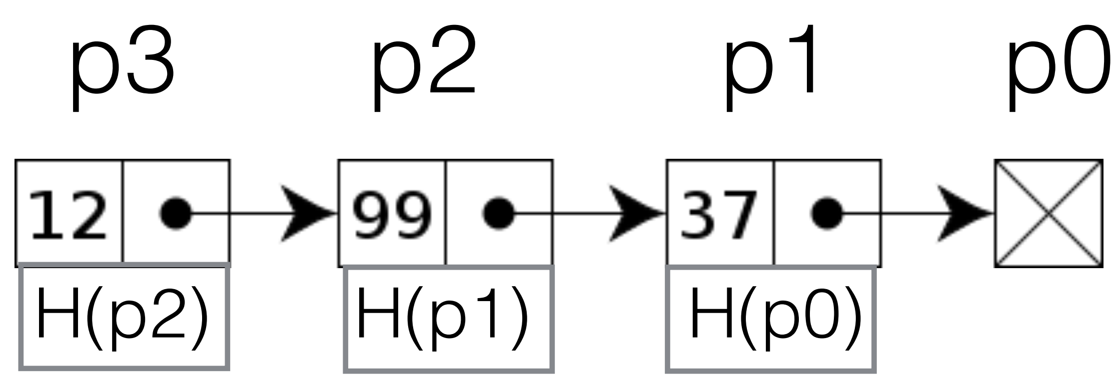


Proof of proper extension= show p3, p2

Security Properties

- If you hold the fingerprint, it unlikely that someone can misrepresent the contents of the data structure.

- e.g., if you hold H(p3) if you are presented with a linked list different than



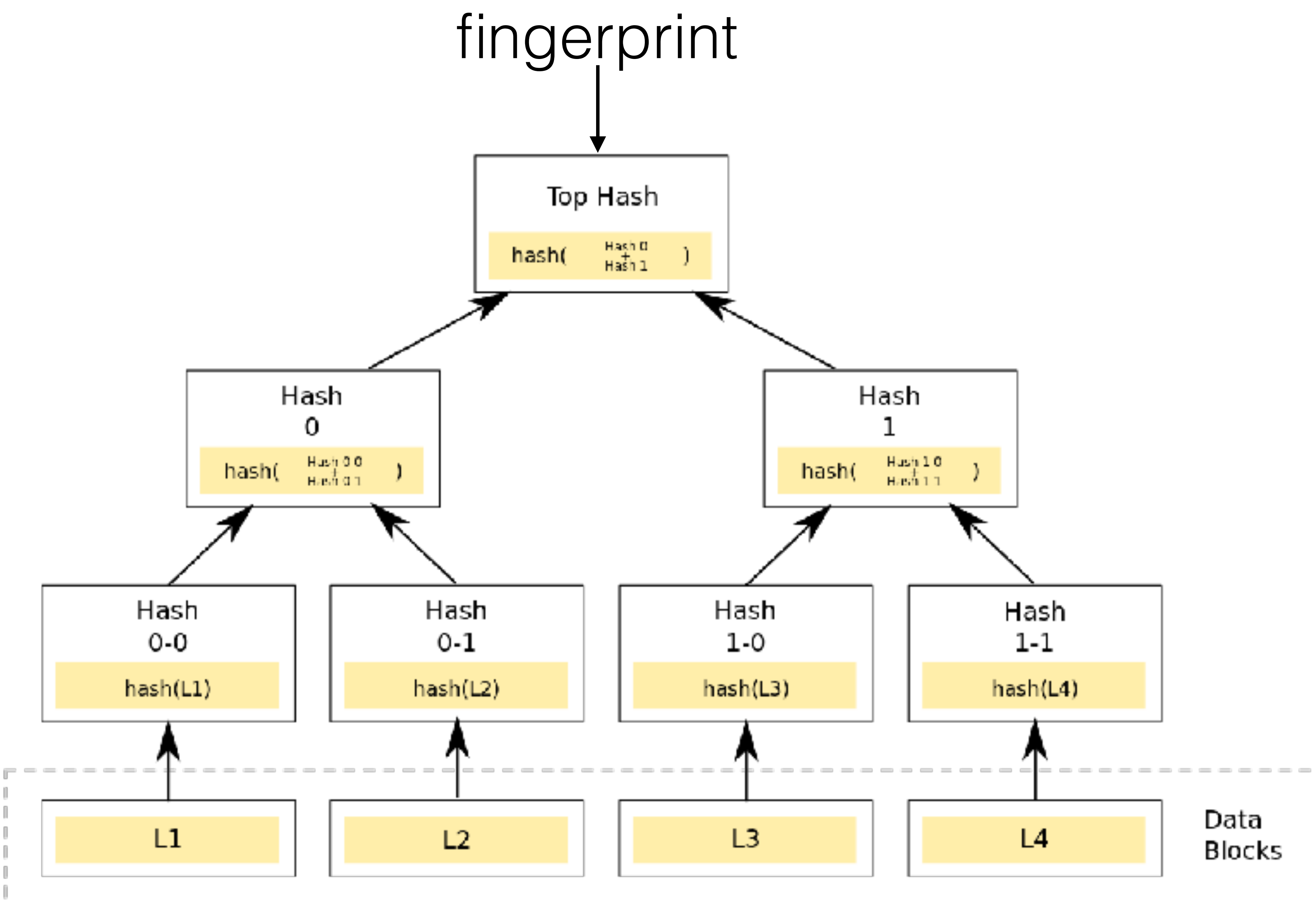
then a collision against H has occurred.

Hash Chain Operations

(assuming one holds only
the fingerprint)

- Efficient
  - append (extend) operation
- Not efficient
  - membership, insert, delete

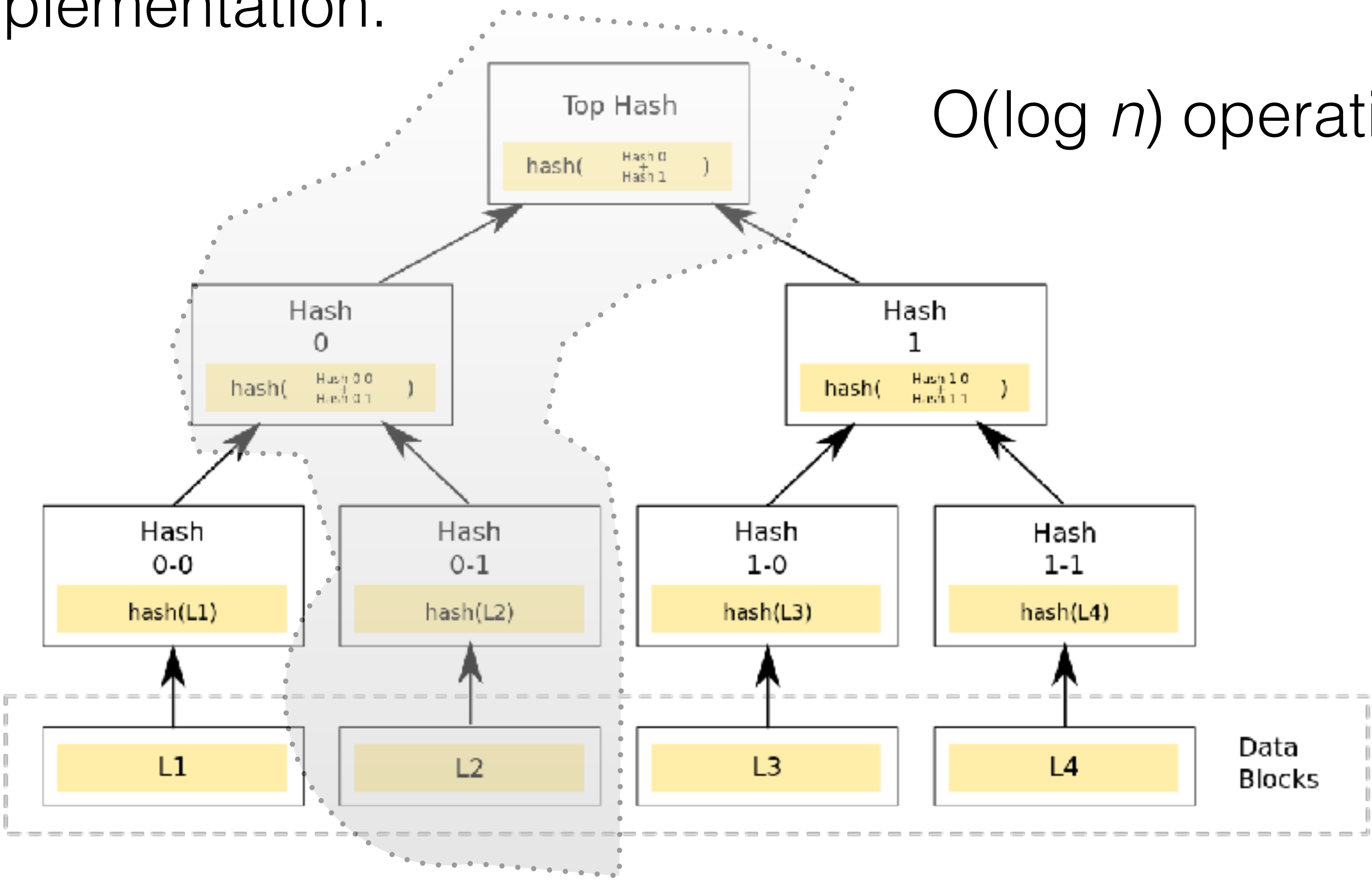
Merkle (Hash) Tree



Properties of Merkle Trees

- More efficient membership, insert, delete, implementation.

O(log *n*) operations



Bitcoin Block Structure

The Block Header :

Field	Purpose	Updated when...	Size (Bytes)
Version	Block version number	You upgrade the software and it specifies a new version	4
hashPrevBlock	256-bit hash of the previous block header	A new block comes in	32
hashMerkleRoot	256-bit hash based on all of the transactions in the block	A transaction is accepted	32
Time	Current timestamp as seconds since 1970-01-01T00:00 UTC	Every few seconds	4
Bits	Current target in compact format	The difficulty is adjusted	4
Nonce	32-bit number (starts at 0)	A hash is tried (increments)	4

Example

Block #488929

Summary	
Number Of Transactions	2049
Output Total	2,313.58218724 BTC
Estimated Transaction Volume	402.7366234 BTC
Transaction Fees	0.41621077 BTC
Height	488929 (Main Chain)
Timestamp	2017-10-08 19:47:29
Received Time	2017-10-08 19:47:29
Relayed By	BTC.TOP
Difficulty	1,123,863,285,132.97
Bits	402717299
Size	999.212 kB
Weight	3632.966 kWU
Version	0x20000000
Nonce	2507403905
Block Reward	12.5 BTC

Hashes	
Hash	000000000000000000392ec35744060a05d3cb3031baaa60541ca956b
Previous Block	000000000000000000f8957d44a4bb57ac9e61a7c4e6f089c6eac0d3e9
Next Block(s)	
Merkle Root	3ac688c88b06241ea5c6850fe178c0de548f1739b3c2874db519674b73

fingerprint of
hash tree of transactions

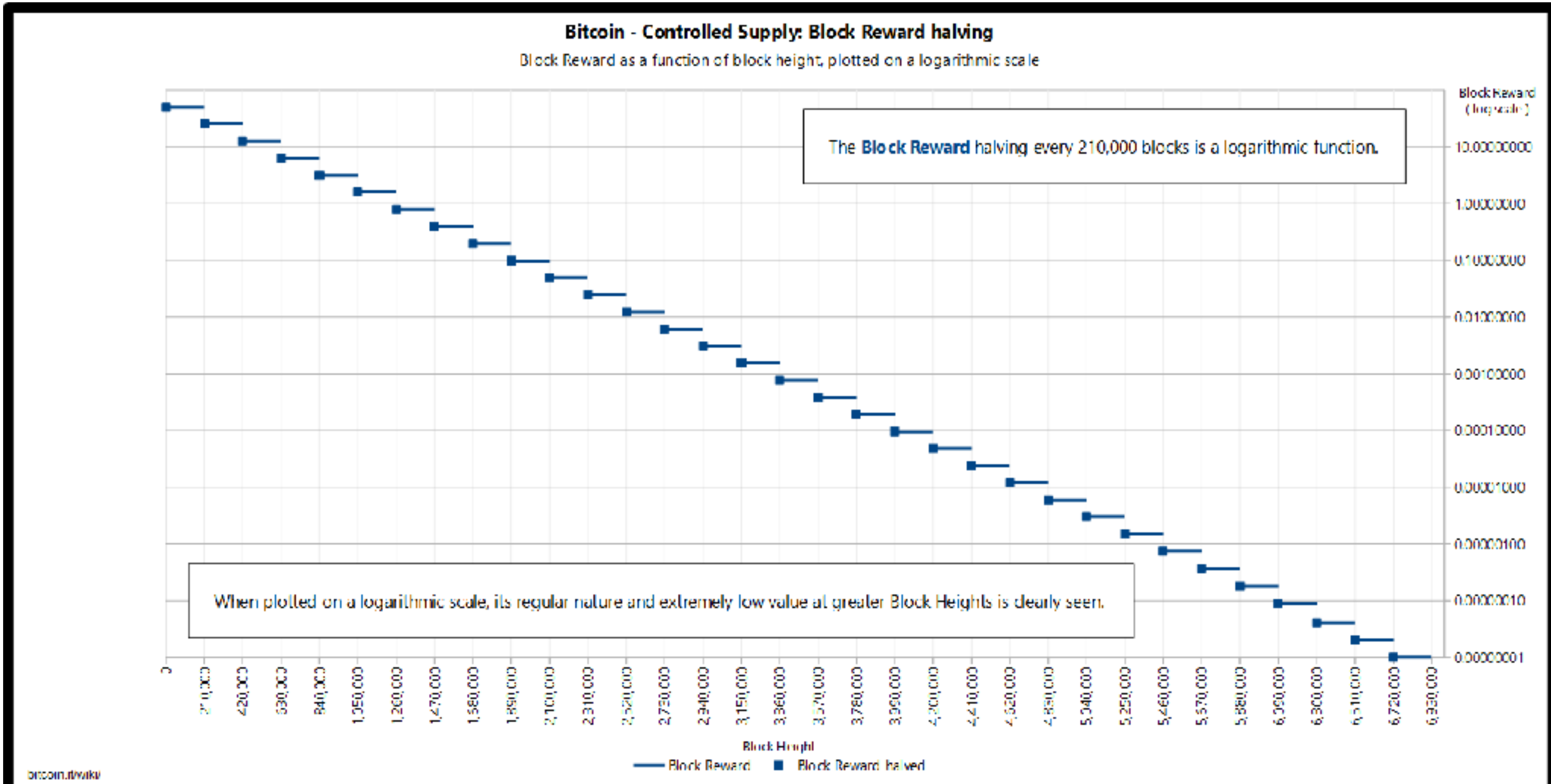
ratio w.r.t. initial target

https://blockchain.info

Bitcoin generation

- A number of bitcoin is created as a reward for mining a new block.
- Special **coin-base** transactions is included in every block.
- Bitcoin supply is limited with 50 BTC being the initial reward per block.

Bitcoin - Controlled Supply



Bitcoin Proof of Work

int counter;
counter **=** 0
while Hash(*block_header, counter*) **>** Target
  increment counter

block_header contains a coinbase transaction which contains an extraNonce parameter

if transactions remain the same extraNonce has to be modified to avoid repeating work

Target Difficulty over time

- Initially required approximately 2^{32} hashing operations.
- Currently, [Oct.2017], 2^{72} hashing operations.

https://www.coindesk.com/data/bitcoin-mining-difficulty-time/

Appreciating Hashing Operations

- Consider a regular PC that can do 30 MHash / sec
- With expectation of 2^{72} hashing operations, mining a block will require ~ **5 million years**.

https://en.bitcoin.it/wiki/Non-specialized_hardware_comparison

Lowering the variance of rewards

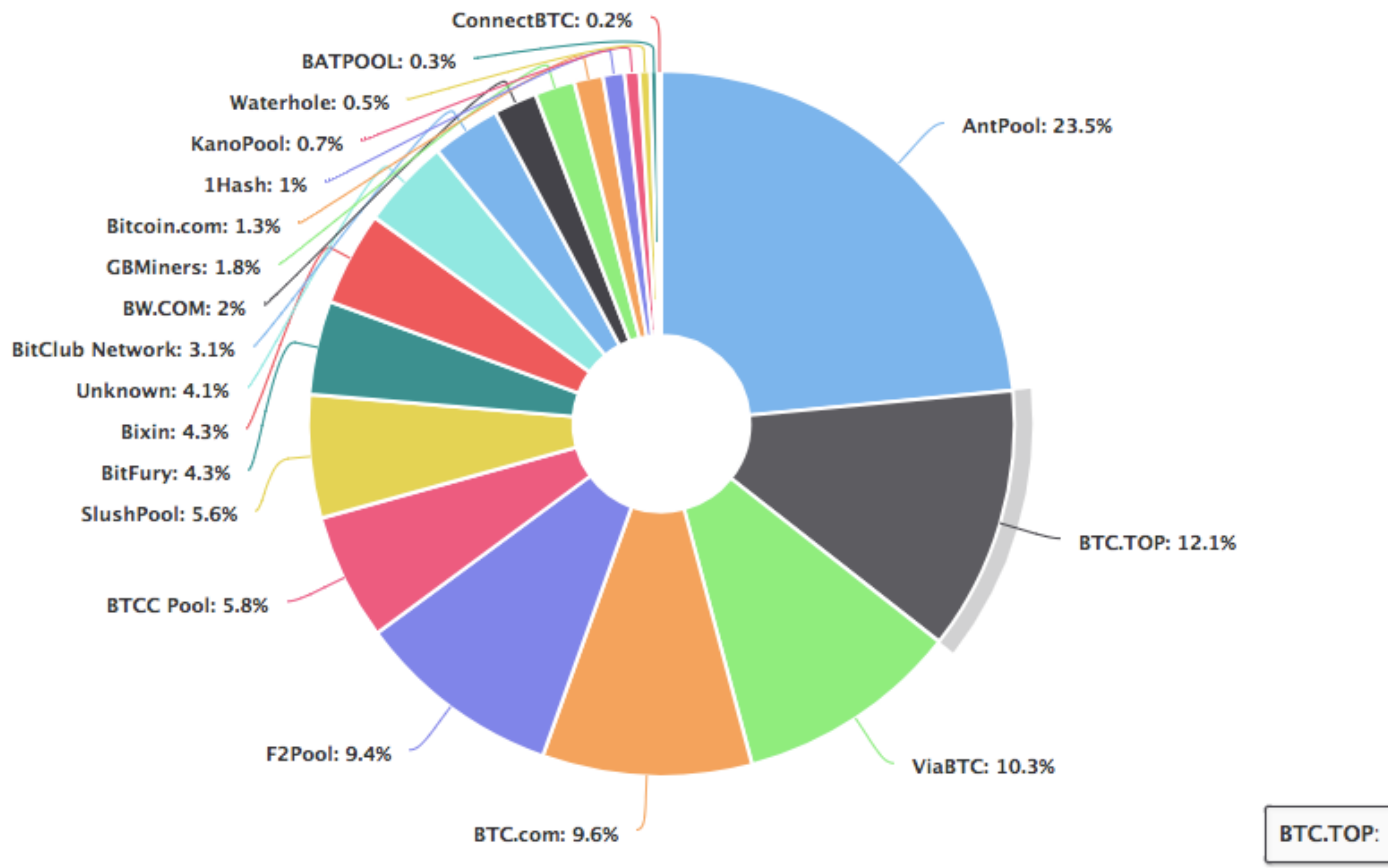
- Bitcoin's Proof of Work puzzle can be parallelized
  - Mining pools
    - Instead of working separately, work **together** to solve the same block.
    - When one member of the pool finds a block the reward is spread to all pool members.
    - Claiming a portion of a reward by collecting **shares** (small hashes that are not quite as small as needed)

Claiming pool rewards

- A pool member can prove that it is a member of a mining pool by providing shares.
  - flat pay per share.
  - proportional since last block.
    - + .. many other variants.

Current Mining Pools

[Oct. 2017]



Evolving populations of miners

- if the number of miners change (or their respective hashing power), the protocol should adjust.
- how to do that?

Revisiting the backbone protocol

maxvalid
is changed so that
parties adopt chain with highest difficulty
linearly related to

$$\sum_i \frac{1}{T_i}$$

The *f* parameter

f = probability of producing a block in a round of interaction

(depends on target *T*, # of miners *n, and duration of round*)

- If *f* becomes too small, parties do not do progress; chain growth becomes too slow. [liveness is hurt]
- if *f* becomes too large, parties "collide" all the time; an adversary, exploiting network scheduling, can lead them to a forked state. [persistence is hurt]

To resolve this in a dynamic environment, bitcoin **recalculates the target** *T* to keep *f* constant $f(T, n) \approx f(T_0, n_0) = f_0$

Target Recalculation

$n_0 =$ estimation of the number of ready parties at the onset

$T_0 =$ initial target

$m =$ epoch length in blocks

$\tau =$ recalculation threshold parameter

$T =$ target in effect

$$pT = \text{prob of a single miner getting a POW in a round}$$

$$\text{next target} = \begin{cases} \frac{1}{\tau} \cdot T & \text{if } \frac{n_0}{n} \cdot T_0 < \frac{1}{\tau} \cdot T; \\ \tau \cdot T & \text{if } \frac{n_0}{n} \cdot T_0 > \tau \cdot T; \\ \frac{n_0}{n} \cdot T_0 & \text{otherwise} \end{cases}$$

$\Delta =$ last epoch duration based on block timestamps

$$n = \frac{m}{pT\Delta}$$

the "effective" number of parties of the epoch

Bahack's Attack

- The recalculation threshold is essential.
  - Without it, an adversary can create a private, artificially difficult chain that will increase the variance in its block production rate; overcoming the chain of the honest parties becomes a non-negligible event.

Understand the attack : clay pigeon shooting

clay pigeons

A clay pigeon shooting game

- Suppose you shoot on targets successively from 10m against an opponent
  - your success probability 0.3 vs. 0.4 that of your opponent.
  - You shoot in sequence 1000 targets. The winner is the one that got the most hits.
- What is your probability of winning?

Analysis, I

- You have an expectation of 300 hits and your opponent has an expectation of 400 hits.
- What is your probability of winning?
- Denote by X, whether you hit a target, and similarly Y for your opponent. From Chernoff bounds

$$\mathbf{Pr}[\sum_{i=1}^{1000} X_i \geq 345] \leq \exp(-(0.15)^2 300/3) < 11\%$$

$$\mathbf{Pr}[\sum_{i=1}^{1000} Y_i \leq 348] \leq \exp(-(0.13)^2 400/2) < 3.5\%$$

Analysis, II

- If the negation of both these events happens you will certainly loose

$$\mathbf{Pr}[X_{<345} \wedge Y_{>348}] = (1 - \mathbf{Pr}[X_{\geq 345}])(1 - \mathbf{Pr}[Y_{\geq 348}]) \geq 85\%$$

- Thus the probability of you winning is below 15%

Analysis, III

- Now you are given a choice: you can decrease the size of the clay pigeon target by a ratio β and augment your "kills" by multiplying with $1/\beta$.
- Suppose your accuracy is just linear with β.
    - do you accept to play like this (while your opponent will keep playing in the same way) ?

Analysis, IV

- The expectation remains the same:

$$E[X_i'] = \mathbf{Pr}[X_i' = 1]/\beta = 0.3\beta/\beta = 0.3$$

$$\mathbf{Pr}[\sum_{i=1}^{1000\beta} X_i \geq 345\beta]$$

$$\leq \exp((-0.15)^2 300\beta/3))$$

β	bound
1,	~10.5%
0.5,	~32.4%
0.25,	~56.9%
0.10,	~79.8%

decreasing β results in
increased variance and
our concentration argument will fail

Alternative Blockchains

Faster block production rate

- Early variations in the parameter *f*.

Cryptocurrency	block gen. rate (sec)	f (blocks/round)	$1/f$
Bitcoin	600	0.021	47.6
Litecoin	150	0.084	11.9
Dogecoin	60	0.21	4.76
Flashcoin	$6-60$	0.21-2.1	0.476-4.76
Fastcoin	12	1.05	0.95
Ethereum[3]	12	1.05	0.95

assuming a round is ~ 12.6 seconds

Variations in Proof of Work

- Bitcoin's proof of work algorithm relies on an inequality of the form SHA256(SHA256(.)) < Target
- SHA256 was not designed specifically with this specific type of application in advance.
- A characteristic that can be advantageous is **memory hardness**.

Memory Hard Functions

- Assume space parameter *n*.
- The function *f* is memory hard, if using for any algorithm computing *f* with space *S* in time *T* it holds that ST = O(*n^2)*
- **Relevance**. Building dedicated hardware to mine will not be as advantageous compared to a general purpose CPU since they will have to be replaced when difficulty changes.
- Example: **scrypt**, notably used in **Litecoin**

Proof of Space

- Instead of proving you have spent time, prove that you have allocated space.

- One advantage lies on the fact that memory (e.g., hard disk space is not a wasted resource and can be repurposed).

- A challenge is that since proofs of space do not require effort to produce analyzing the security of the resulting blockchain protocol requires a different approach.

Spacemint

Proof of Elapsed Time

- Prove that time has elapsed using a trusted execution environment (TEE) like Intel's SGX.
- No waste of resources whatsoever.
- Analysis essentially identical to that of bitcoin is possible.
- The main disadvantage is being locked-in and dependent to the TEE manufacturer.

Sawtooth Lake

Proof of Stake

- Determine eligibility to issue next block based on coin balance as reported in the blockchain.
- The advantage is that no physical resource is wasted whatsoever.
- A challenge is that since proofs require no effort to produce a different analysis is needed. Moreover, a randomized selection would be necessary which can be subject to adversarial bias.

NXT, PPCoin, Cardano

How to design a PoS?

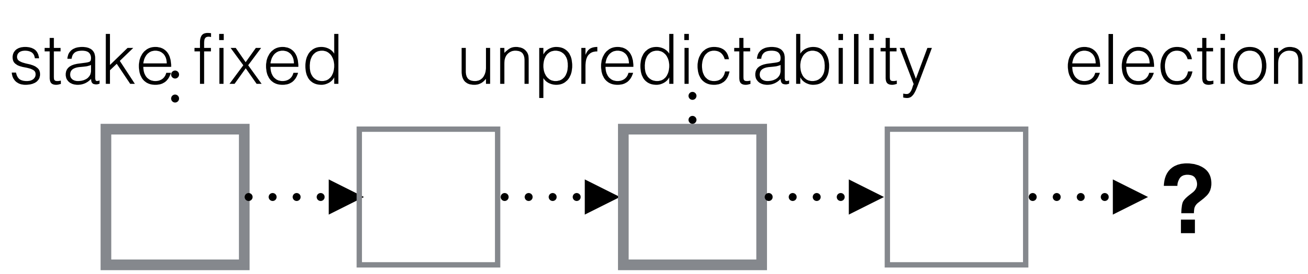
- Start with an initial stakeholder distribution.
- Divide time in communication rounds.
- Determine the winner of each round at random proportional to its stake as reported in the blockchain.

Electing Stakeholders,I

- How to elect someone proportional to its stake?

- If the election function can be computed in advance, then an attacker can try many possible accounts until it finds one that is likely to win. It can then transfer funds to that account.

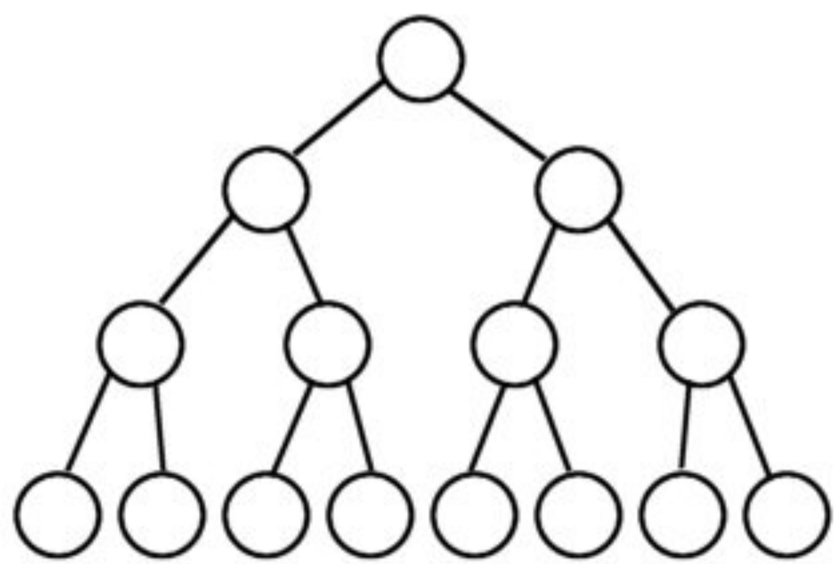
- One way to resolve this is to use somewhat “old” stake and then have some **randomness** incorporated into the election function.



Electing Stakeholders,II

A simple process for selecting stakeholders:

follow the satoshi



flip a coin
at each step

all satoshis in circulation

Return the account that owns the satoshi

[satoshi = smallest denomination in bitcoin]

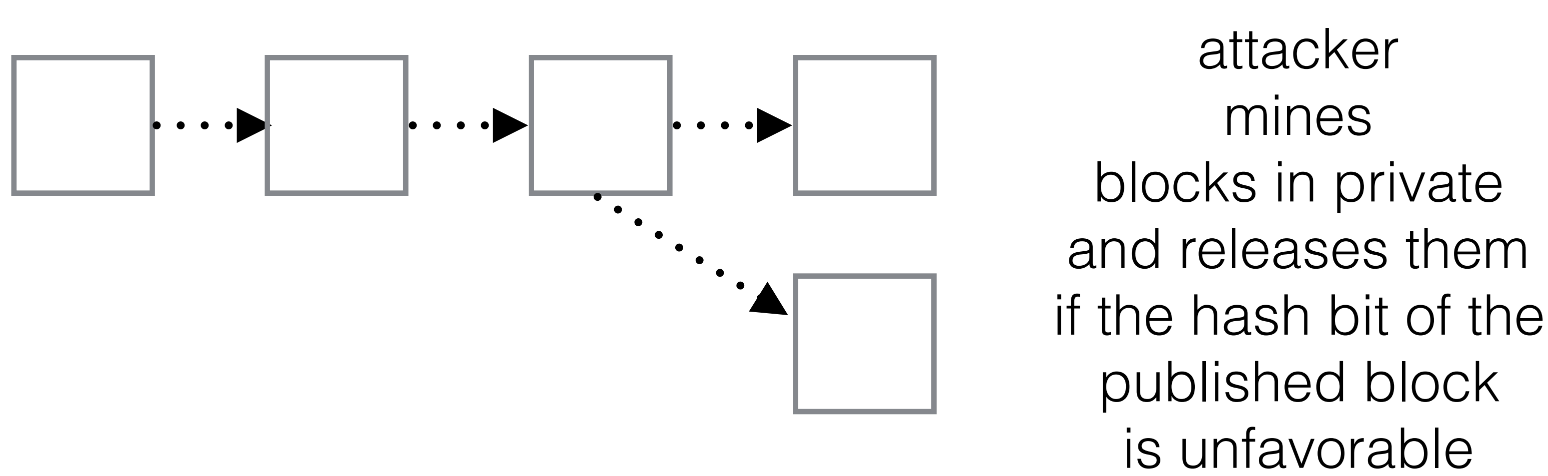
Publicly Verifiable Randomness, I

- But how to flip a coin?
  - Requirements: (i) nobody should be able to bias it to their advantage, (ii) the outcome should be publicly verifiable.

Publicly-Verifiable Randomness, II

- A first approach: just use (some bits of) the hash of the previous block.

- Downside: assuming ‘longest chain’ or similar rule is used this can be prone to a block withholding attack.



Example

- Imagine a coin is flipped publicly and you also flip a coin behind your back.
- Then with probability γ you are given the opportunity to switch the two coins (if you like).
- What is the distribution of this experiment?

$$\mathbf{Prob}[out = 1] = \frac{1}{2}(1-\gamma) + \frac{1}{2}\gamma\frac{1}{2} = 1/2 - \gamma/4$$

Publicly Verifiable Randomness, III

- Can we remove the bias? Recall Blum's coin flipping protocol.
- Alice commits to Bob, a random coin. Bob responds with his own coin. Alice opens the commitment and they both produce the output.
- The protocol easily generalizes to *n* parties.
- But what if one of the parties **aborts**? Observe that there is always going to be one of the parties that knows the output of the protocol in advance.
  and how do we publicly verify the outcome? **publicly verifiable secret-sharing** can be used.

Other Approaches to Verifiable Randomness

- Use an external randomness source.


The New York Times.

ARMISTICE SIGNED, END OF THE WAR

OUSTED KAISER FLEES TO HOLLAND

  - Why do you trust it?
  - How do you ensure that everyone has access to it?

Picture by Geoff Elston - Society for Popular Astronomy, Solar section, http://www.popastro.com/solar/solarobserving/chapter.php?id_pag=30

End of Lecture 04

- Next lecture
    - Incentives and Blockchain Protocols

Extra Slides

Secret-Sharing

- Consider random *N* values subject to the constraint

$$\sum_{i=1}^{N} x_i = x \quad \text{(over a finite group)}$$

- This is called a secret-sharing.

- Observe knowledge of any *N*-1 values is not helpful in any way to infer information about *x*

First step towards Fair Coin Flipping

- Players commit their coin in the blockchain and also include a secret-sharing of the opening of the commitment so that any subset of $N/2$ parties can reconstruct the opening. Shares should be encrypted with

- Thus, in this way if one party aborts the protocol, assuming at least $N/2$ parties continue they can recover the share.

Publicly Verifiable Secret-Sharing

- A powerful primitive for multiparty protocols.
  - Each party has a public-key.
  - The dealer creates shares that are distributed in encrypted form.
  - The shares provided by the dealer can be **publicly verified** as correct.
  - Verifiability should not leak information about the secret.

PVSS Challenges

- Assuming $$\sum_{i=1}^{N} x_i = x \qquad \psi_i = \mathcal{E}_i(x_i)$$ $$\psi = \mathbf{Com}(x)$$
- Verify that the value committed in ψ_i satisfies the equation with respect to the values encrypted in ψ

The cryptographic tool that solves the above problem is called a **zero-knowledge proof**.