Stanford University

NUS National University of Singapore

VISA Research

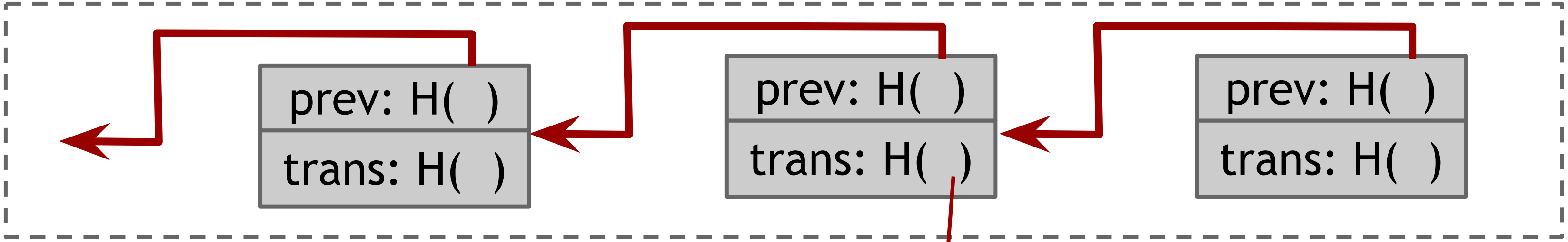
FLYCLIENT

SUPER LIGHT CLIENT FOR CRYPTOCURRENCIES

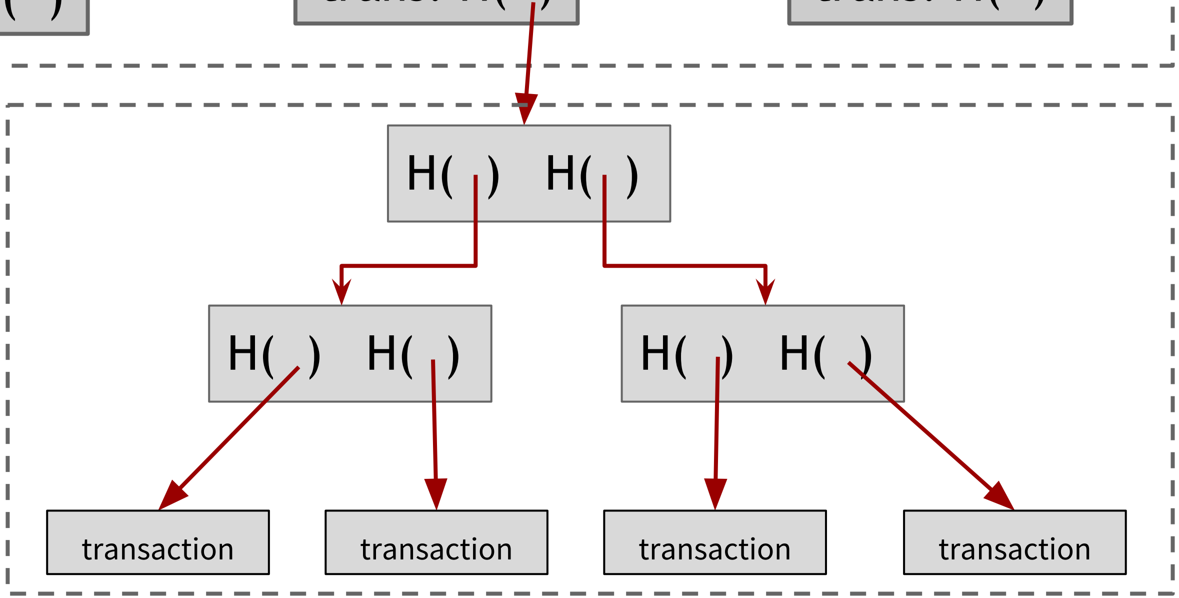
Loi Luu, Benedikt Bünz, Mahdi Zamani

Recall: Bitcoin blockchain format

Hash chain of blocks

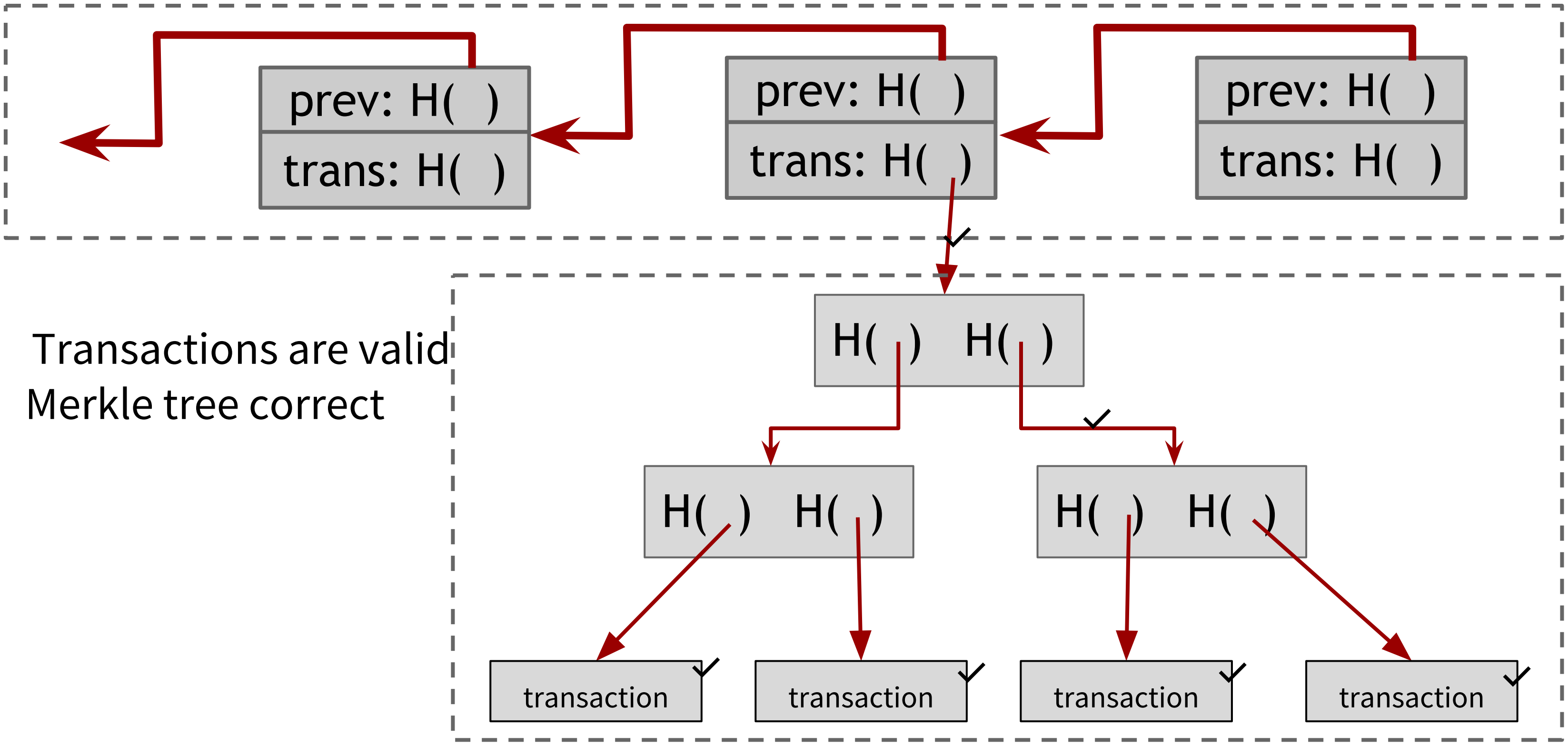


Hash tree (Merkle tree) of transactions in each block



Validity of a blockchain

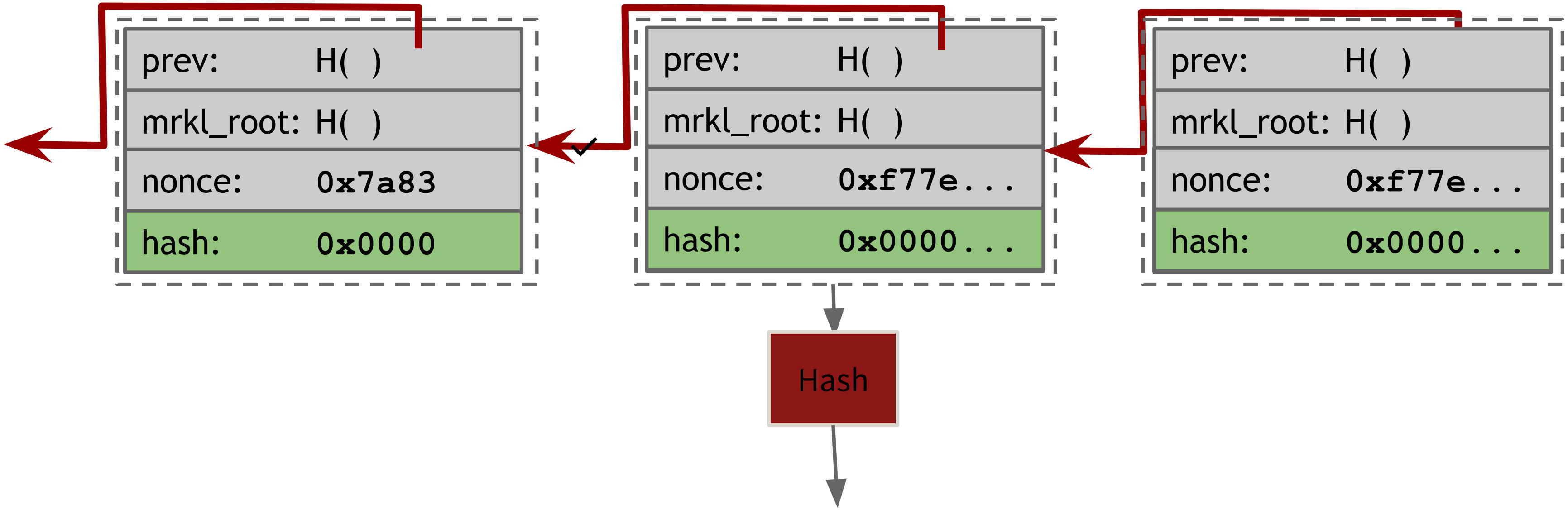
Hash chain of blocks



1. Transactions are valid
2. Merkle tree correct

Stanford University

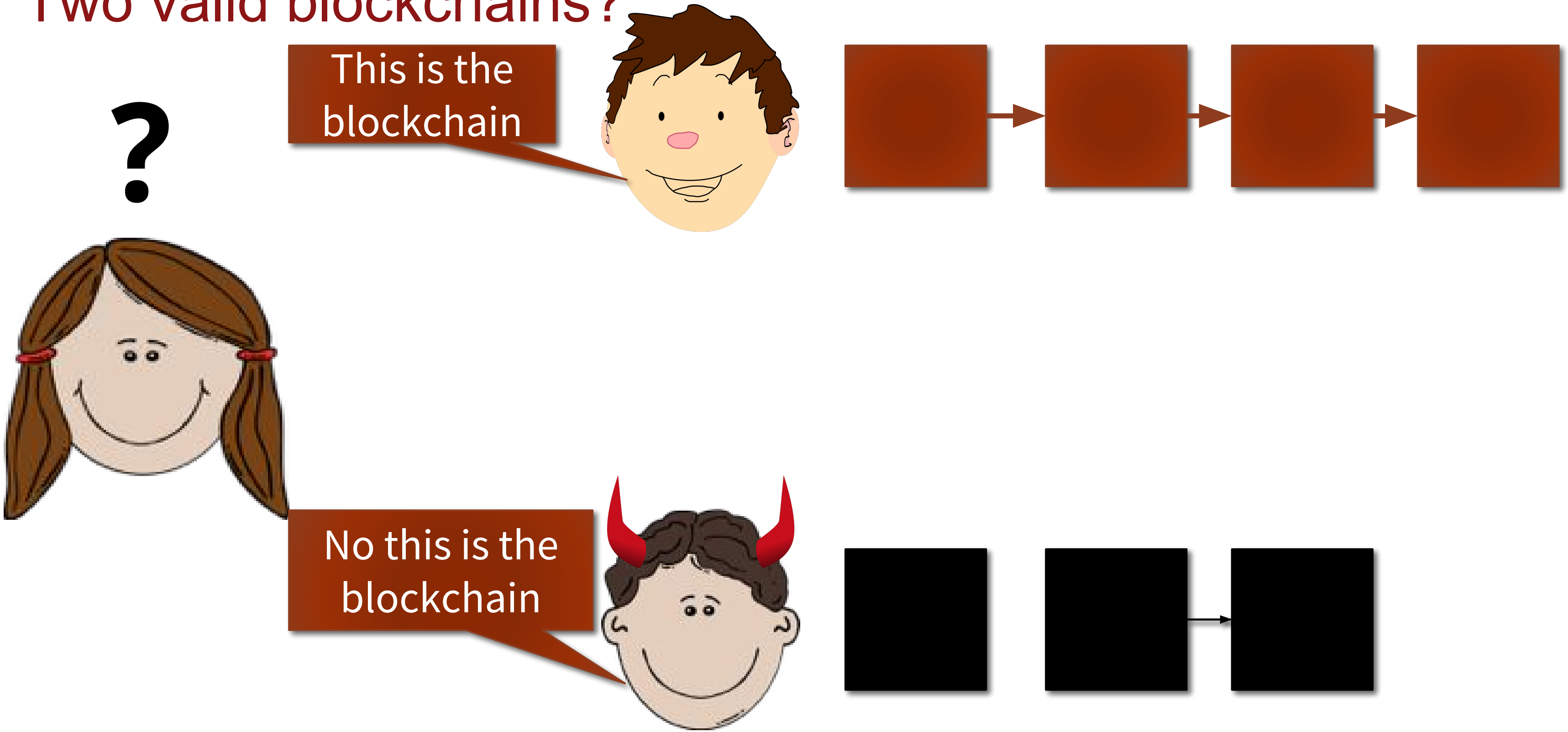
3. Validity of a block header



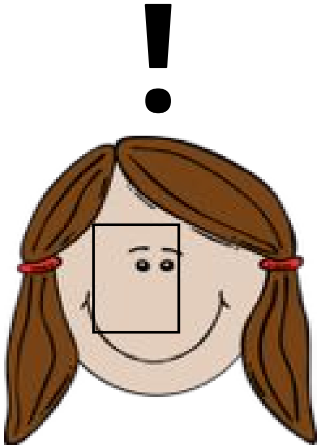
≤ 00000000000000001FB8930000000000000000000000000000000000000000000000

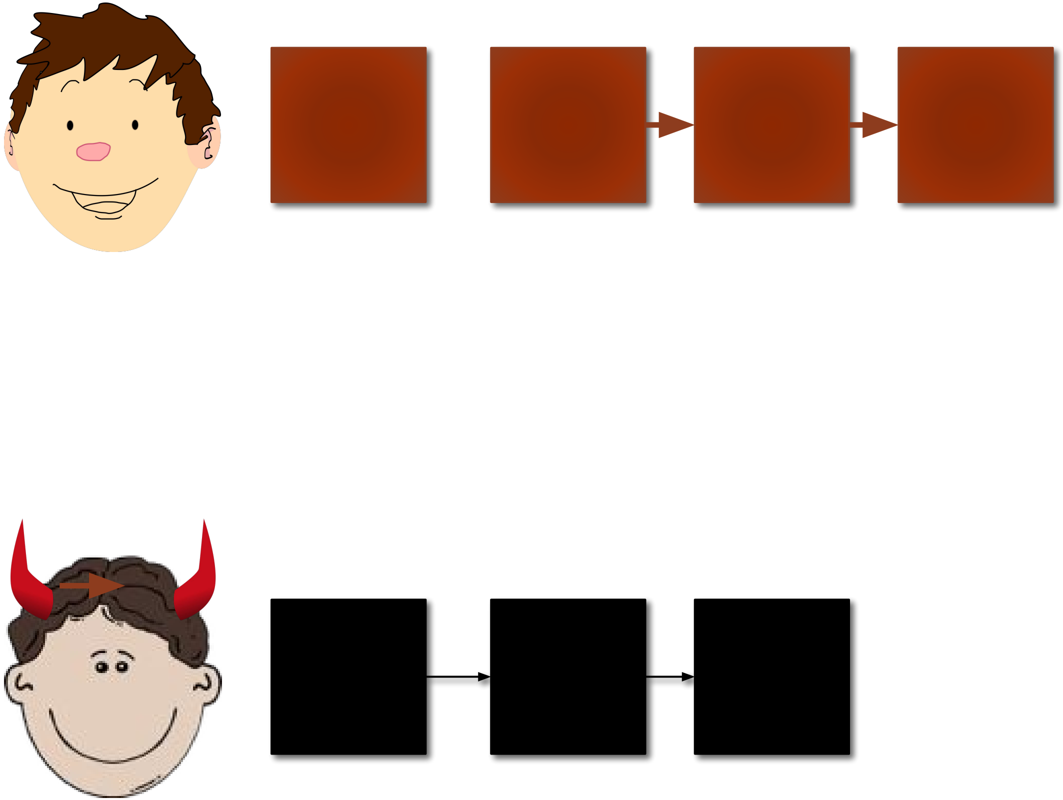
70+ leading zeroes required… ✓

Two valid blockchains?



Longest chain rule





Take the longest chain!
Harder to produce.

Proof of work conjecture

- Honest mining is a dominant equilibrium strategy
- The majority of miners act rational

- Implies that longest chain follows the rules of the network

- Sleeping beauty property: You can always distinguish honest and honest chains after being offline

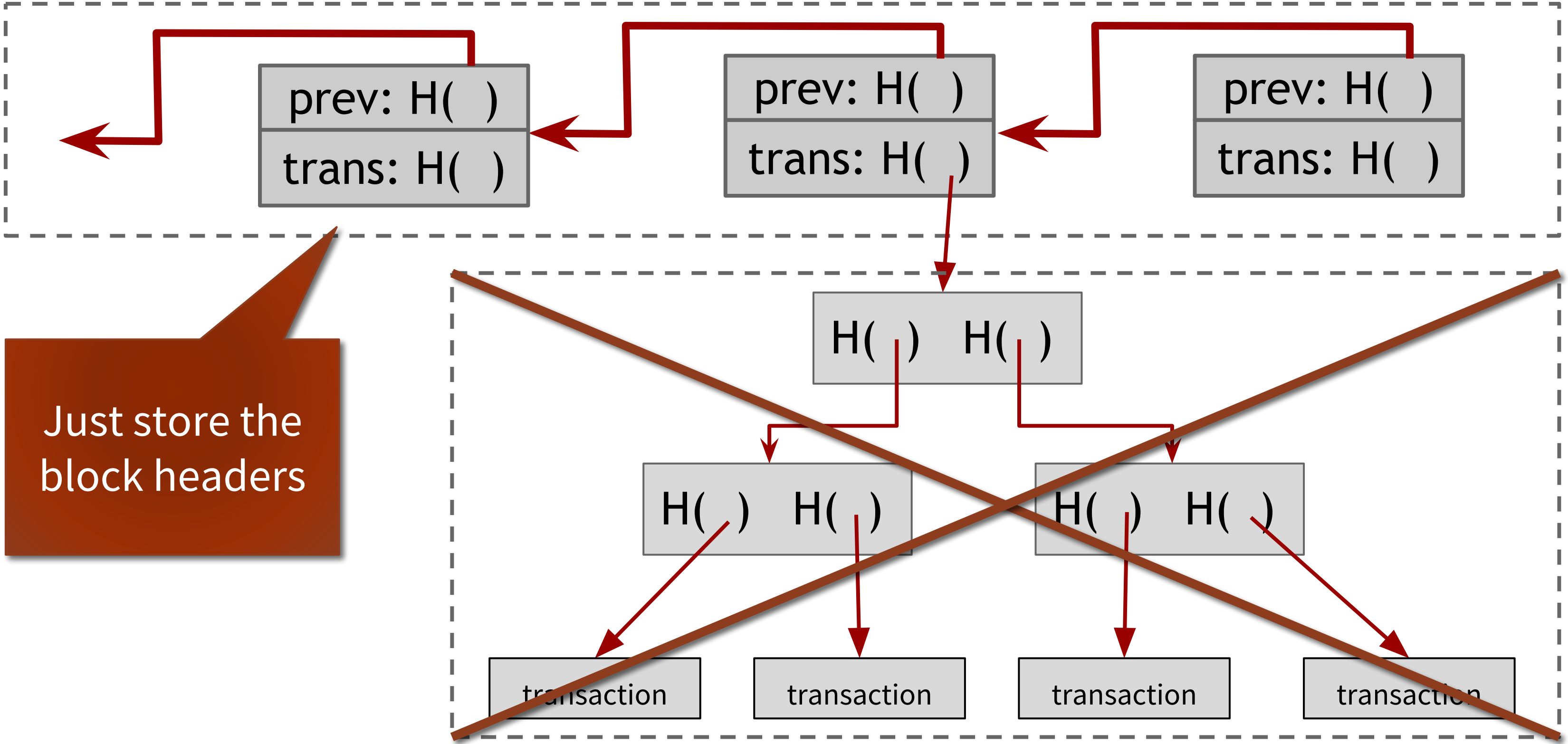
- Does not (necessarily) hold for proof of stake

- As long as one of the nodes you are connected to is honest you will find the best chain

Blockchain size: A growing problem

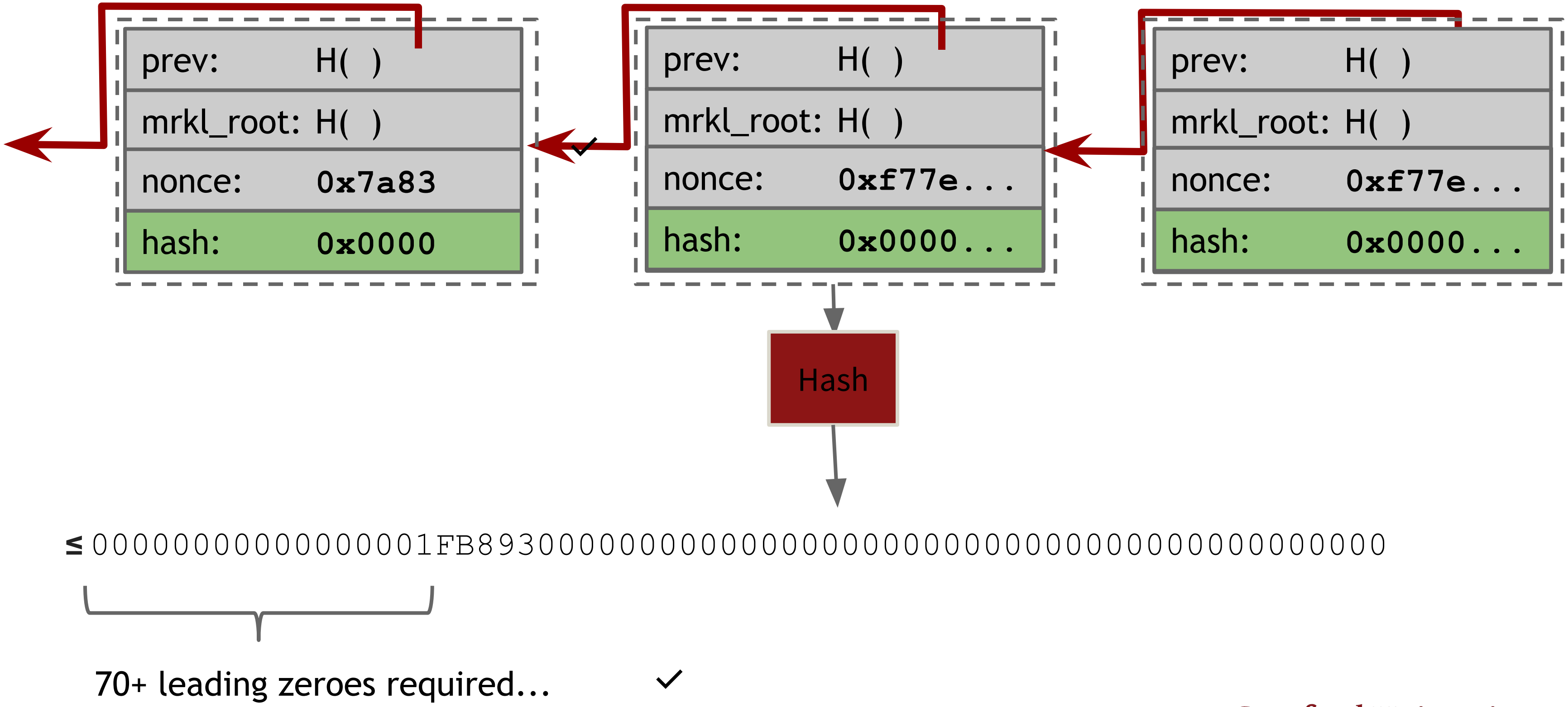
Stanford University

Simple Payment Verifying Client (Satoshi 2008)

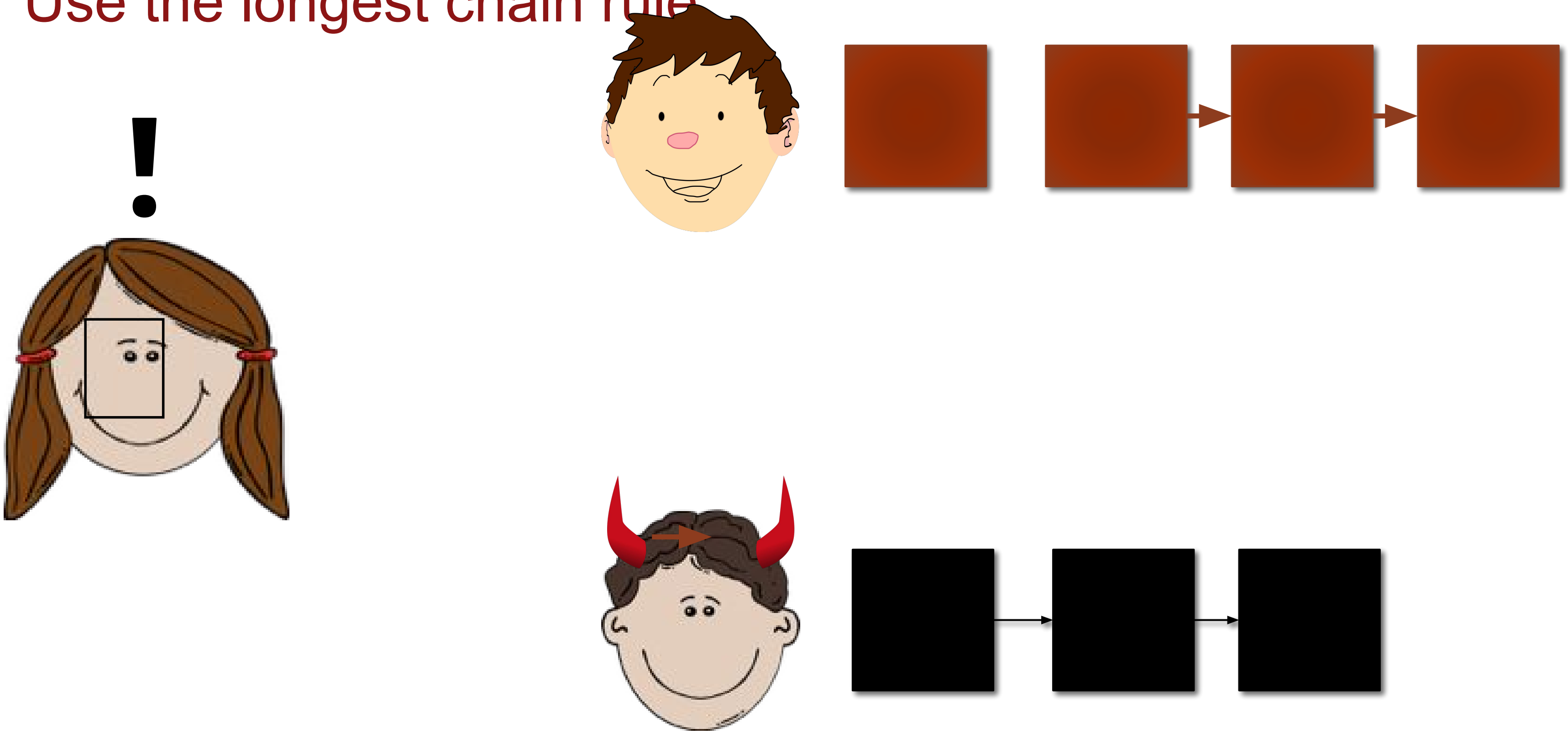


Just store the block headers

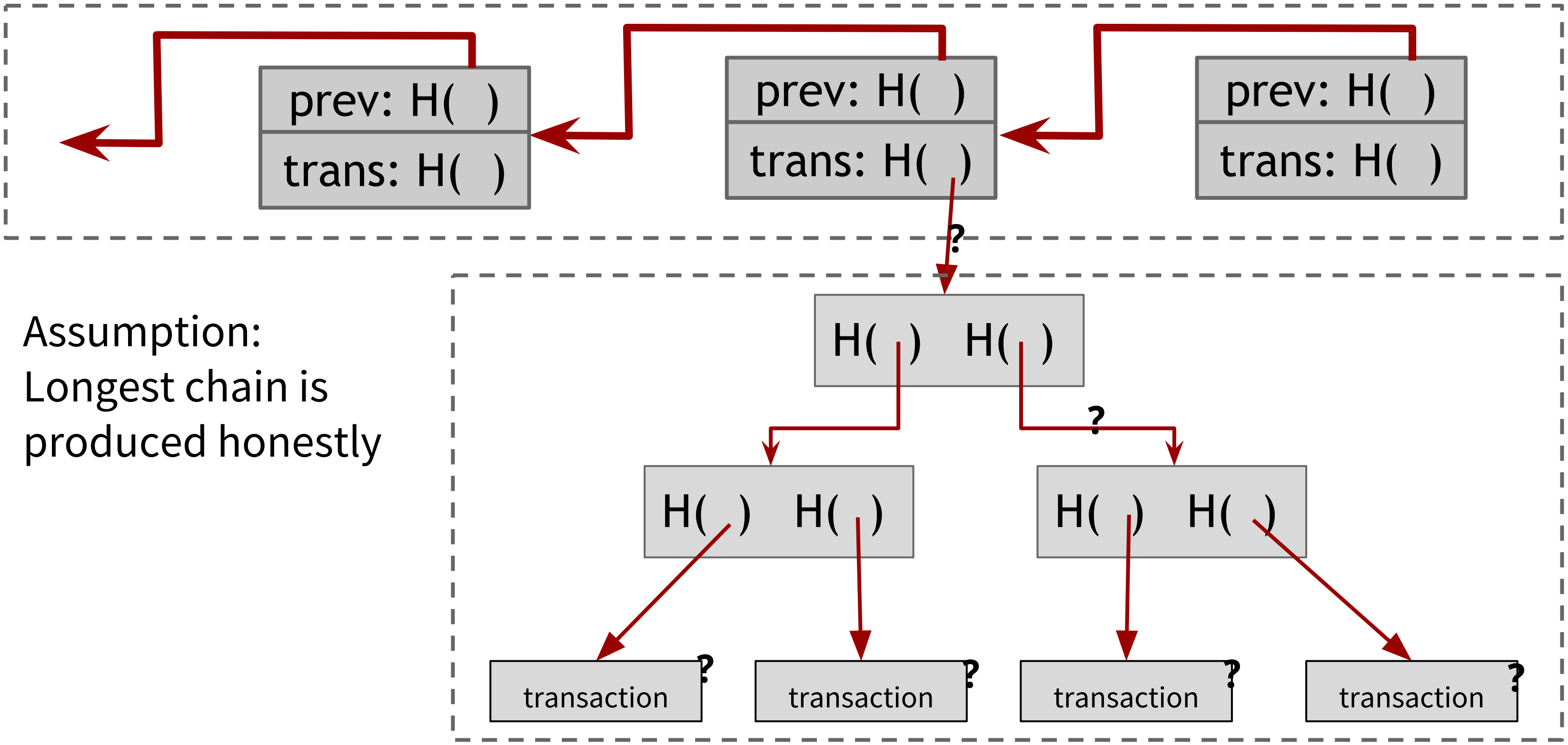
Verify block headers



Use the longest chain rule

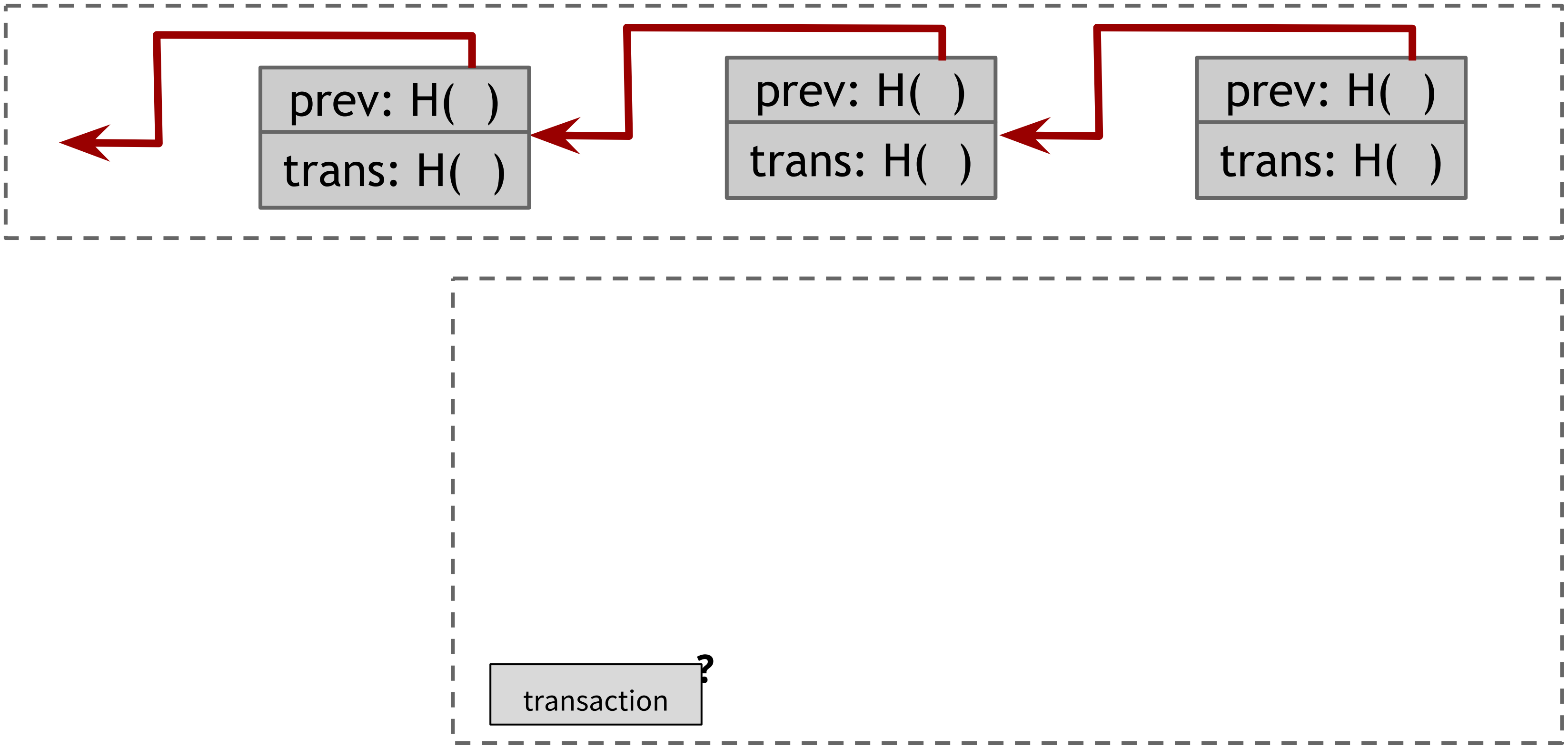


Can’t verify all transactions (but that’s ok)

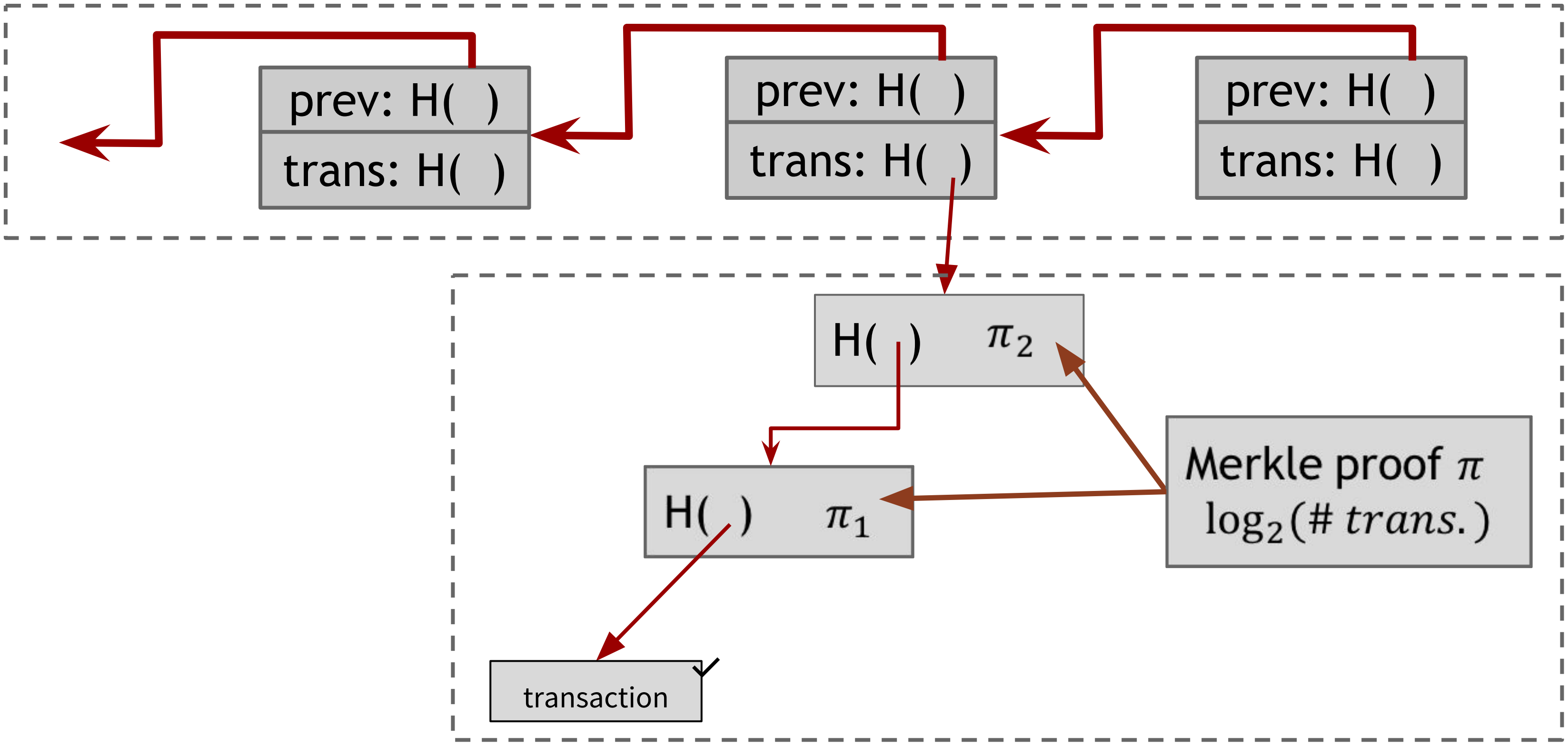


Assumption:
Longest chain is
produced honestly

Can verify specific transactions (with help)



Can verify specific transactions (with help)



Stanford University

SPV Properties and Problems

- Can determine the longest chain
- Can verify transaction inclusion
- Does not grow with #transactions
- 80 bytes * #blocks (Bitcoin)
- 508 bytes * #blocks (Ethereum)
- Sufficient for sidechains and swaps

- Can't verify all transactions
- Grows with #blocks
- Less block time-> larger SPV client
- 40 MB in Bitcoin
- 2.2 GB in Ethereum
- Especially bad for multi-chain clients

Sublinear SPV-Clients: SNARKs

- SNARK or CS-Proof/CIP/STARK (Micali 91, Ben-Sasson et al. 17)
  - Constant size non-interactive proof that chain has length X
  - Circuit verifies full blockchain
  - Not practical for prover
  - SNARKs closer to being practical but trusted setup

Sublinear SPV-Clients: NiPoPoWs

- Kyriasis, Miller, Zindros 17
- Based on Kiayias, Lamprou, Stouka 16 and Back et al. 14

- Insight: If I want to find x such that H(x) has n 0s then I will find 2 x’ such that H(x’) has n-1 0s, 4 x’’ such that H(x’’) has n-2 0s …

- Best quality proof of work indicates quality of whole chain

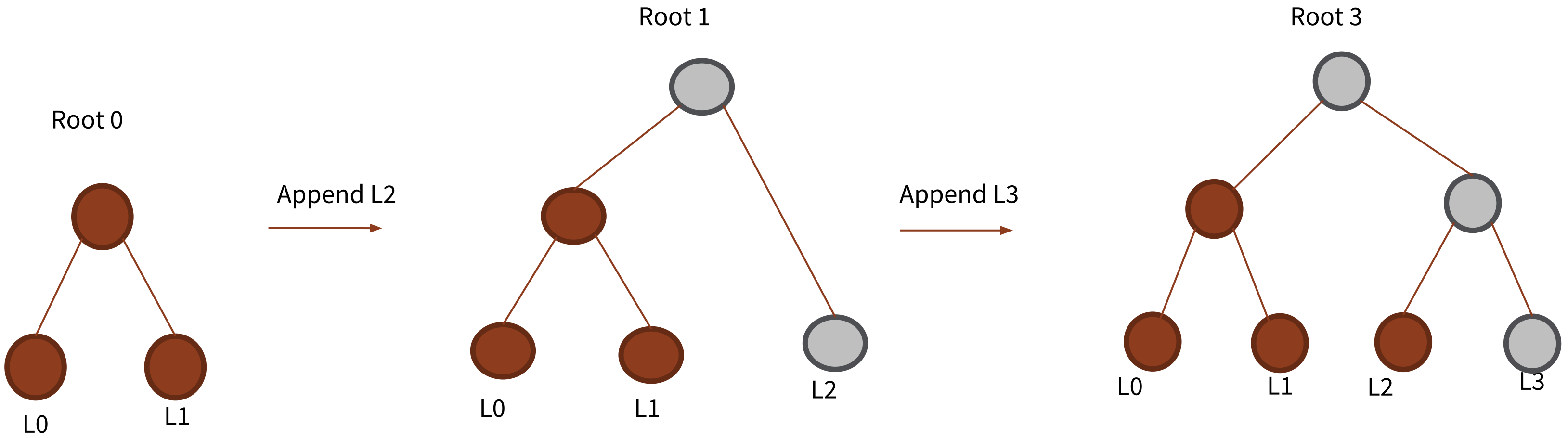
- Use a skiplist to point to proofs with less proofs of work

- O(log(n)*log(log(n))) proof size

NiPoPoW bribery attack

- High quality blocks do not give extra reward
- But they are important for NiPoPows[1]
- Bribe ~~honest~~ rational miners to throw away super high quality blocks
- Main chain “looks” worse which makes fooling SPV client easier
- Does not violate NiPoPow’s security proof because honest mining and not rational mining is assumed
- Motivates search for different NiPoPows

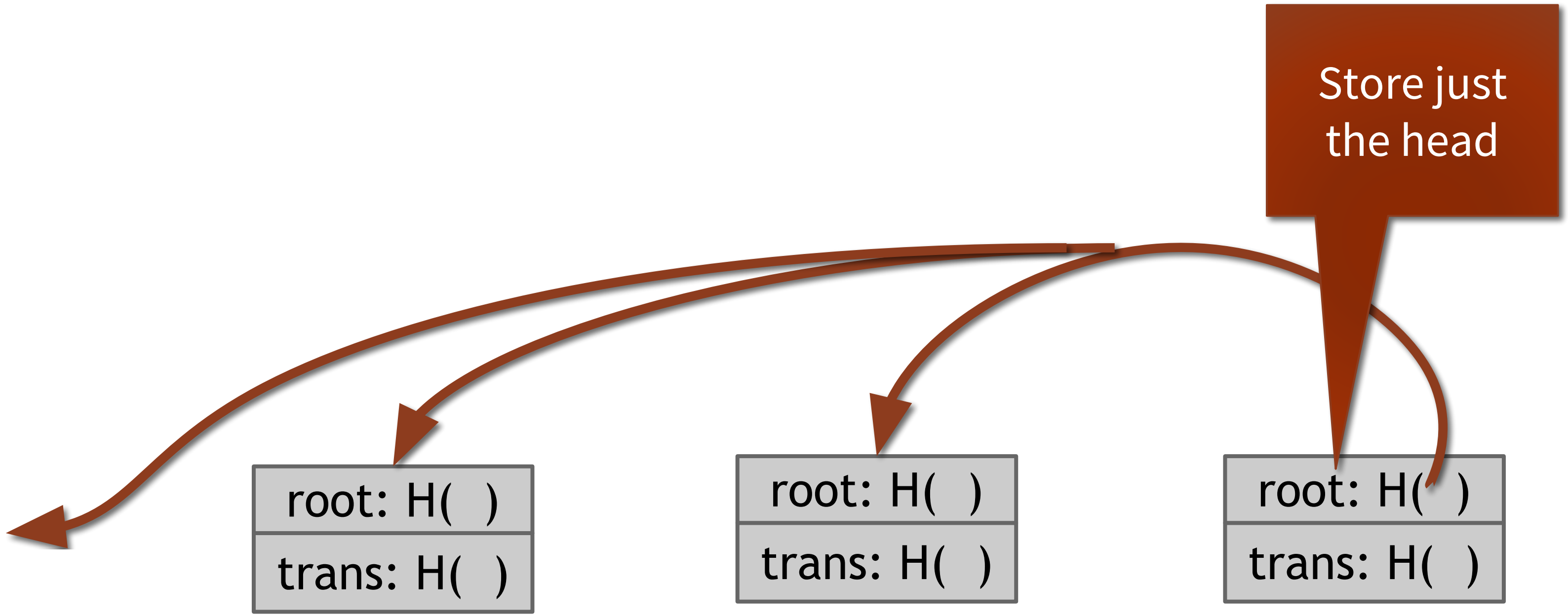
Merkle Mountain Ranges (Todd 16)



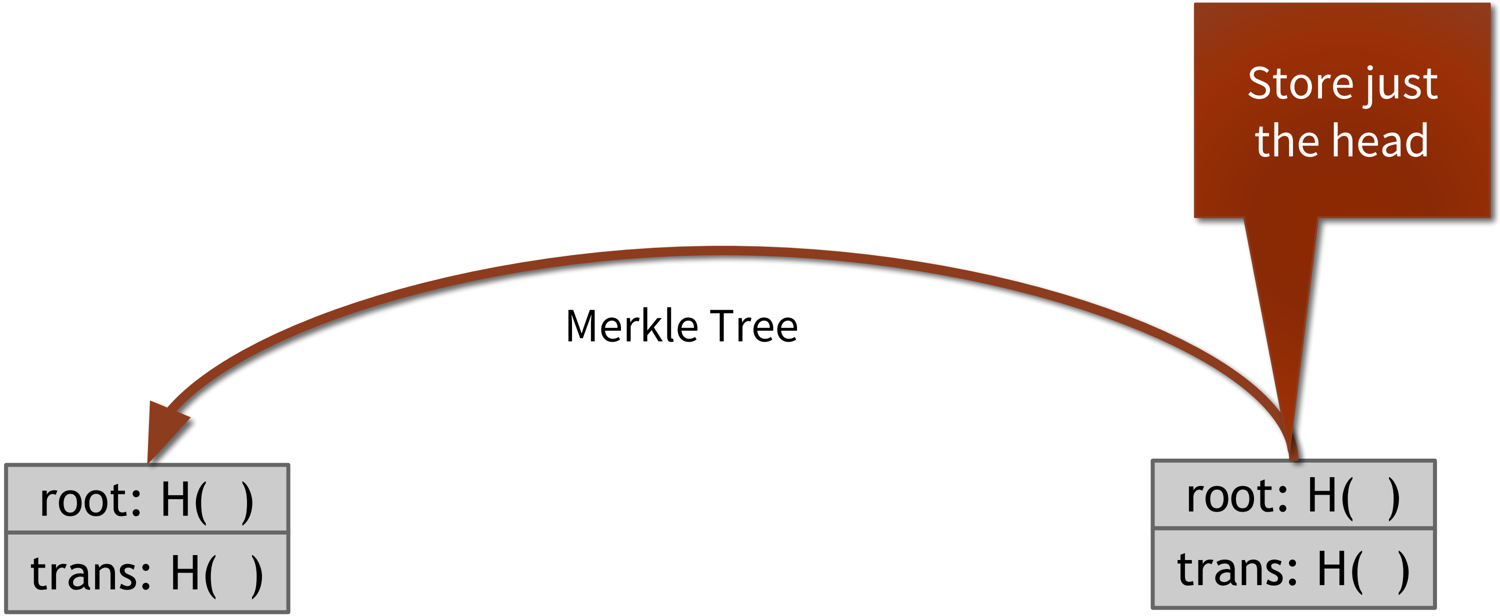
Log(n) inclusion proofs
Log(n) updates
nth tree commits to kth tree k<n
Log(n) difference proofs

Flyclient: A different approach to super-light clients

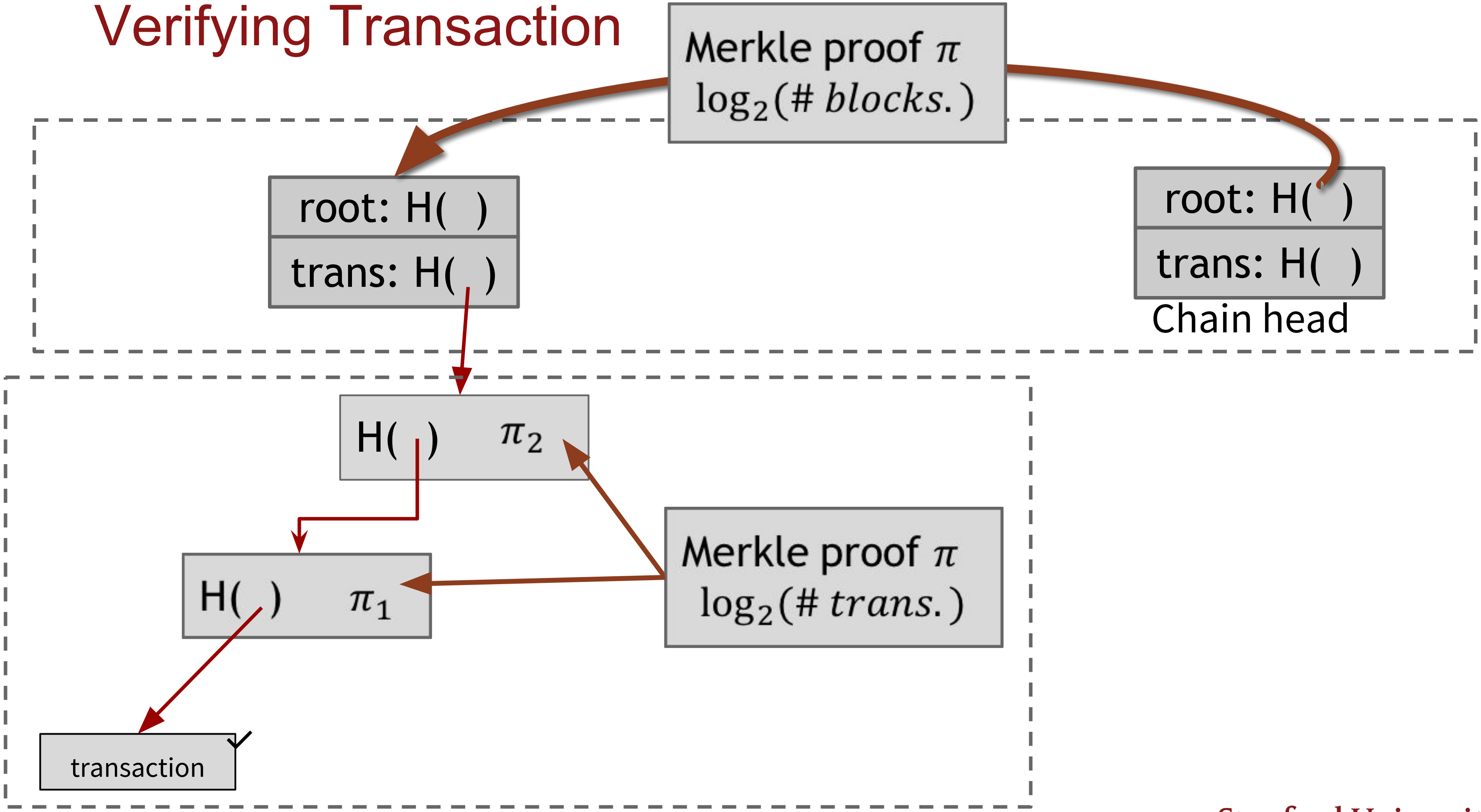
Flyclient: A different approach to super-light clients



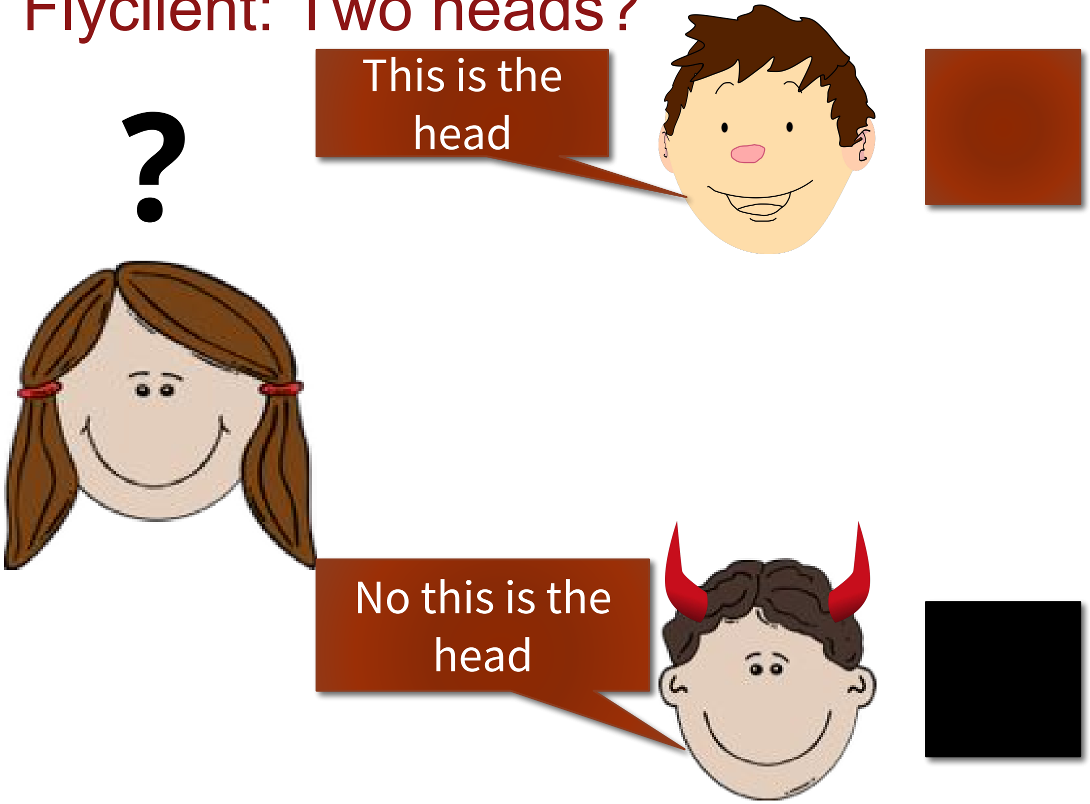
Flyclient: A different approach to super-light clients



Verifying Transaction



Flyclient: Two heads?

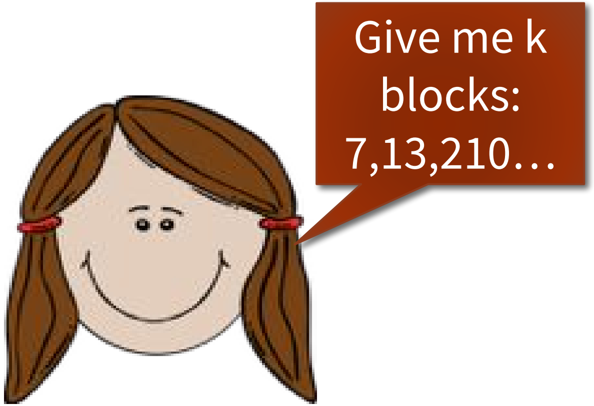


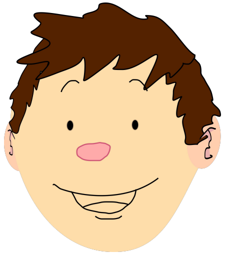
Assumption:
At least one chain is honest

Other one has at most a c fraction of the mining power
Ex: c=1/3

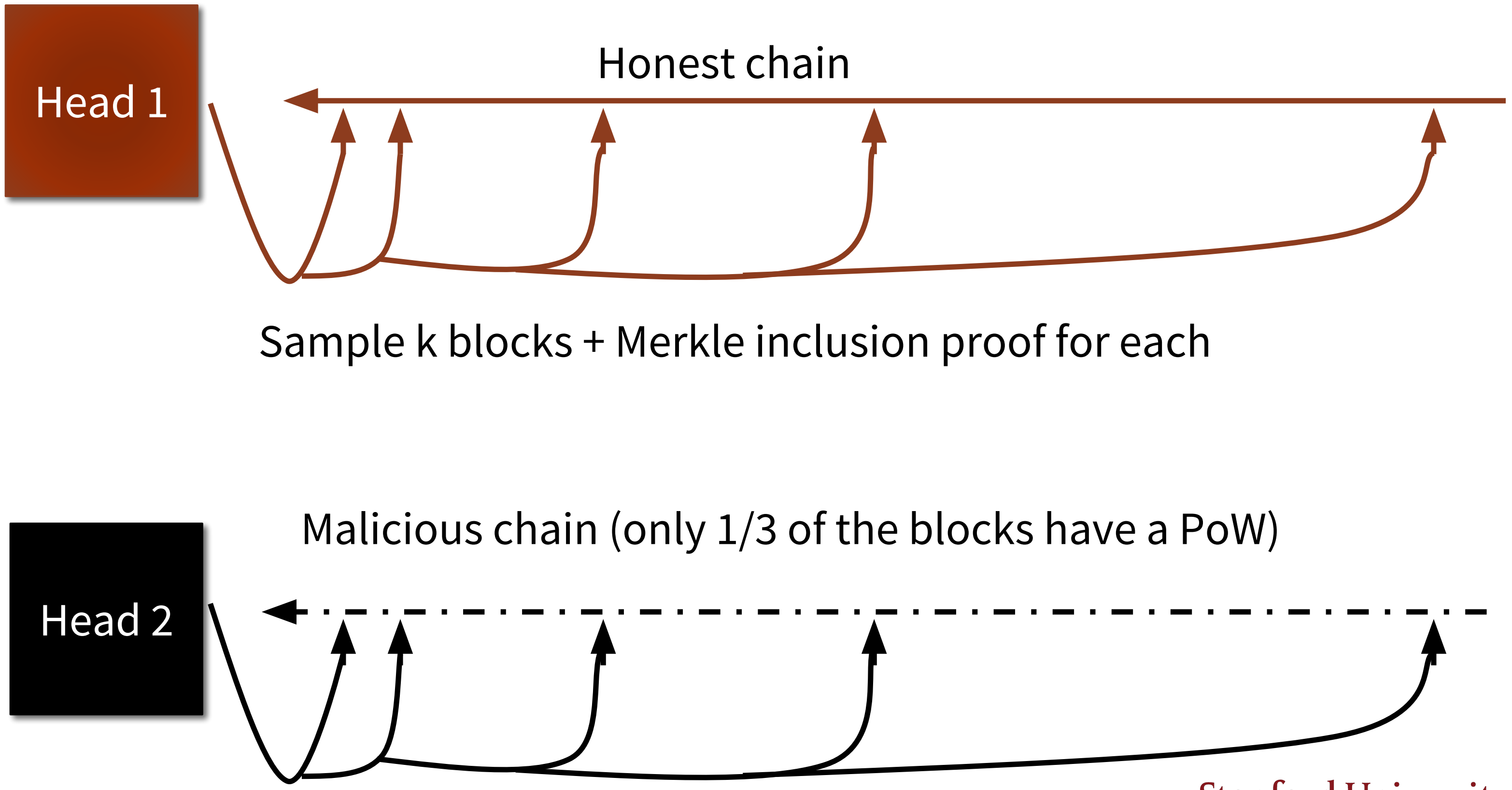
Stanford University

Flyclient Strawman 1

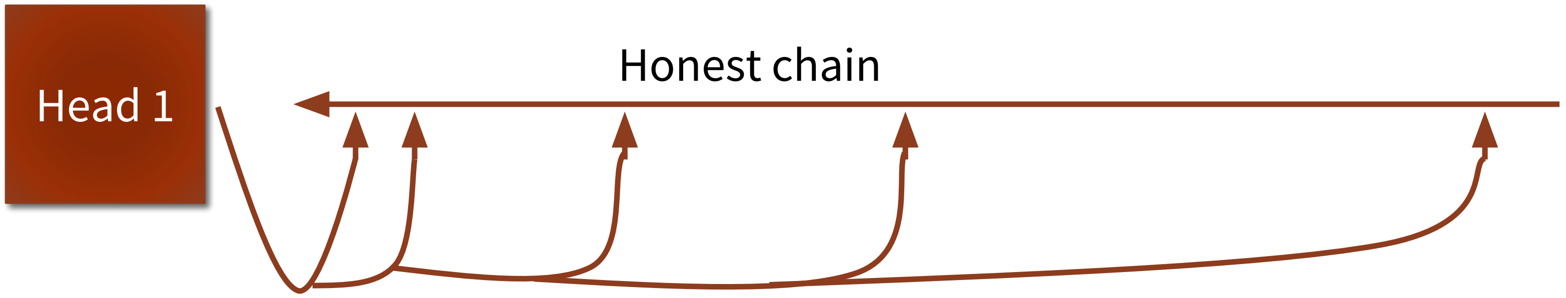




Flyclient Strawman 1: sample constant # of blocks

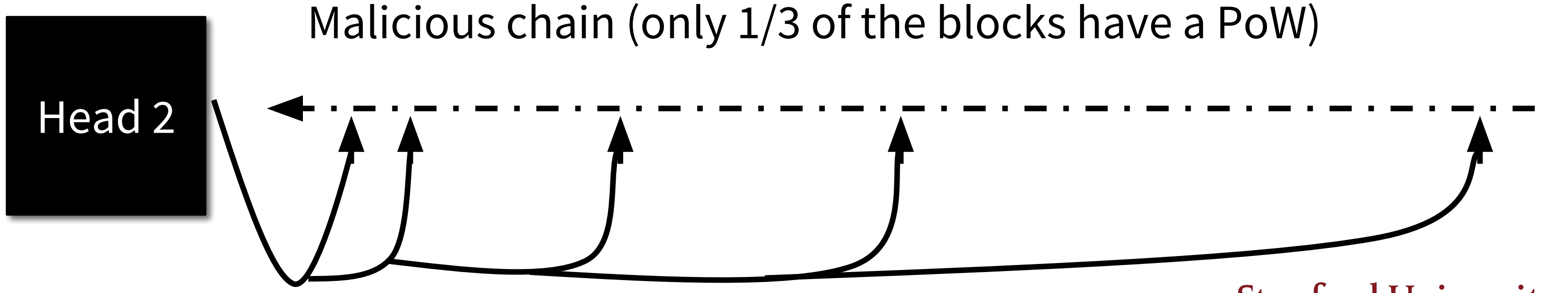


Flyclient Strawman 1: sample constant # of blocks

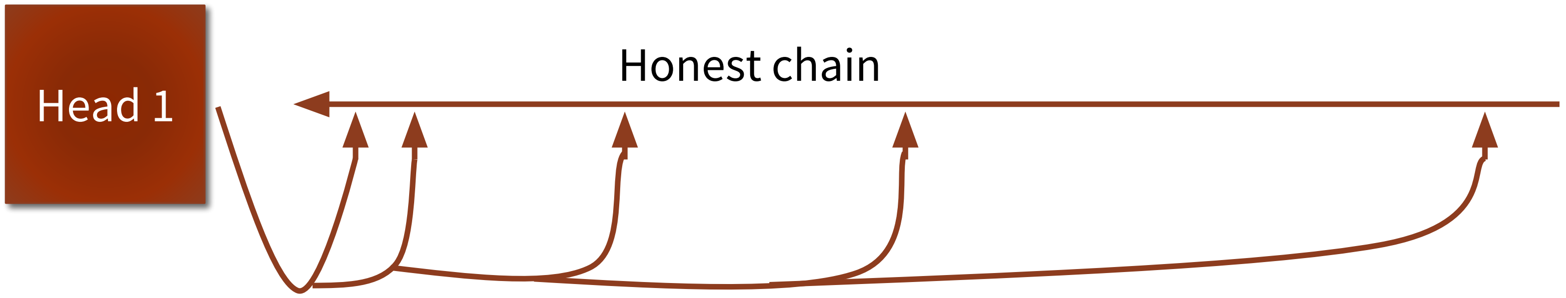


k samples and $\frac{1}{3}$ mining power →Advers. wins with $\frac{1}{3}^k$

$$k = 81 \rightarrow P[Cheating] < 2^{-128}$$

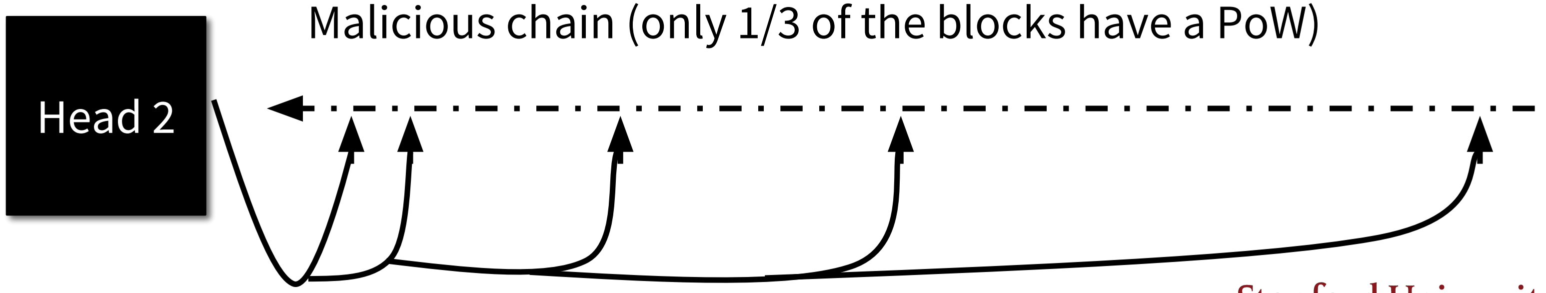


Flyclient Strawman 1: sample constant # of blocks

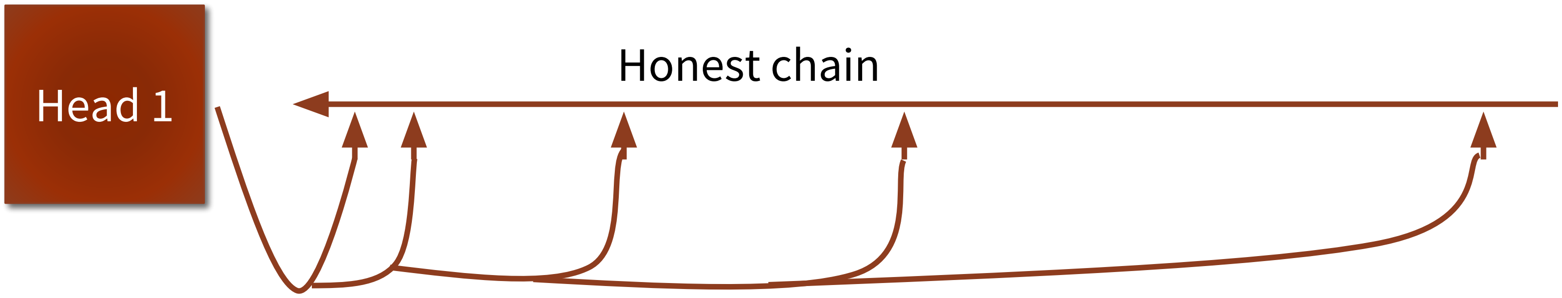


k samples and $\frac{1}{3}$ mining power →Advers. wins with $\frac{1}{3}^k$

$$k = 81 \rightarrow P[Cheating] < 2^{-128}$$

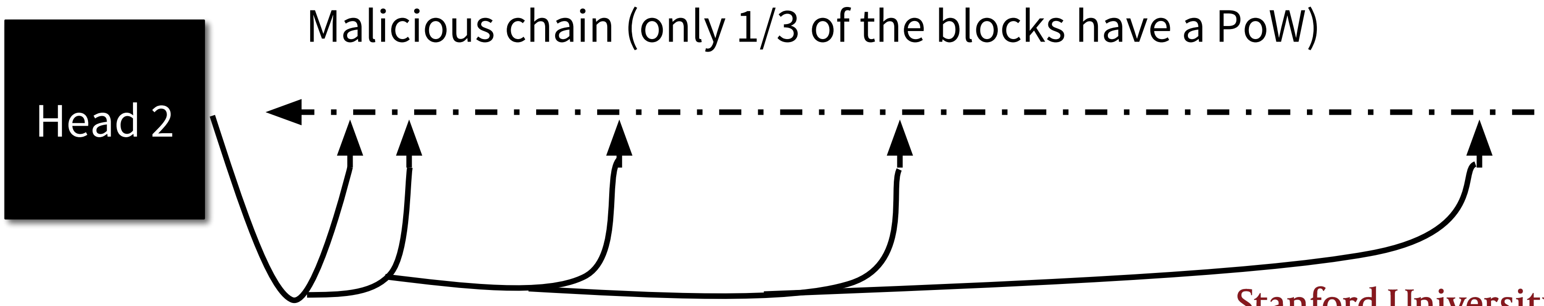


Flyclient Strawman 1: sample constant # of blocks



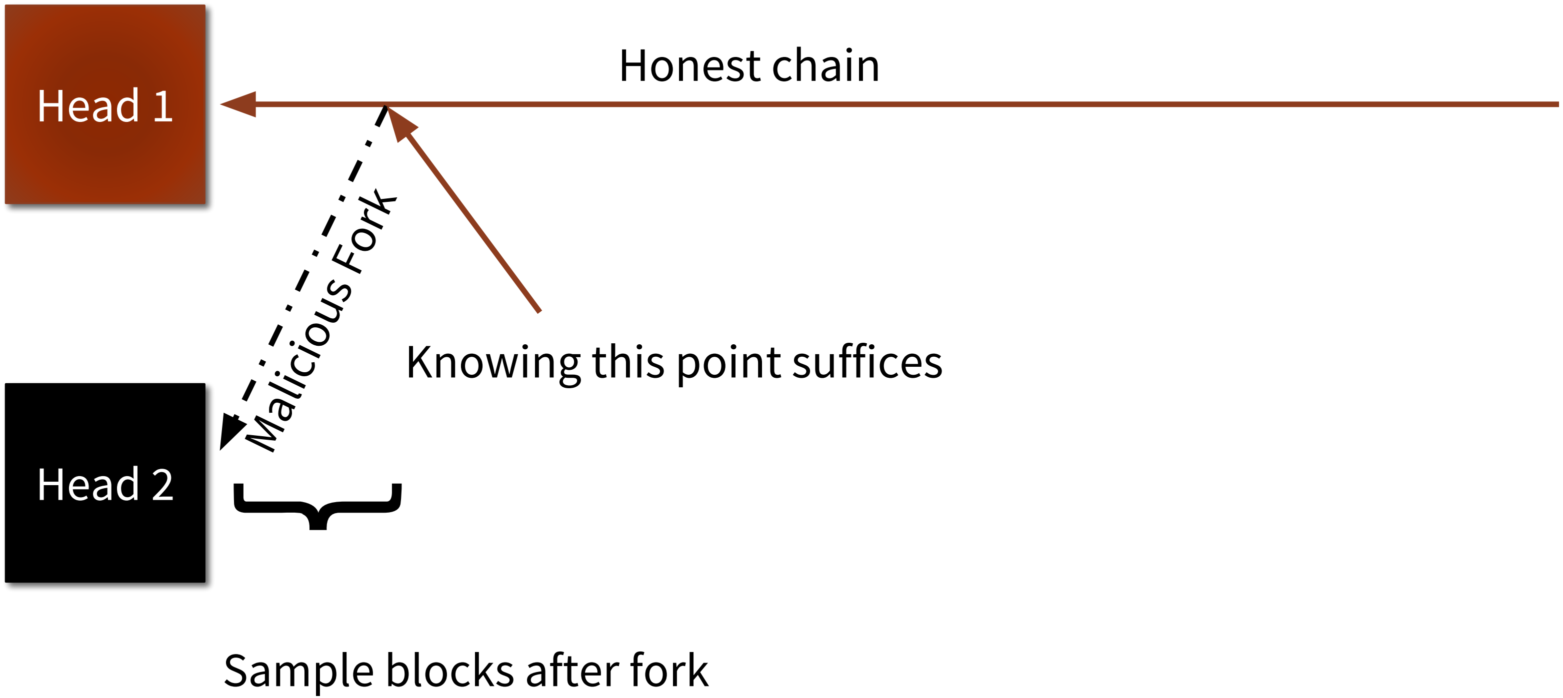
k samples and $\frac{1}{3}$ mining power →Advers. wins with $\frac{1}{3}^k$

$$k = 81 \rightarrow P[Cheating] < 2^{-128}$$

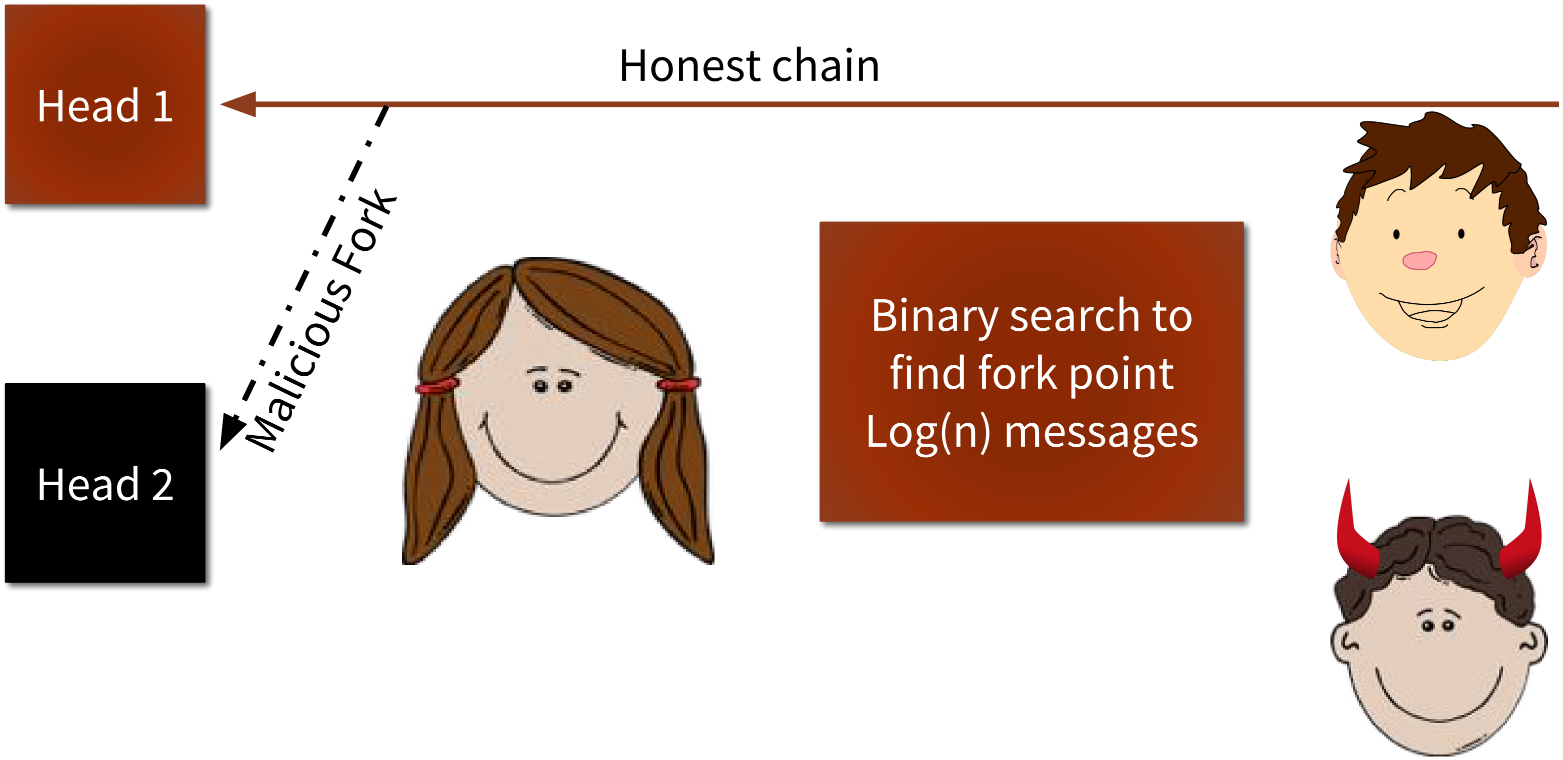


Flyclient Strawman 1 problem: Forking

Flyclient idea: Find Fork Point



Flyclient Strawman 2: Interactive Binary Search



Flyclient Strawman 2: Interactive Binary Search

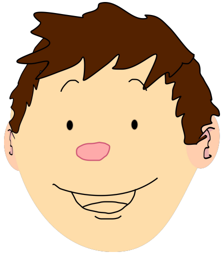
Head 1

Honest chain

Malicious Fork

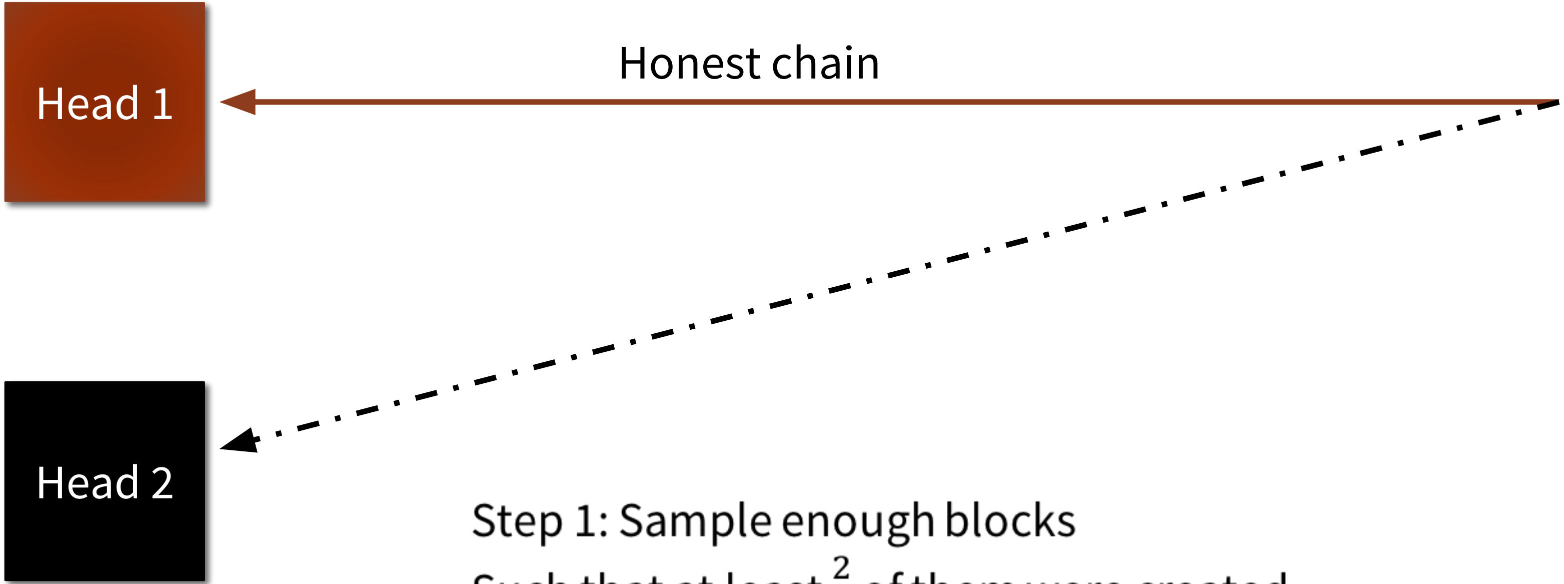
Head 2

Binary search to
find fork point
Log(n) messages



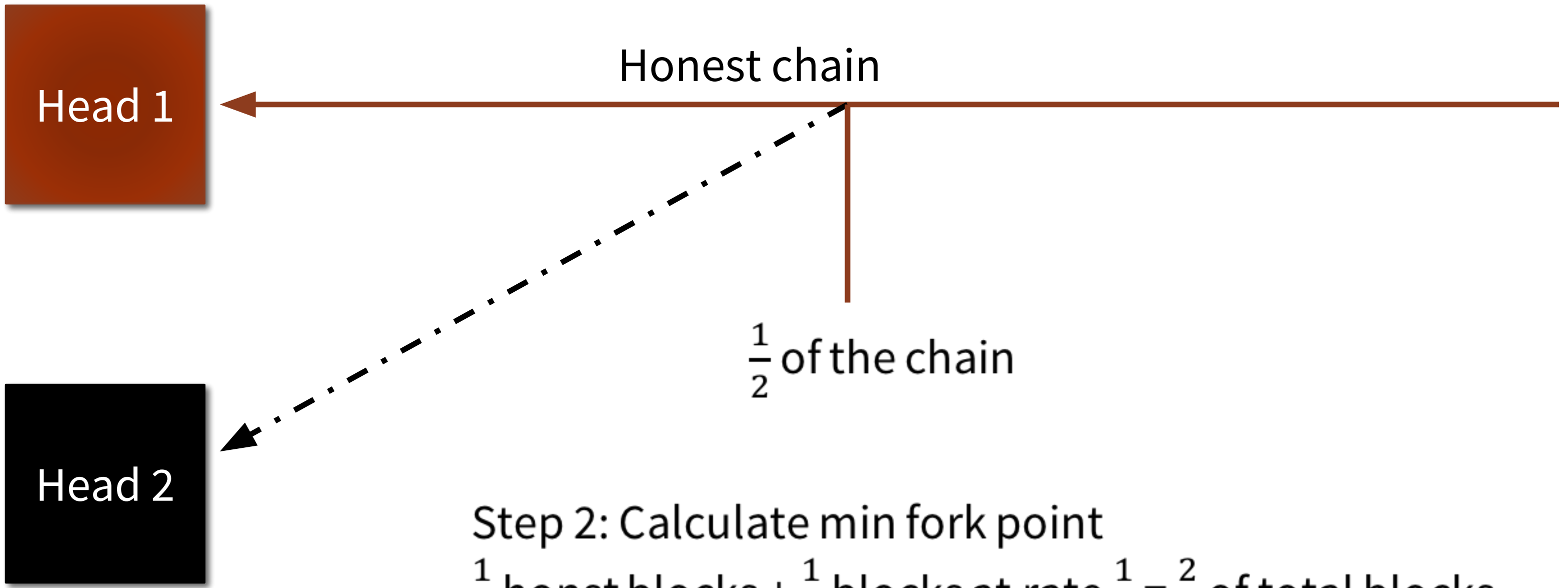
Works but two provers may
not want to interact

Flyclient: Idea bound forking point



Step 1: Sample enough blocks
Such that at least $\frac{2}{3}$ of them were created

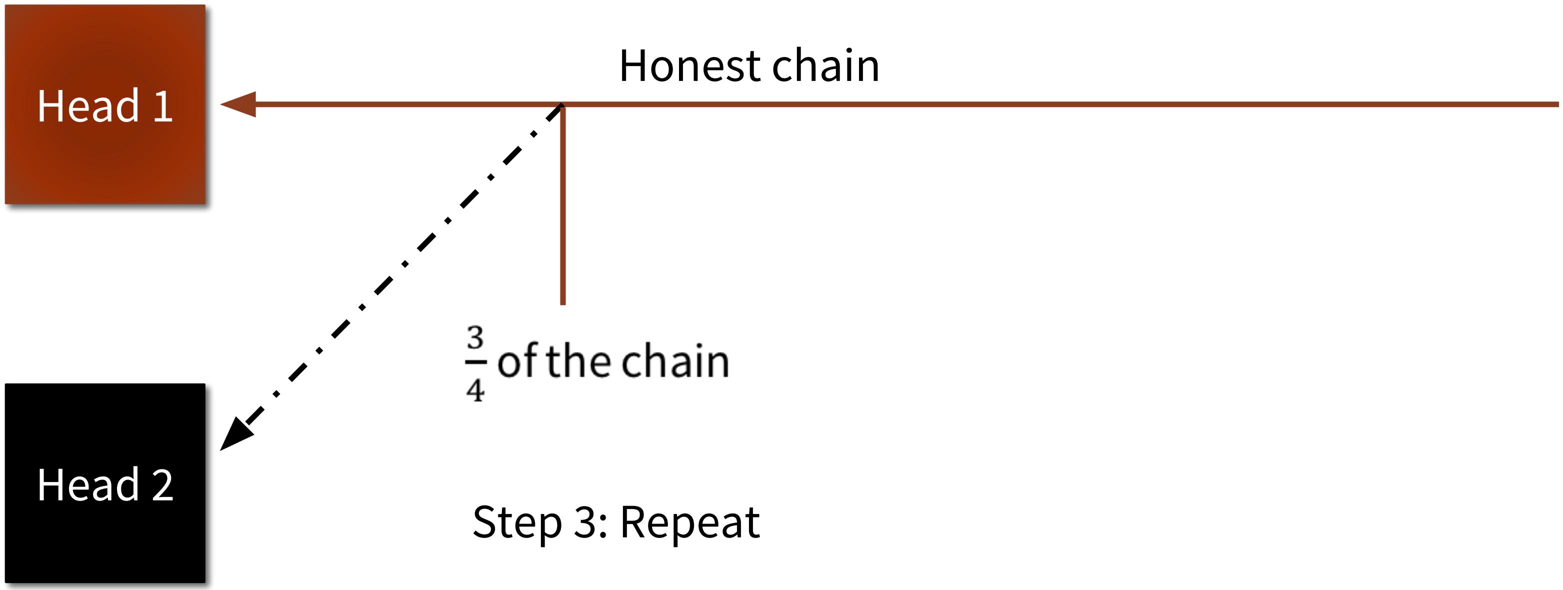
Flyclient: Idea bound forking point



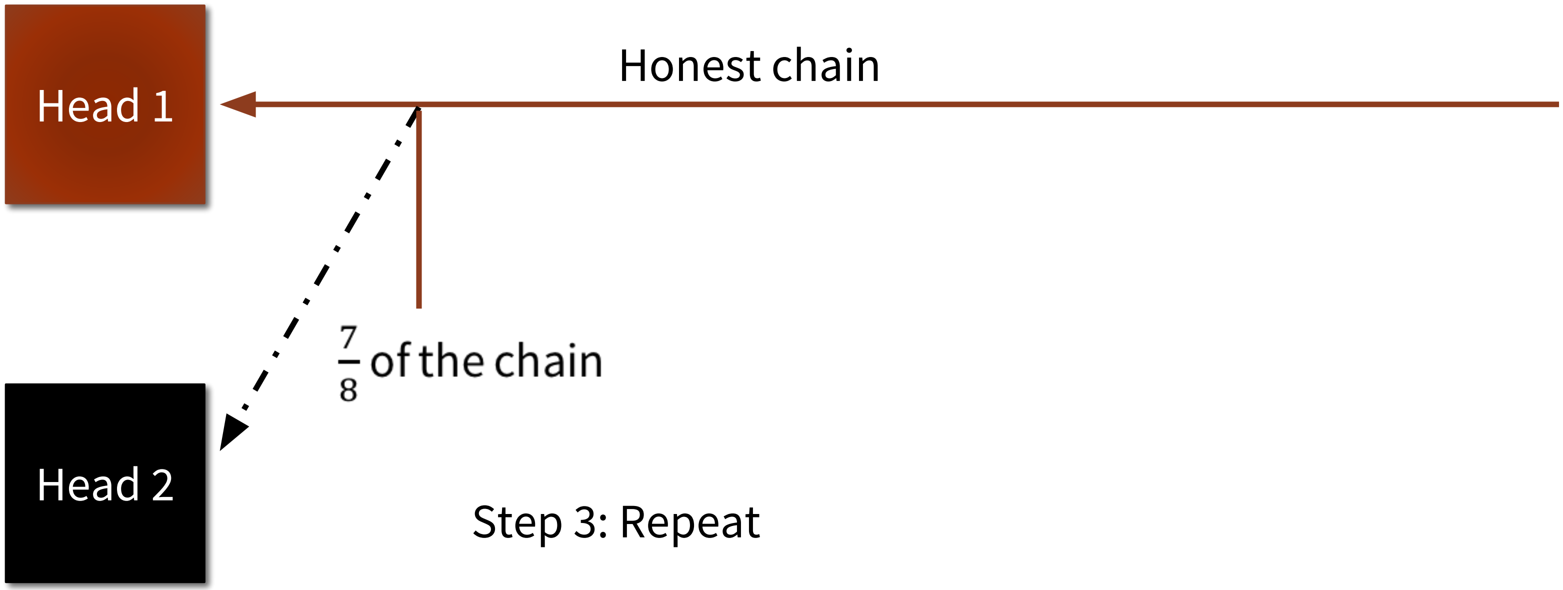
Step 2: Calculate min fork point

$\frac{1}{2}$ honst blocks + $\frac{1}{2}$ blocks at rate $\frac{1}{3} = \frac{2}{3}$ of total blocks

Flyclient: Idea bound forking point

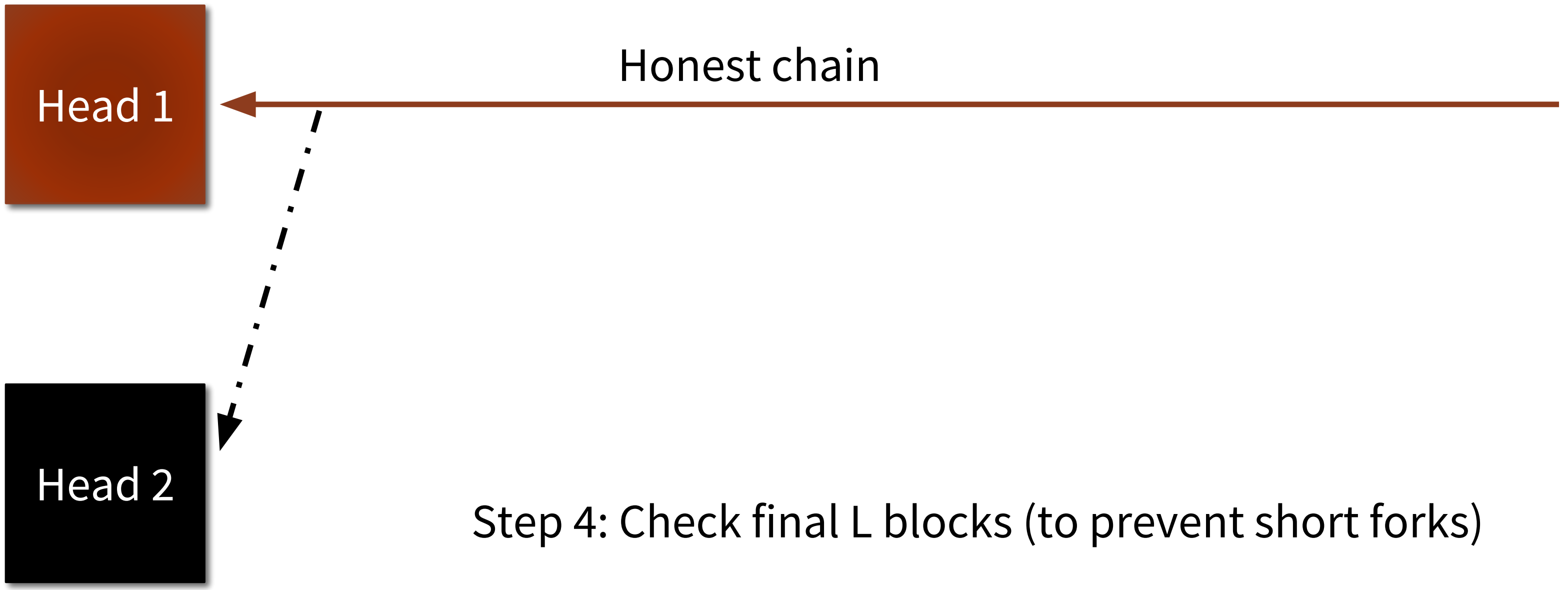


Flyclient: Idea bound forking point



Step 3: Repeat

Flyclient: Idea bound forking point



Step 4: Check final L blocks (to prevent short forks)

Flyclient Analysis

- In each interval check k blocks, k independent of chain length n
- k dependent on attacker strength
- Check log(n) intervalls
- For each block do log(n) merkle inclusion proof
- O(log(n)^2) overall
- For n=1000000->

Non Interactive Flyclient

- Verifier just requests random blocks
- Get randomness from hash function and chain head (Bonneau et al. 15)
- Also known as the Fiat-Shamir heuristic
- 2^{-128} soundness not needed because new hash -> new head -> new PoW
- Create proof once and reuse
- Simulation: <3 MB for Ethereum instead of 2.2GB

Thanks

BUENZ@CS.STANFORD.EDU

bbuenz.github.io