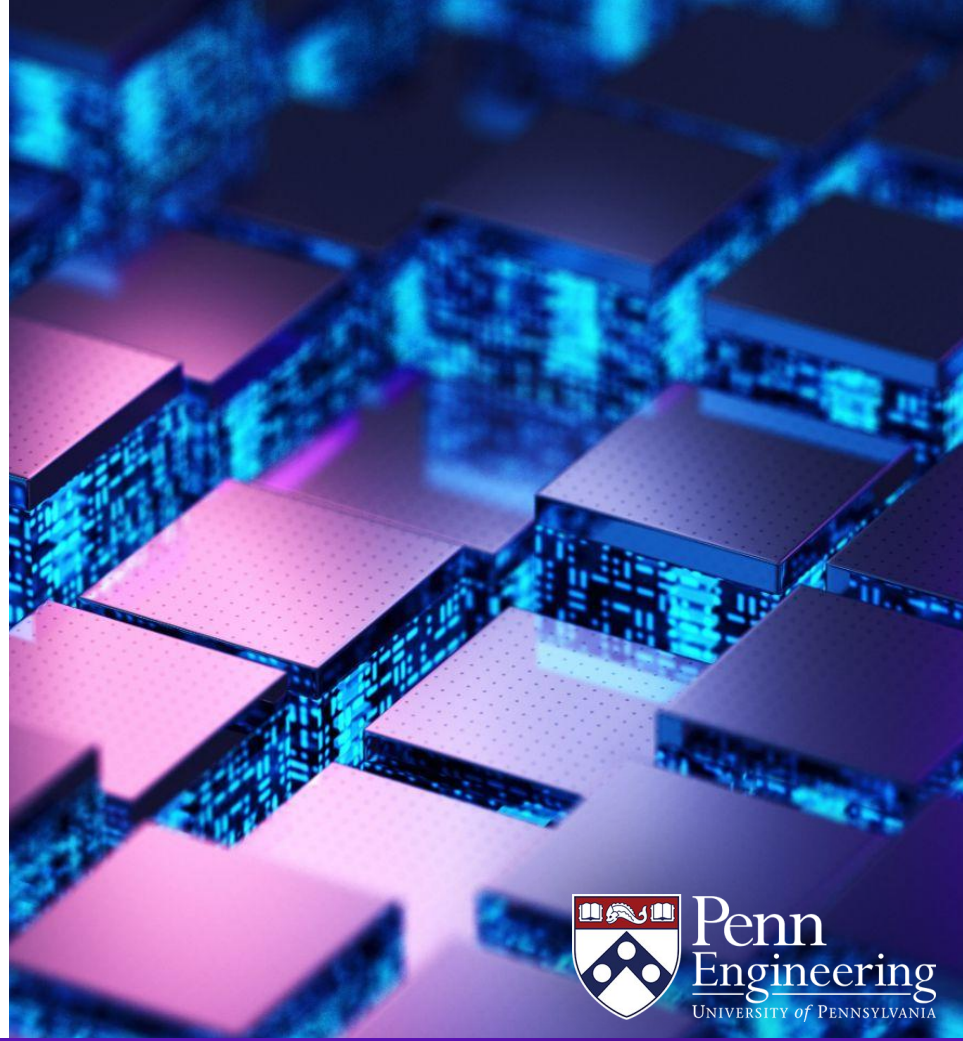
EAS 5830: BLOCKCHAINS

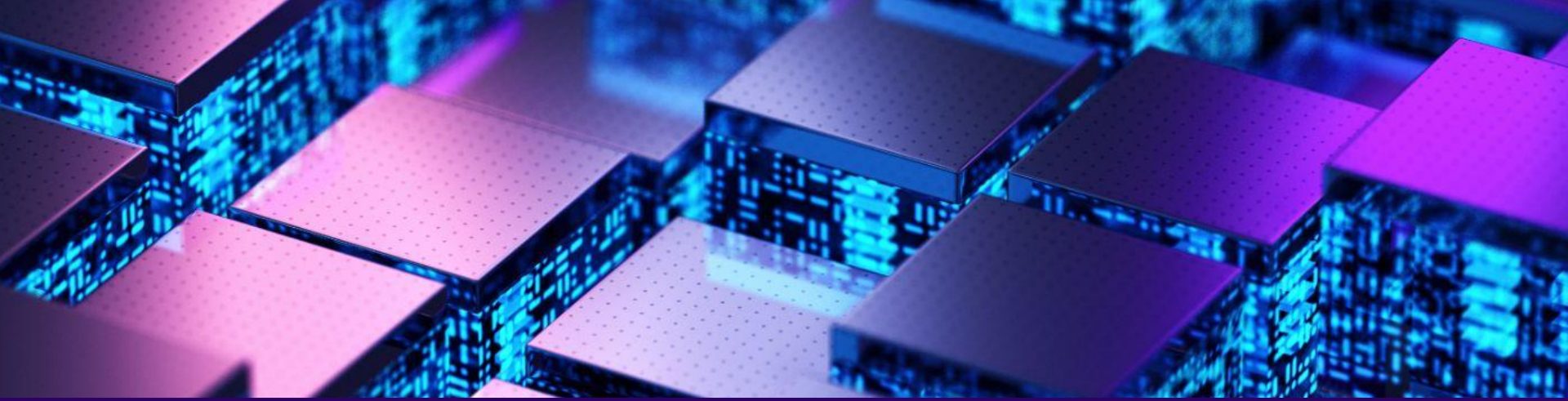
Choosing Block Producers in PoS Systems

Dr. Brett Hemenway Falk



Bitcoin

- Chance of becoming block producer is proportional to hash power
- Anyone can try to produce a block
- A bitcoin block is only “valid” if its hash is less than a “target” value



Lottery-based leader election

Leader election in Proof of Stake

- Chance of becoming leader is proportional to stake
  - "Follow the Satoshi"
  - Enumerate all minted coins (in the order they were minted)
  - Generate a pseudorandom number between 1 and the total number of coins
  - The owner of that coin gets to be the next block producer
  - **Caveat: generating randomness on the blockchain is hard**

Method 1: Hashing

- Hash the previous block
  - $H(B^{r-1})$ looks “random”

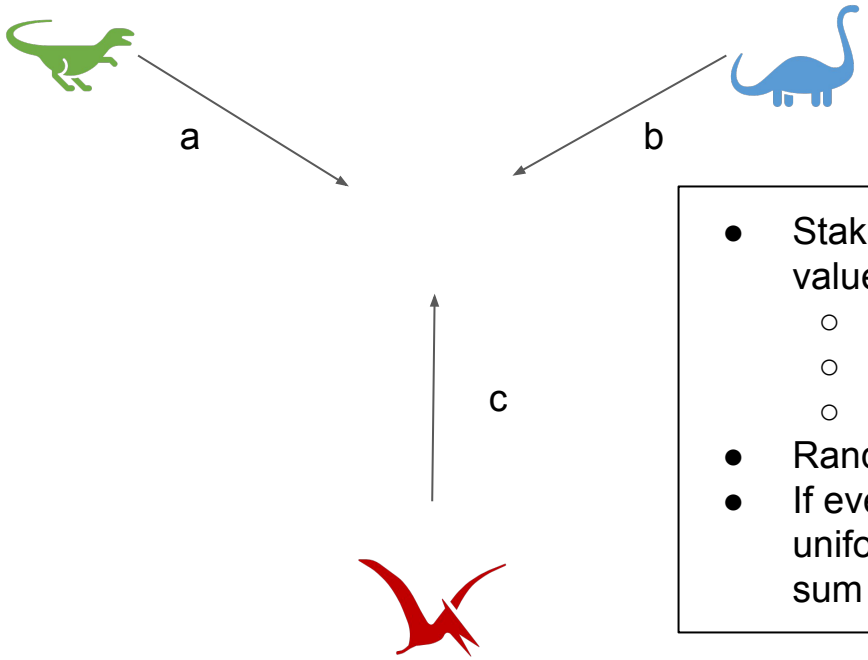
Method 1: Hashing

- Hash the previous block
  - $H(B^{r-1})$ looks "random"

Problem: Susceptible to "grinding" attacks

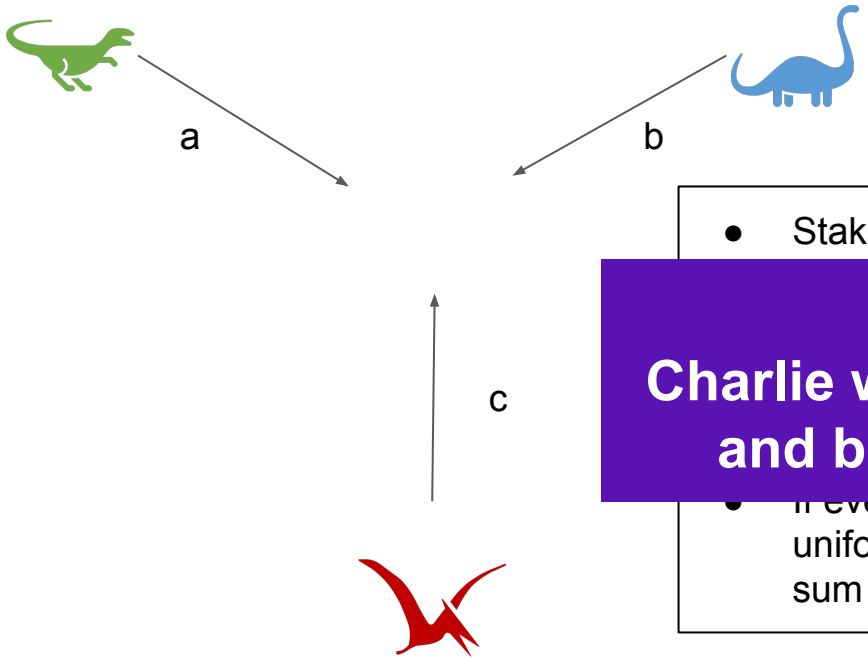
- Attacker can insert transactions into blocks to ensure $H(B^{r-1})$ chooses a coin they own
- Similar attacks on related deterministic schemes

Method 2: Commit-Reveal



- Stakers all submit a random value to the blockchain
  - Alice submits a
  - Bob submits b
  - Charlie submits c
- Randomness is a + b + c
- If even one of a,b,c was uniformly random, then so is the sum

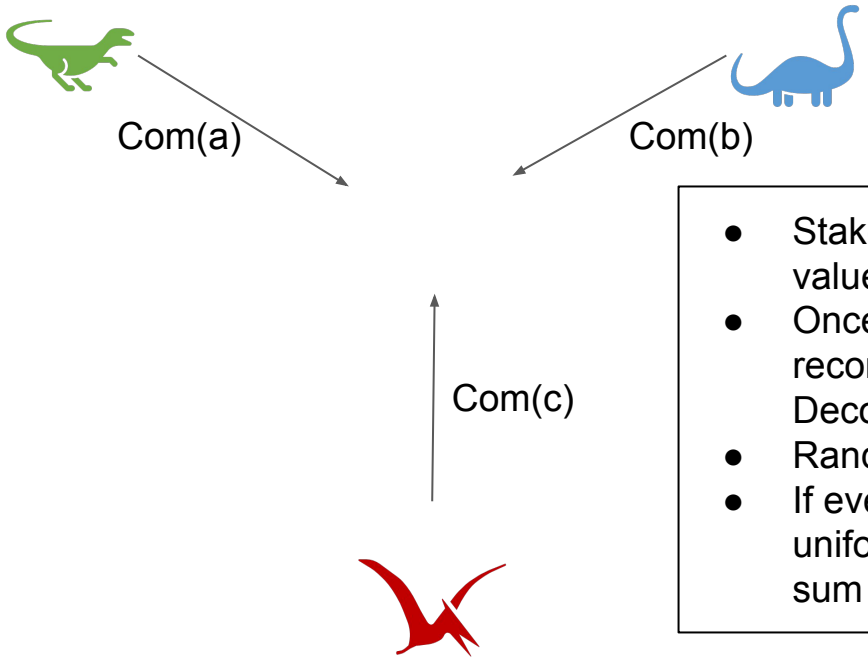
Method 2: Commit-Reveal



- Stakers all submit a random
- uniformly random, then so is the sum

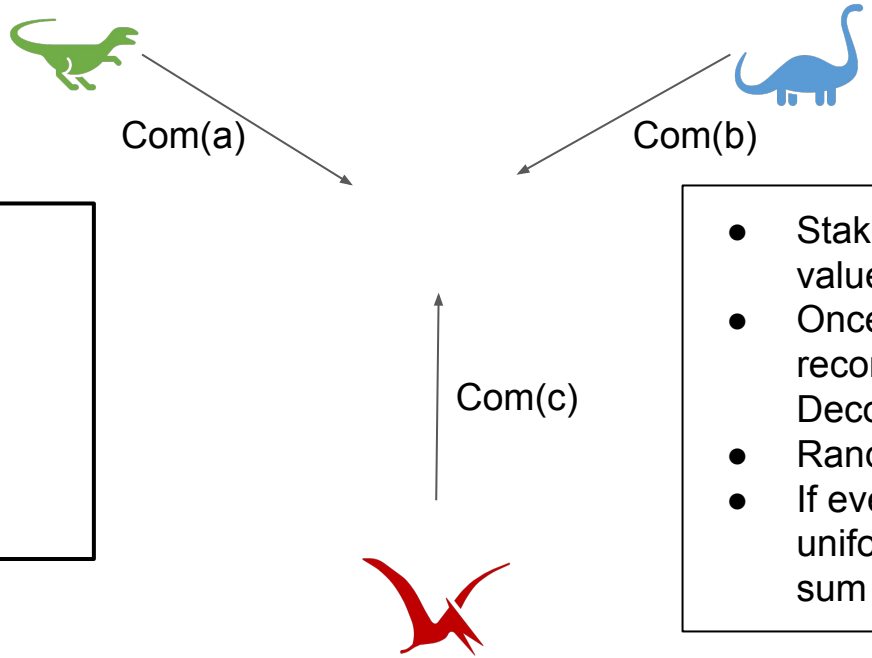
Problem:
Charlie waits until he sees a and b, then chooses c.

Method 2: Commit-Reveal



- Stakers Commit to their random values
- Once all commitments are recorded on chain Stakers Decommit
- Randomness is a + b + c
- If even one of a,b,c was uniformly random, then so is the sum

Method 2: Commit-Reveal



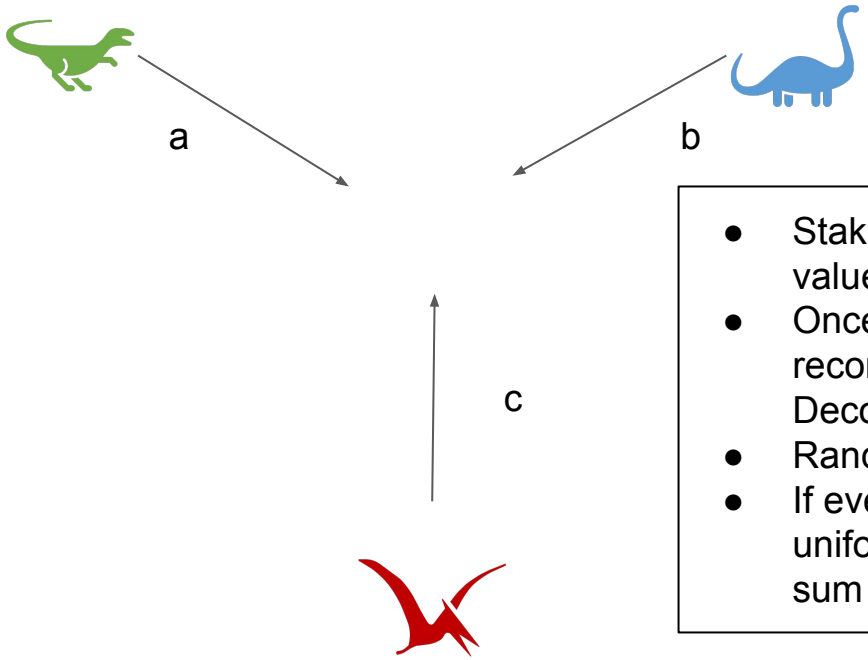
To commit:

$\mathrm{Com}(a) = \mathrm{H}(a,r)$ for some random r

To decommit:
Reveal a, r

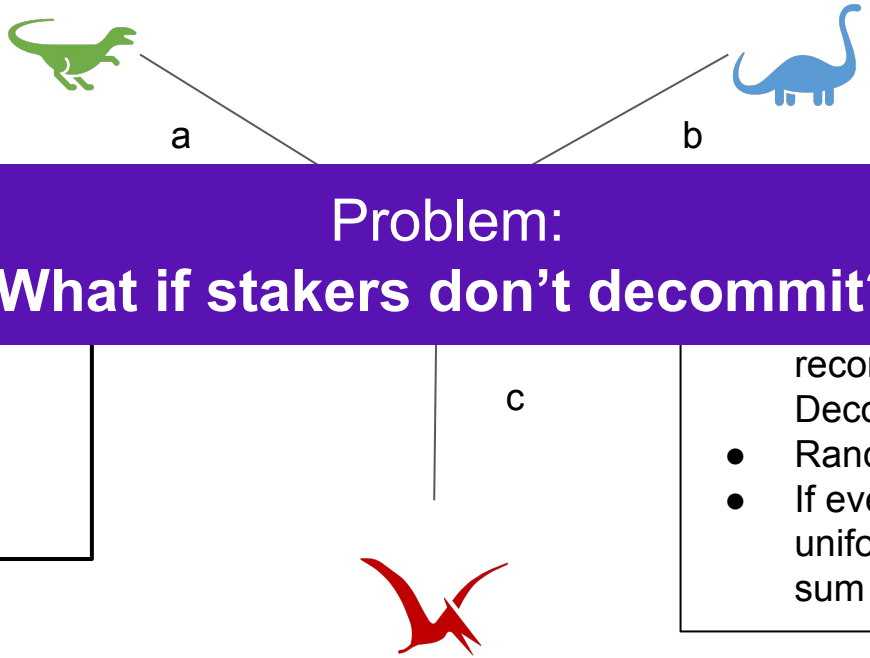
- Stakers Commit to their random values
- Once all commitments are recorded on chain Stakers Decommit
- Randomness is a + b + c
- If even one of a,b,c was uniformly random, then so is the sum

Method 2: Commit-Reveal



- Stakers Commit to their random values
- Once all commitments are recorded on chain Stakers Decommit
- Randomness is a + b + c
- If even one of a,b,c was uniformly random, then so is the sum

Method 2: Commit-Reveal



To commit:

Com(a) = H(a,r) for some random r

To decommit:
Reveal a, r

Problem:
What if stakers don't decommit?

Commit to their random

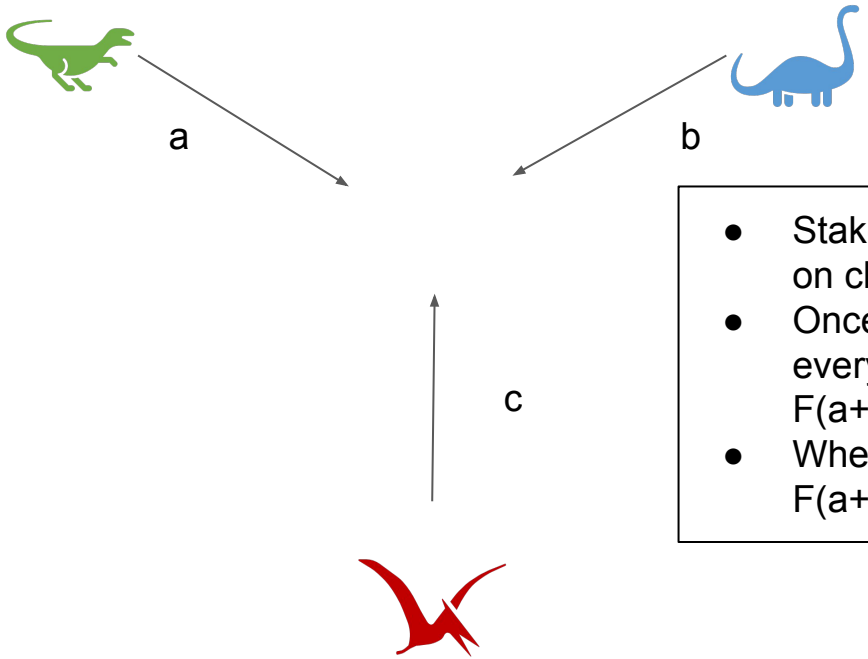
commitments are recorded on chain Stakers Decommit

- Randomness is a + b + c
- If even one of a,b,c was uniformly random, then so is the sum

Method 3: Verifiable Delay Functions (VDFs)

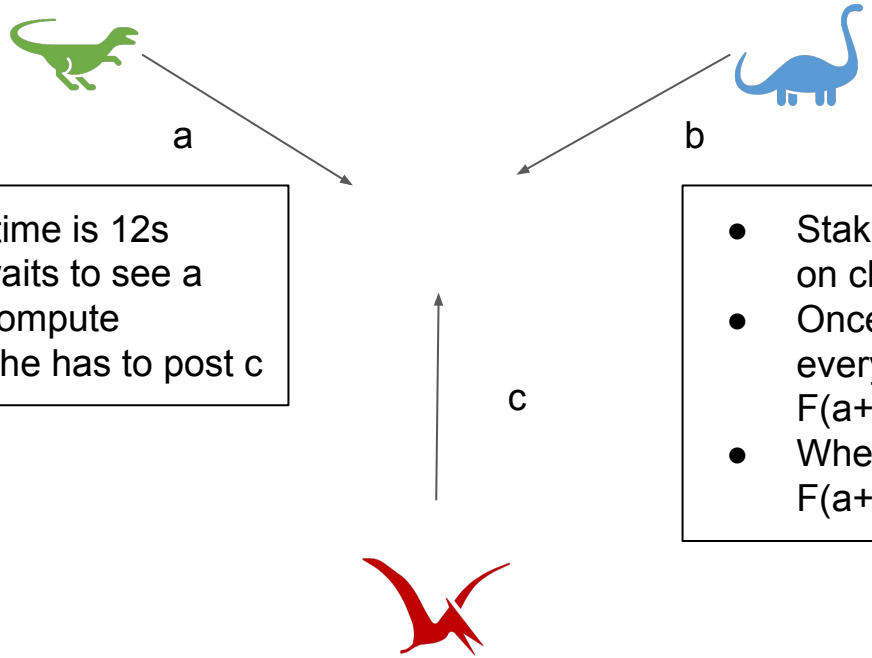
- What if you had a function, F, with the properties:
  - Delay: It takes at least 30 seconds to evaluate $F()$ using the best hardware
  - Verifiable: Given x and y, you can check whether $F(x) = y$ almost instantly
- This solves the decommitment problem

Method 3: VDFs



- Stakers post their randomness on chain.
- Once the block is finalized, everyone begins computing F(a+b+c).
- When computation is finished, F(a+b+c) is posted on chain

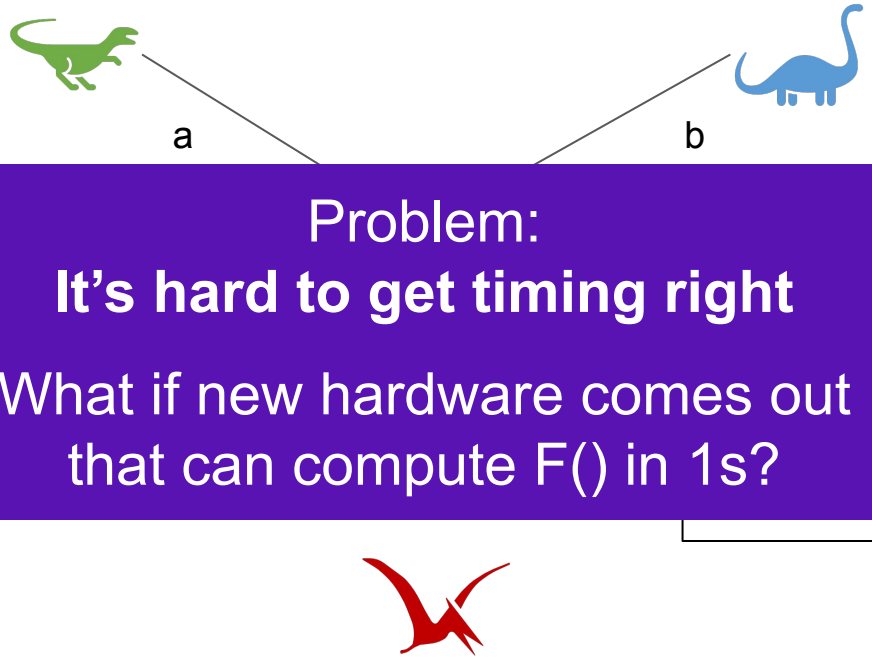
Method 3: VDFs



- Ethereum block time is 12s
- Even if Charlie waits to see a and b, he can't compute F(a+b+c) before he has to post c

- Stakers post their randomness on chain.
- Once the block is finalized, everyone begins computing F(a+b+c).
- When computation is finished, F(a+b+c) is posted on chain

Method 3: VDFs



- Ethereum bloc
- Even if Charlie
  and b, he can't
  F(a+b+c) befor

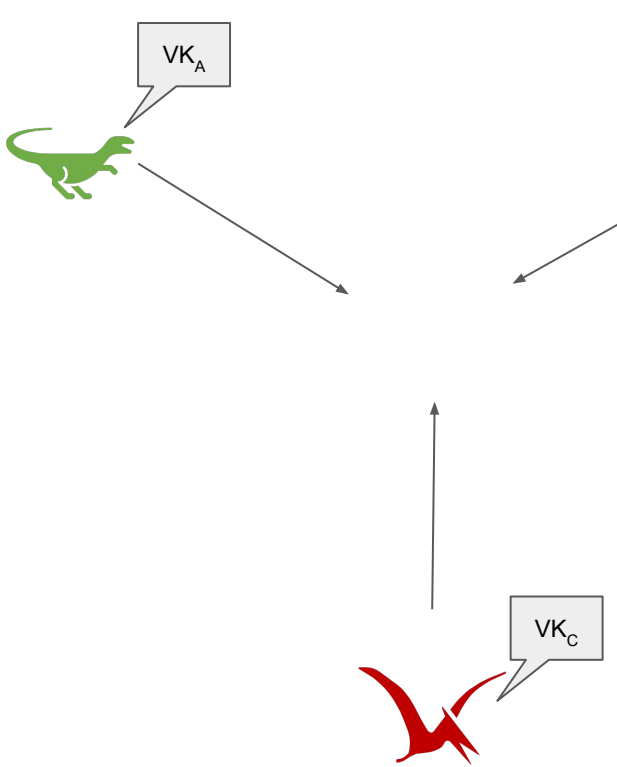
ers post their randomness
hain.
e the block is finalized,
yone begins computing
b+c).
n computation is finished,
b+c) is posted on chain

Problem:

It's hard to get timing right

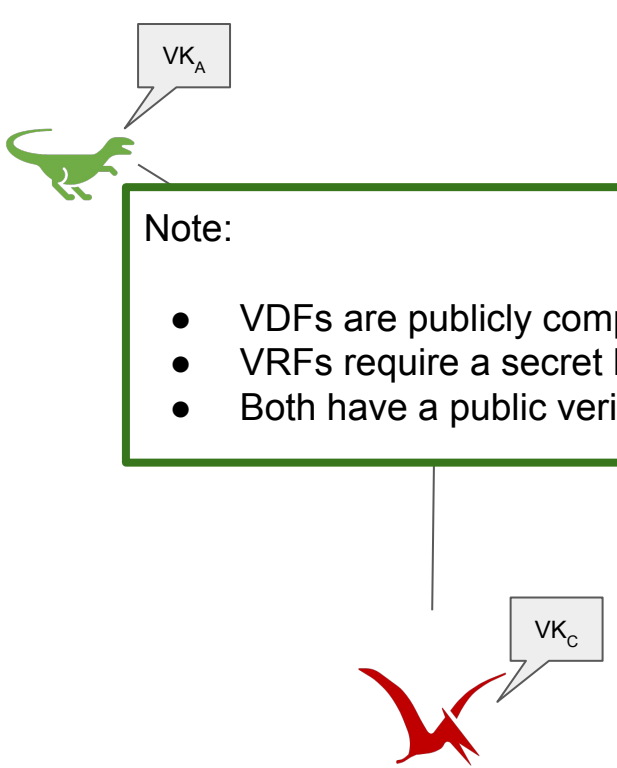
What if new hardware comes out that can compute F() in 1s?

Method 4: Verifiable Random Functions (VRFs)



- Stakers post a Verification Key for a VRF
  - This is usually just the same PK for their signature scheme
- Stakers submit their VRF applied to the previous block
- Randomness is a + b + c
- Publishing VKs effectively commits stakers to their inputs

Method 4: Verifiable Random Functions (VRFs)



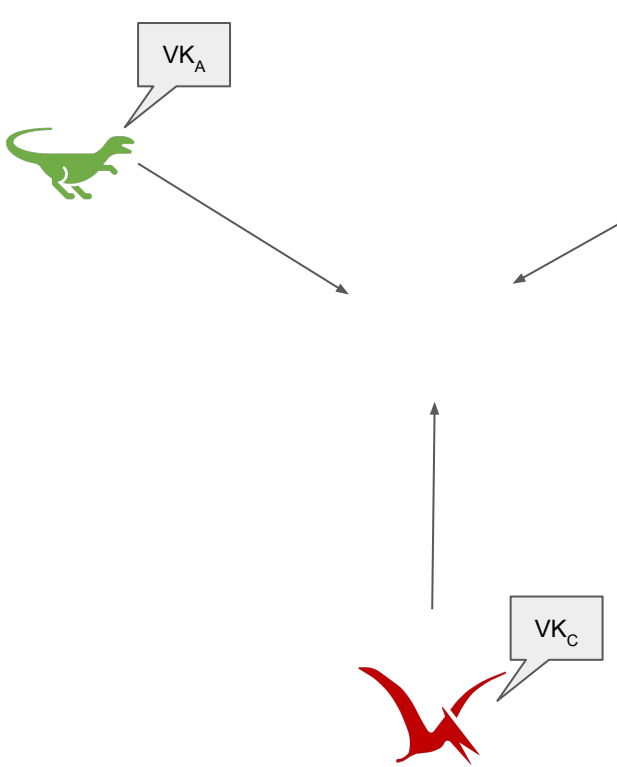
Note:

- VDFs are publicly computable
- VRFs require a secret key to compute
- Both have a public verification key

s post a Verification Key
RF
his is usually just the same PK for their signature scheme

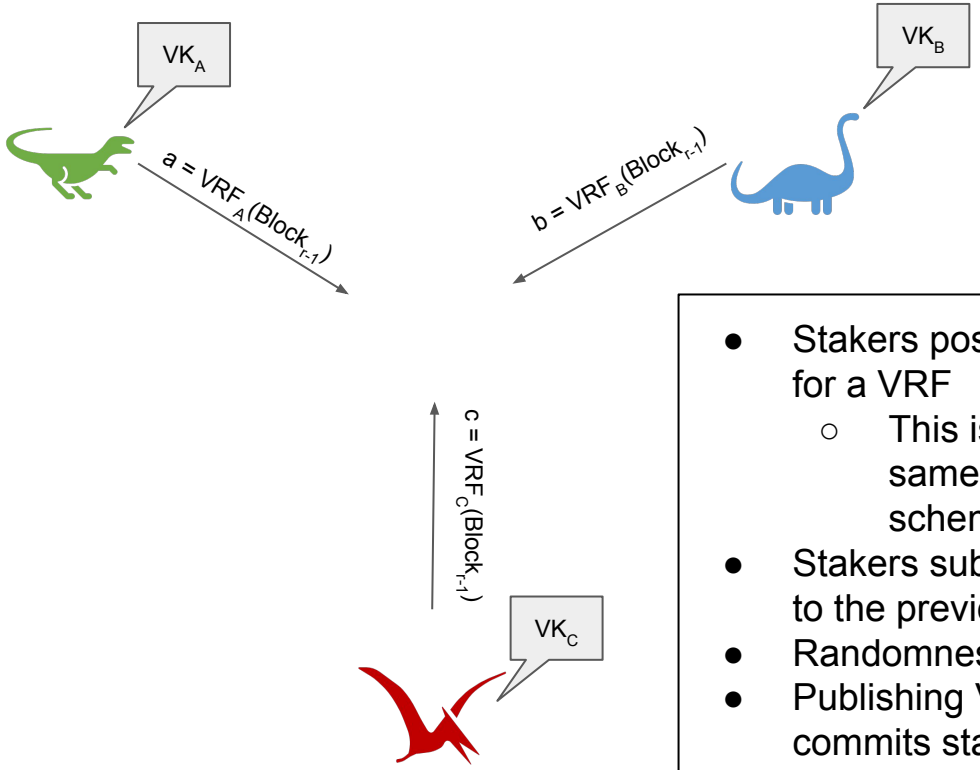
- Stakers submit their VRF applied to the previous block
- Randomness is a + b + c
- Publishing VKs effectively commits stakers to their inputs

Method 4: Verifiable Random Functions (VRFs)



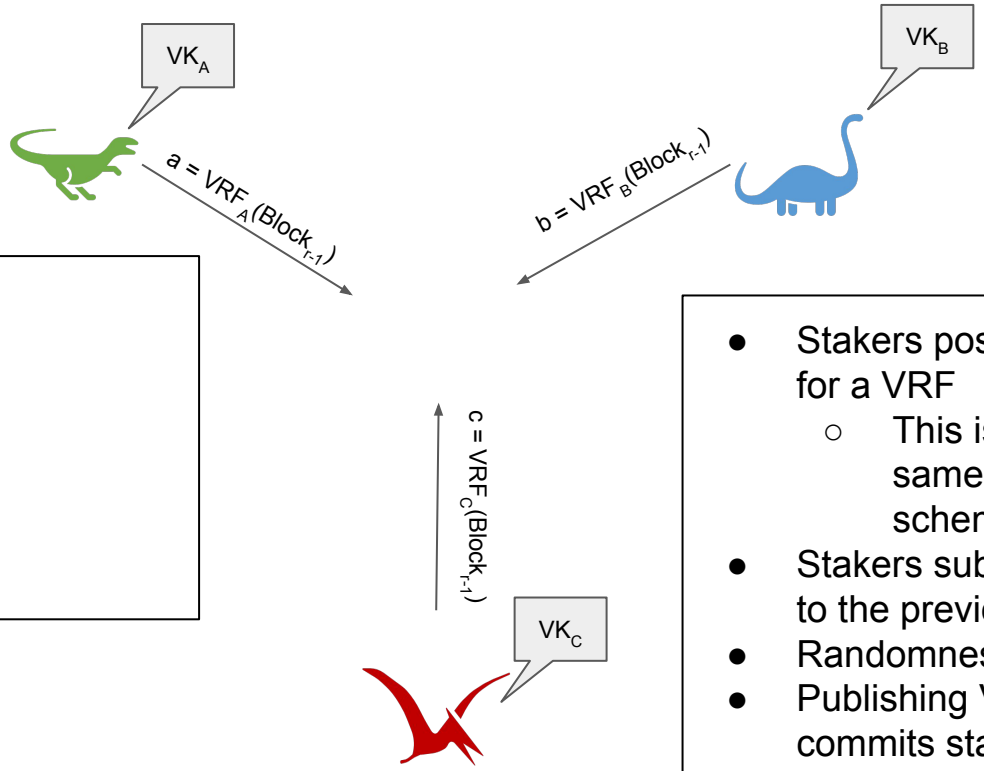
- Stakers post a Verification Key for a VRF
  - This is usually just the same PK for their signature scheme
- Stakers submit their VRF applied to the previous block
- Randomness is a + b + c
- Publishing VKs effectively commits stakers to their inputs

Method 4: Verifiable Random Functions (VRFs)



- Stakers post a Verification Key for a VRF
  - This is usually just the same PK for their signature scheme
- Stakers submit their VRF applied to the previous block
- Randomness is a + b + c
- Publishing VKs effectively commits stakers to their inputs

Method 4: Verifiable Random Functions (VRFs)

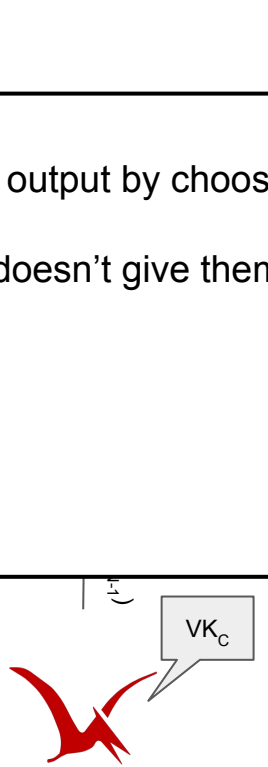


Widely used

- Ethereum
- Algorand
- Cardano
- Polkadot

- Stakers post a Verification Key for a VRF
  - This is usually just the same PK for their signature scheme
- Stakers submit their VRF applied to the previous block
- Randomness is a + b + c
- Publishing VKs effectively commits stakers to their inputs

Method 4: Verifiable Random Functions (VRFs)



Challenging to analyze:

- Validators can bias output by choosing not to submit randomness
- Need to show this doesn't give them a big advantage

Widely used

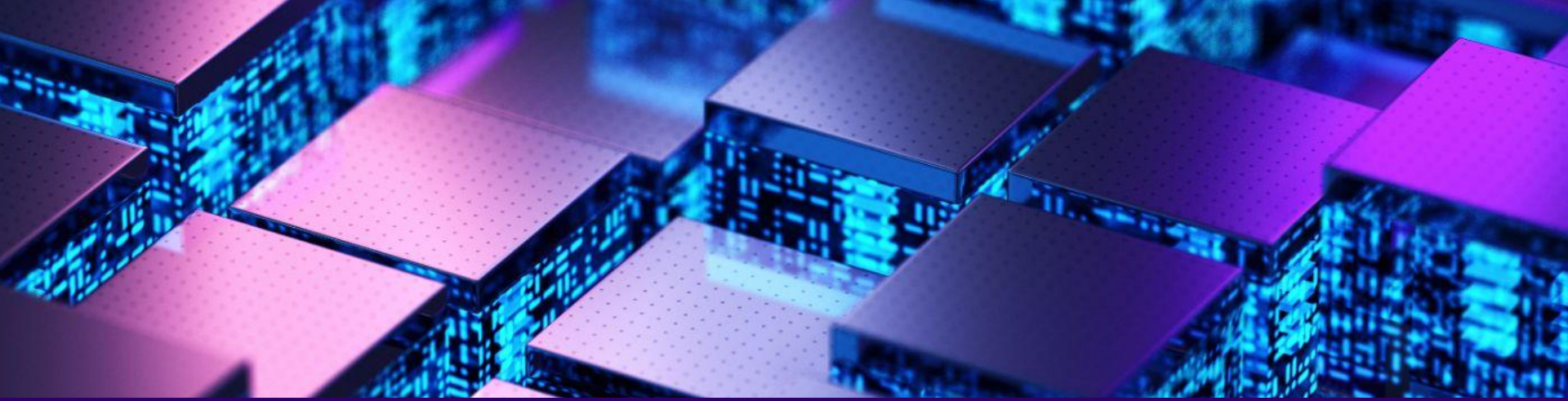
- Ethereu
- Algoran
- Cardan
- Polkadc

erification Key

ually just the
or their signature

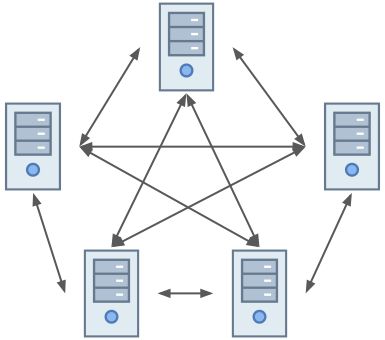
heir VRF applied to the previous block

- Randomness is a + b + c
- Publishing VKs effectively commits stakers to their inputs



Voting-based leader election

Voting for block producers



- Stakeholders cast stake-weighted votes to elect a committee
  - Sometimes called “delegation”
- Committee members take turns proposing blocks
- Committee runs a classical (permissioned) consensus algorithm to certify blocks
  - PBFT
  - Tendermint
    - CometBFT
  - Clique
  - Aura
  - Hotstuff
  - Consensus succeeds as long as 2/3rds of the committee is “honest”

Validator churn

- Election-based systems have validator sets that tend to ossify
  - On the Cosmos Hub only top 180 validators (by stake) participate
  - Thorchain forces “churn”
- Lottery-based systems can pull in validators with lower stake