CS 489 / 698: Software and Systems Security

Module 6: Non-technical Aspects of Security
a brief introduction on blockchains

Meng Xu *(University of Waterloo)*
Spring 2023

Outline

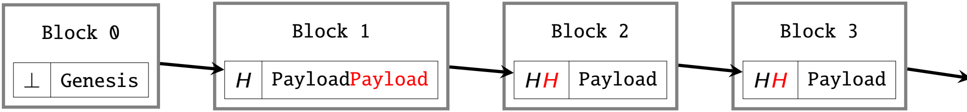
What is a blockchain?

A blockchain is ... a chain of blocks!

Block 1 → Block 2 → ...... → Block N

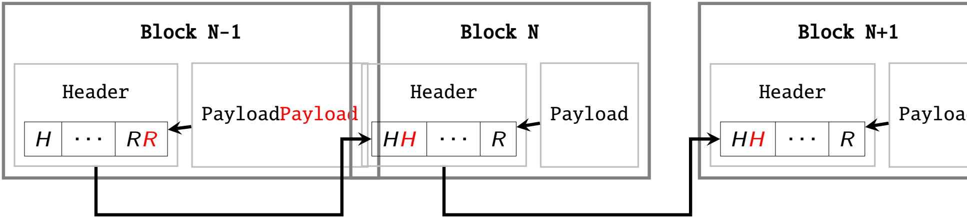
- What does chaining mean here?
  - Linked list? Some cryptographic construct?
- What goes into these blocks?
  - Anything? A fixed format? What makes a block valid?
- Who can put up a block?
  - A single entity? A group of people? Anyone with Internet access?
- How to ensure a same view of the chain?
  - Centralized? Distributed? How to resolve a dispute?

A basic chaining scheme



Each block contains a **cryptographic hash** of the previous block.

A better chaining scheme

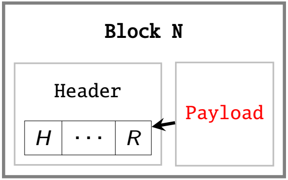


Each block is split into two parts:

- A *header* that contains at least two critical values:
  - A **cryptographic hash** of the previous block header.
  - A **cryptographic hash** of the current block payload.
- A *payload* that contains application-specific information

Q: Why this is a better chaining scheme?

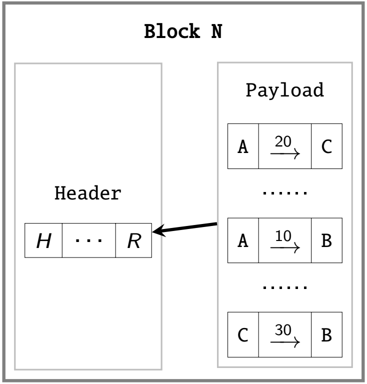
What goes into the payload?



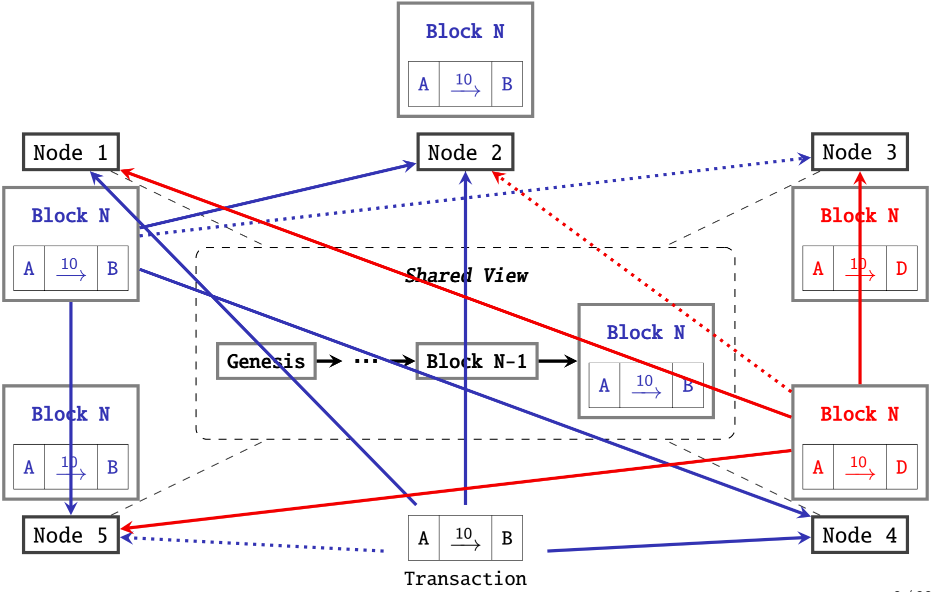
Anything! Depending on how you plan to use this blockchain.

- Bitcoin blockchain: ledger
- Ethereum blockchain: state machine

Payload example: a ledger



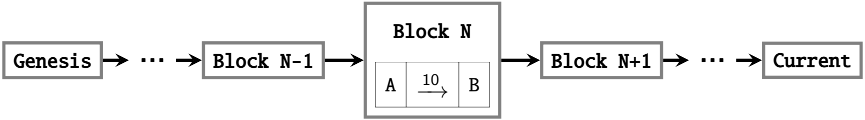
How does the data get into the block?



The power of consensus

Imagine Alice goes to Bob's Pizzeria and orders a pizza, she has the following payment options:

- cash, debit card, credit card, e-transfer (e.g., Interac®)
- an entry in the blockchain-based ledger



To the best of Bob~~Bob~~ everyone's knowledge:

- It is hard for Alice to produce such a chain of blocks
- There does not exist a better chain of blocks as of now

Summary

Pay attention to two aspects when you design/analyze a blockchain:

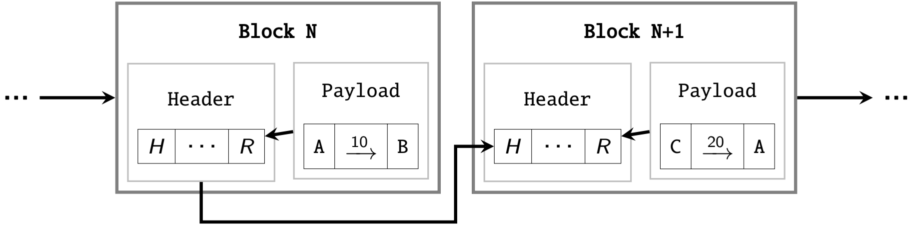
- What goes into a block?
- How to ensure consensus?

In most blockchain systems, these two aspects are orthogonal.

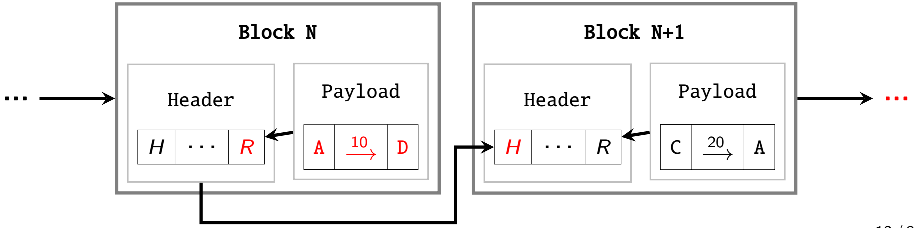
Outline

1. An overview of blockchain design space
2. Consensus: Proof-of-Work
3. Consensus: Proof-of-Stake

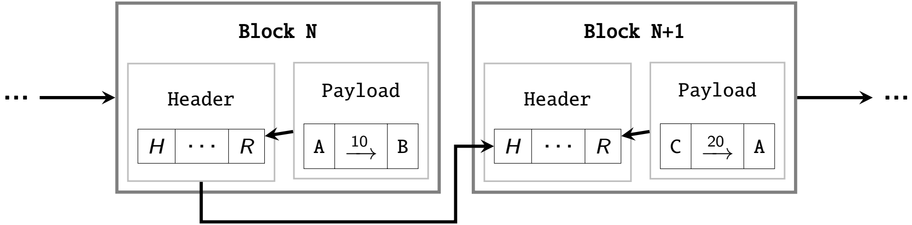
How hard it is to alter this chain?



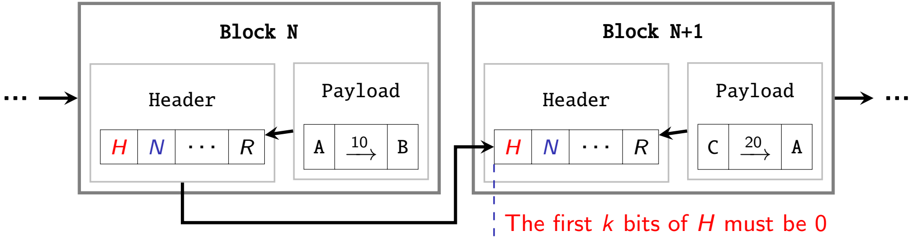
This is the chain Alice shows Bob w.r.t her payment to Bob.
It is not hard at all for Alice to revert the payment to Bob!



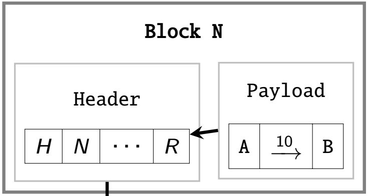
Increase the difficulty



Bob decides to make it harder for Alice to alter her payment



Mining for a valid hash



$N = 0 \implies Hash(H \,\|\, N \,\|\, \ldots \,\|\, R) =$ `0x349c1a7e...` ✗

$N = 1 \implies Hash(H \,\|\, N \,\|\, \ldots \,\|\, R) =$ `0x6ffde7bf...` ✗

......

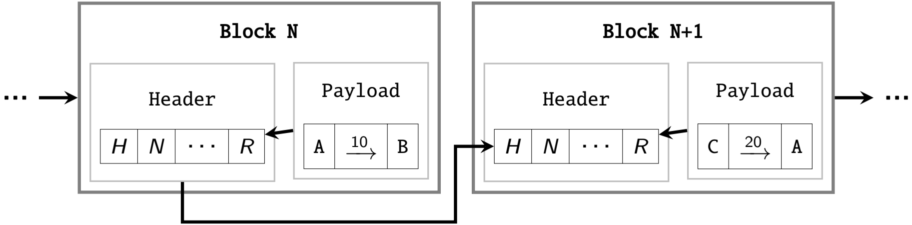
$N = x \implies Hash(H \,\|\, N \,\|\, \ldots \,\|\, R) = \texttt{0x00.}_k\texttt{.004f7fed1a}$

Q: What is the chance of finding a valid N assuming an m-bit hash?

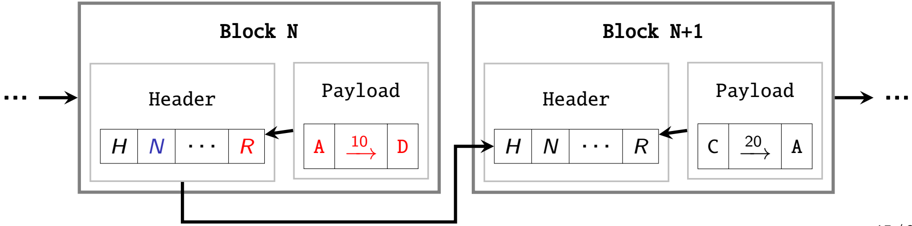
A: $\frac{2^{m-k}}{2^m}$, a larger $k \implies$ a higher difficulty of finding N

i.e., expect 2^k hash operations to find a valid N on average.

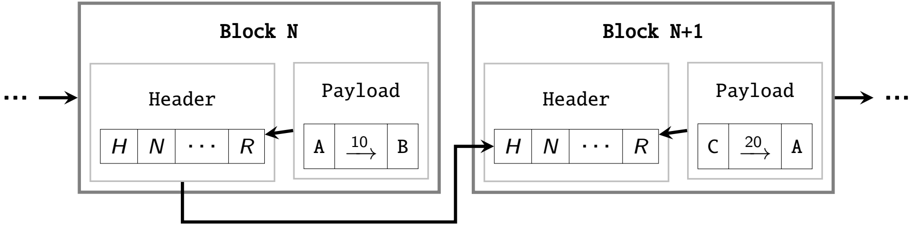
How does mining deter alteration? - Case 1



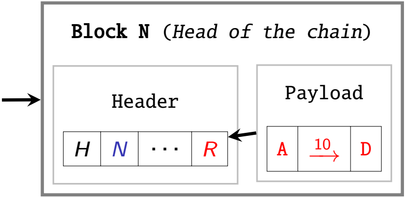
Case 1: Alice re-mines block N and finds a new nonce such that the block header hash remains unchanged



How does mining deter alteration? - Case 2

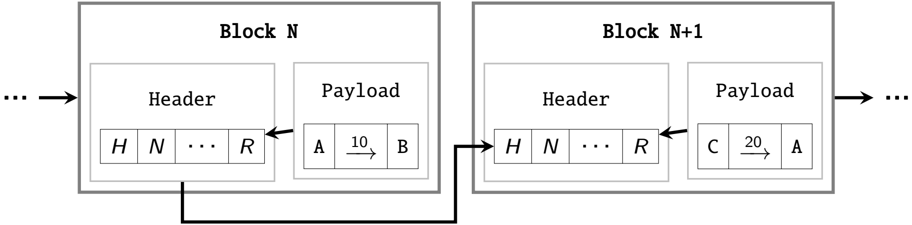


Case 2: Alice re-mines the nonce for block N and stops there

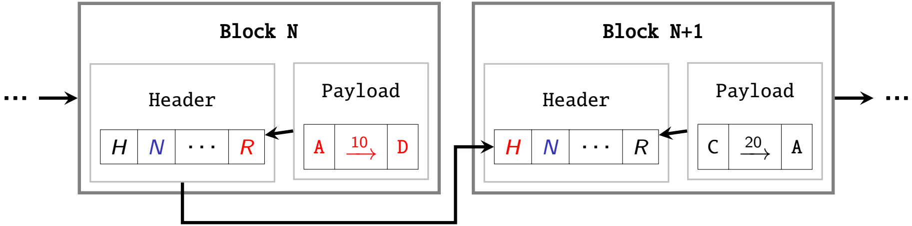


Deterrent: longer chains are preferred over shorter chains.

How does mining deter alteration? - Case 3



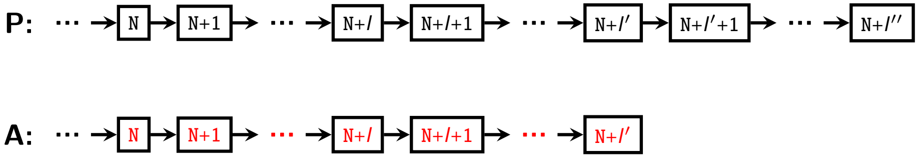
Case 3: Alice re-mines all the nonces since block N



Deterrent: If there are l blocks between and including block N and

51% attack

There is a catch in the deterrent:
Alice mines slower than the rest of the participants combined.

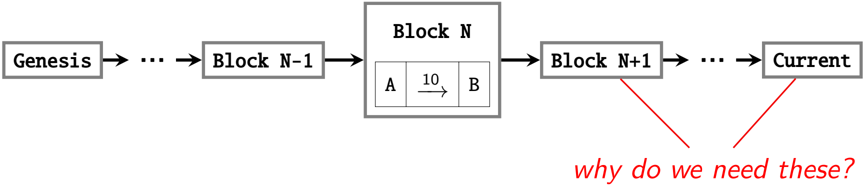


$\Longrightarrow$ the public chain grows faster than Alice's chain.

Q: what if Alice mines faster?
A: Alice gets to rewrite the history.

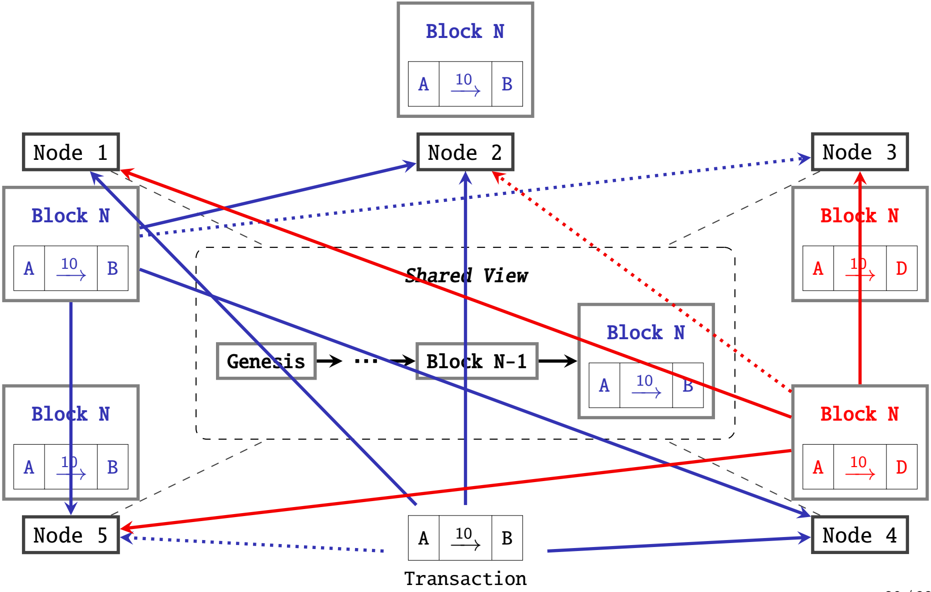
Confirmation level

Recall that when we show a proof of payment, we need a few extra blocks after the block that hosts the ledger entry.

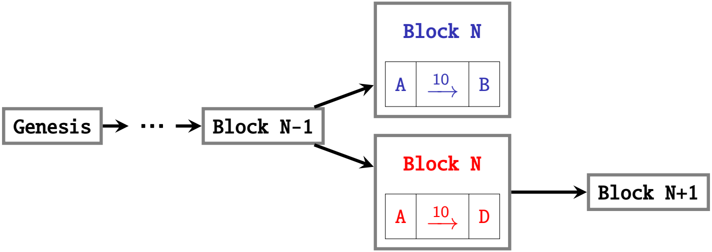


Q: Why do we need these extra blocks even when
1) Alice does not control over 50% of computational power and
2) everyone else is honest and cooperative?

How does the data get into the block?



Back to confirmation level



To trigger a fork, Alice could

- Send two transactions in a short time window
- Send two transactions to separate halves of the network
- Pre-mine one block and only reveal it after the first transaction is sent to the network

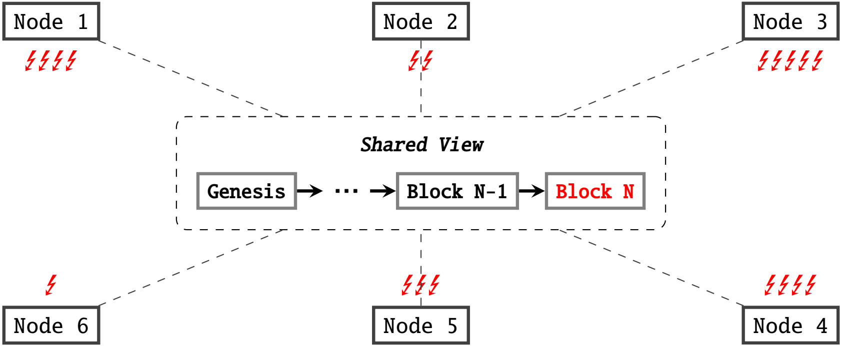
Drawbacks of Proof-of-Work consensus

- Speed of confirmation
  - E.g., a Bitcoin transaction takes on average 10 minutes to confirm
  - Even worse, it is advised to wait for 6 confirmations, i.e., 1 hour.
- Vulnerable to 51% attacks
  - In 2014, mining pool `Ghash.io` obtained 51% hash rate in Bitcoin
  - Bitcoin Gold, was hit by such attacks twice in 2018 and 2020
- Energy consumption
  - Hashing itself is not useful
  - And such useless operations are repeated across the fleet of nodes

Outline

1. An overview of blockchain design space
2. Consensus: Proof-of-Work
3. Consensus: Proof-of-Stake

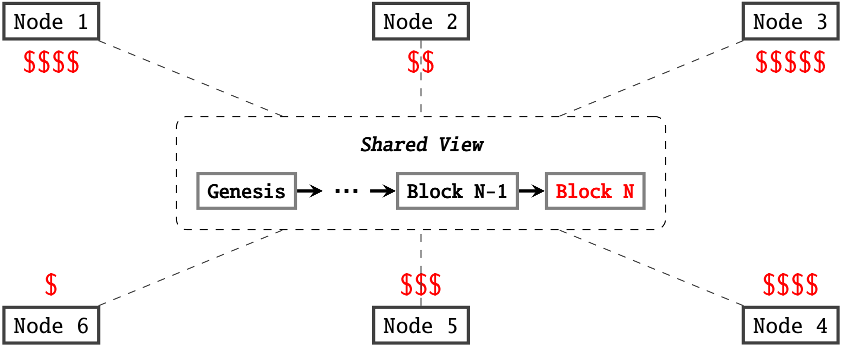
Block production as election



In a proof-of-work scheme,

- the chance of which node is elected to propose a new block is proportional to its hashing power
- collisions are allowed and are resolved by the longest chain rule

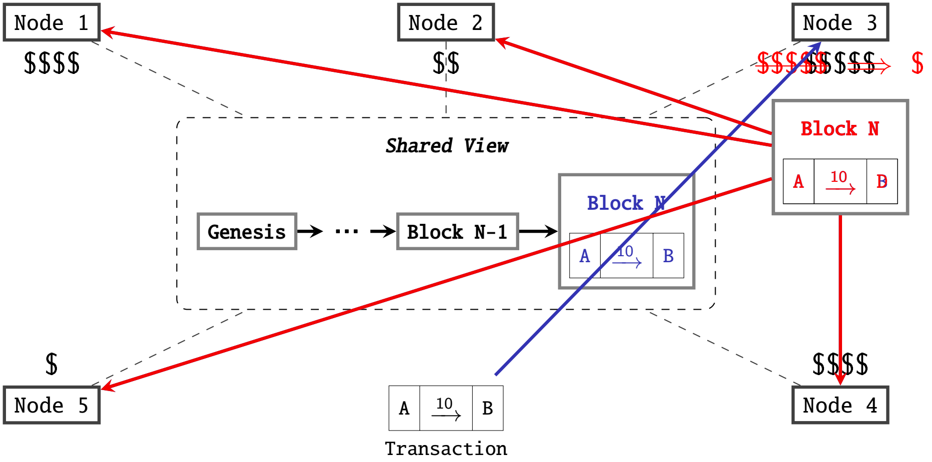
Block production as election



In a proof-of-stake scheme,

- the chance of which node is elected to propose a new block is proportional to its staked value
- collisions are not allowed by design, only the leader creates a block

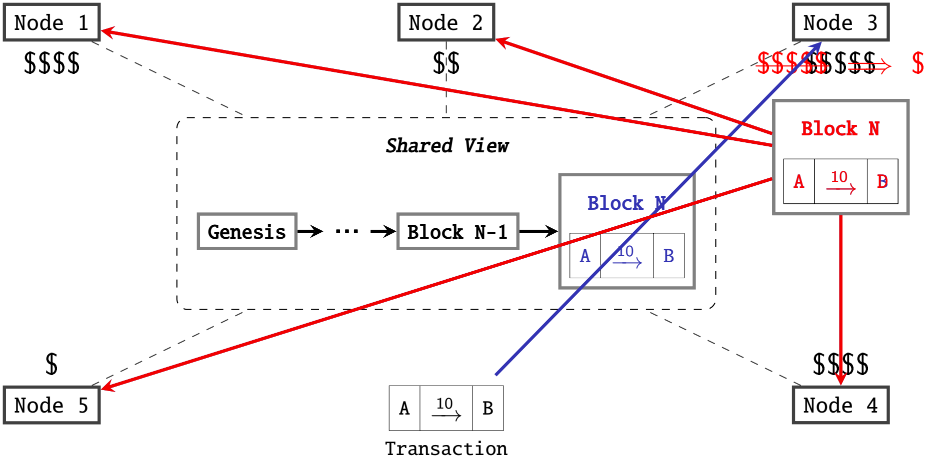
Transaction lifecycle in PoS



The 51% attack in PoS

Q: What if the attacker controls $\geq$ 50% of staked resources?

Transaction lifecycle in PoS



The 51% attack in PoS

Q: What if the attacker controls $\geq$ 50% of staked resources?

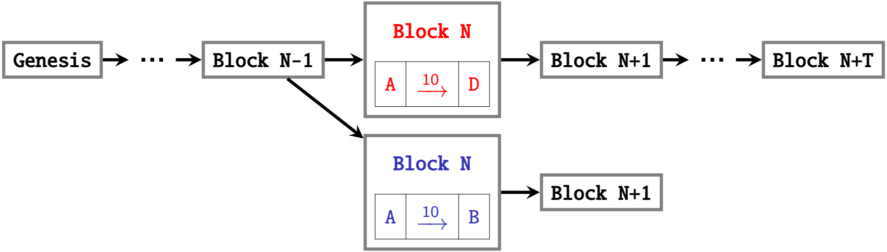
A: The attacker can prove fraudulent transactions.

Q: Is 51% attack less likely in PoS compared with PoW?

A: Yes, because in PoS, the attacker losses the weapon to future attacks, i.e., all the stake are gone, and is not easily recoverable!

Hard fork as a recovery of a 51% attack

To recover from a 51% attack, the only solution is to hard fork the blockchain in order to invalidate the fraudulent transactions added by the attackers.



NOTE: the forked chain can be shorter than the previous chain!

$\Longrightarrow$ a higher level of social coordination is required

Hard fork as a recovery of a 51% attack

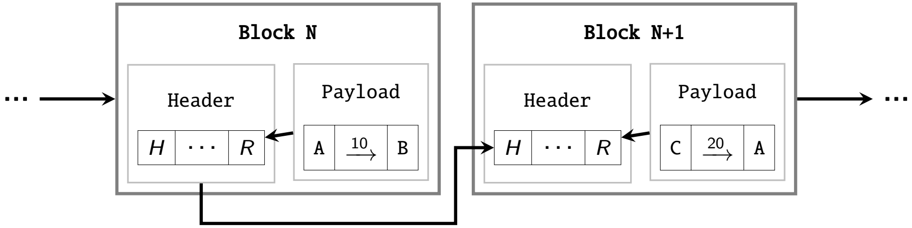
In PoS, we do a hard fork to invalidate fraudulent transactions AND wipe out the attacker who controls $\geq$ 50% of the staked resources.

In PoW, the hard fork can only invalidate transaction WHILE the $\geq$ 50% computational power is still controlled by the attacker.

Chain validation

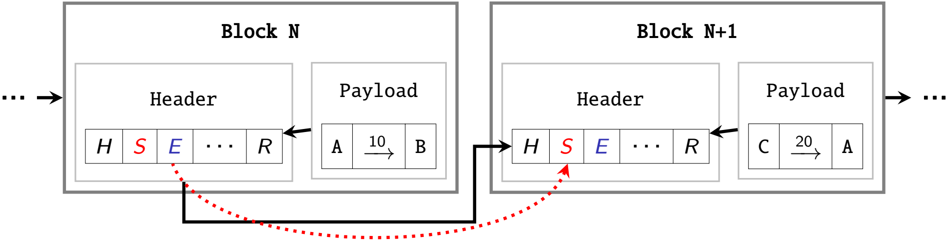
If Alice shows Bob, the Pizzeria owner, the following blockchain, why would Bob accept it? Why would Bob believe that

- It is hard for Alice to produce such a chain of blocks
- There does not exist a better chain of blocks as of now



Chain validation

This turns out to be an extremely complicated problem!

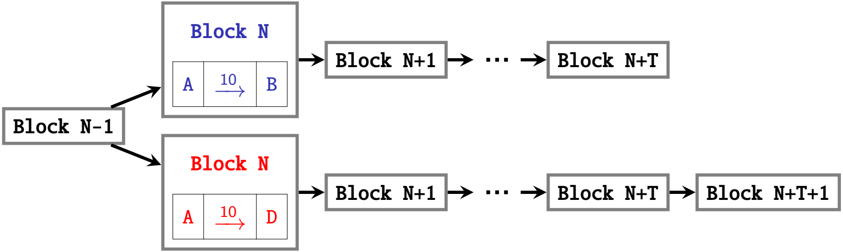


- S - Signature of the proposer of this block
- E - Election packet that records how this proposer is elected

Q: What are the issues with this scheme?

The Nothing-at-Stake problem

Assuming Alice has some stake (e.g., 1%) and can be elected as a block proposer:



In one of her turn as a block proposer, Alice triggers a fork in the chain with an attempt to double-spend. The next block proposer, even honest, has no incentive to select which chain to converge on. The proposer has no idea which chain will survive in the future, the logical thing to do is to mine on both. When its Alice's turn again, she only append a block to the chain that is more favorable

The Nothing-at-Stake problem

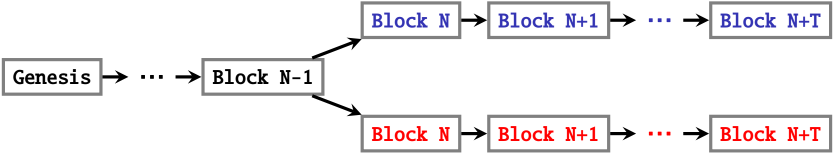
Solution? There is no common solution. Different PoS chains adopt different mechanisms.

The Slash protocol (Ethereum PoS candidate) has two rules:

- Penalize those who "equivocated" on a given block, i.e., voted on two different versions of it.
- Penalize those who voted on the wrong block, regardless of whether or not they double-voted.

Long-range attacks (the bootstraping problem)

Bob first joins the network, which chain should he accept?



Q: Why this is not a problem in PoW?

A: Because it is computationally expensive to create a counterfeit chain in PoW. But it is easy (almost no cost) in the PoS case.

Long-range attacks (the bootstraping problem)

Solution? In short, there is no simple solutions.

- Casper (Ethereum's PoS protocol) depends on trusted nodes to broadcast the correct block hash.
- Peercoin, broadcasts the hash of the "legitimate" chain on a daily basis.
- Extremely complicated solutions have been proposed e.g., Ouroboros Genesis.

$\langle$ **End** $\rangle$