

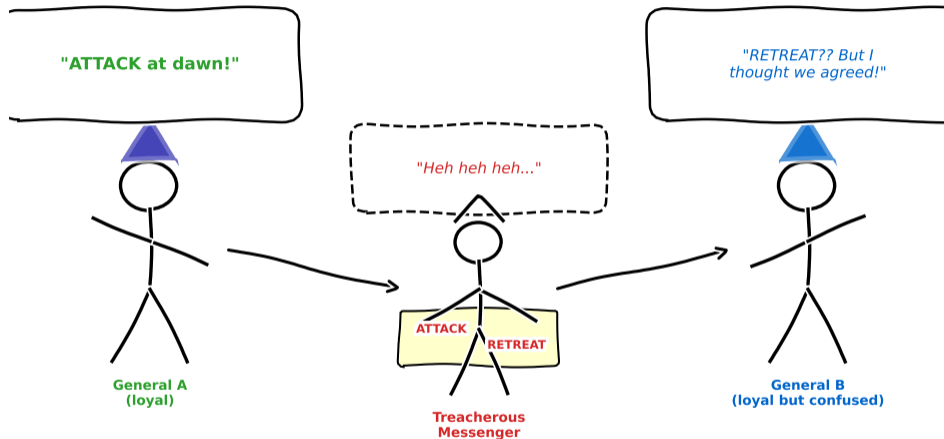
# Lesson 3.2: Consensus Mechanisms and Blockchain Architecture

## Module 3: The Trust Problem

Prof. Dr. Joerg Osterrieder

Digital Finance — BSc Course

# The Generals' Dilemma



**The Byzantine Generals Problem: How do you reach agreement when the messenger might be a liar?**

After completing this lesson, you will be able to:

- 1 **Explain** the Byzantine Generals Problem and why it matters for distributed financial systems [Understand]
- 2 **Describe** how Proof of Work uses computational puzzles, difficulty adjustment, and block rewards to achieve consensus [Apply]
- 3 **Compare** Proof of Stake mechanisms (validator selection, slashing, finality) with Proof of Work [Analyze]
- 4 **Analyze** the blockchain trilemma and evaluate the trade-offs each consensus mechanism makes [Analyze]
- 5 **Evaluate** which consensus mechanism is most appropriate for a given financial application [Evaluate]

**Bloom's levels covered:** Understand, Apply, Analyze, Evaluate

---

Objectives follow Bloom's taxonomy: Understand → Apply → Analyze → Evaluate.

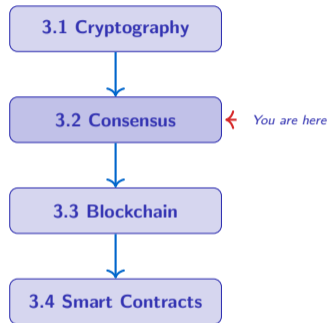
**Lesson 3.1 gave us:** The cryptographic toolkit — hash functions, Merkle trees, digital signatures.

**The unsolved problem:** Two valid, conflicting transactions with correct signatures. **Who wins?**

**We have the cryptographic tools. Now: how do we build agreement?**

- Cryptography ensures **integrity and authentication**
- But it cannot establish a **global ordering** of transactions
- For that, we need **consensus mechanisms**
- Consensus = how strangers agree on a single truth

**This lesson:** Byzantine Generals → Proof of Work → Proof of Stake → BFT variants → the blockchain trilemma.



---

Consensus mechanisms bridge the gap between cryptographic integrity and trustless transaction ordering.

# What Is the Byzantine Generals Problem?

## Definition: Byzantine Generals Problem

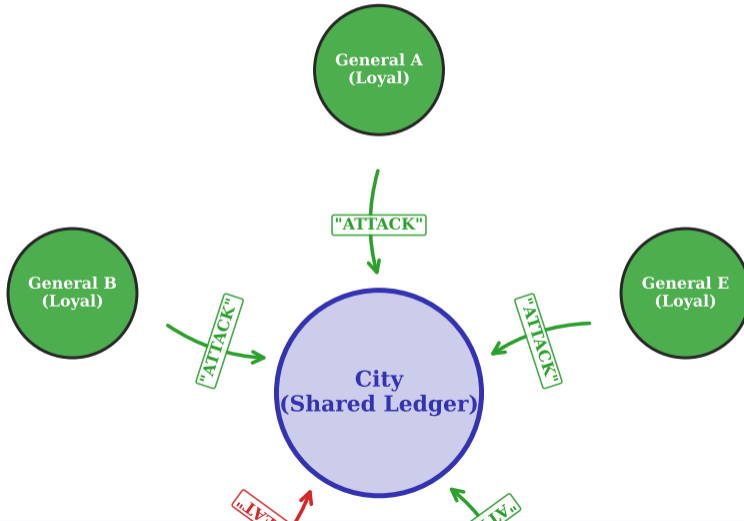
Several generals must coordinate an **attack or retreat**, but they can only communicate through **messengers who might be traitors**. The challenge: reach the same decision even when some participants send conflicting messages.

### Mapping to blockchain:

Military Metaphor	Blockchain Equivalent
Generals	Network nodes (miners, validators)
Attack / Retreat	Transaction ordering (which block is correct?)
Traitors	Malicious nodes sending conflicting data
Messengers	Network communication (Peer-to-Peer or P2P protocol)
Agreement	Consensus on a single blockchain state

The Byzantine Generals Problem (Lamport et al., 1982) formalized the challenge of agreement in the presence of faulty participants.

## The Byzantine Generals Problem



## Definition: Byzantine Fault Tolerance

A distributed system is **Byzantine Fault Tolerant (BFT)** if it continues to function correctly even when up to  $f$  out of  $n$  participants behave arbitrarily (maliciously, crash, send conflicting messages). Classical BFT requires  $n \geq 3f + 1$ .

### Types of faults:

- **Crash fault:** A node goes offline (benign failure)
- **Byzantine fault:** A node actively lies, sends different messages to different peers, or colludes with other faulty nodes

### The $n > 3f$ bound:

- With 4 generals and 1 traitor: agreement is possible
- With 3 generals and 1 traitor: **impossible** — the traitor can cause a split
- Bitcoin's innovation: make Byzantine behavior **economically irrational** rather than technically impossible

---

BFT is the theoretical foundation — Proof of Work and Proof of Stake are practical implementations.

## Definition: Proof of Work (PoW)

A consensus mechanism where participants (“miners”) compete to solve a computationally expensive puzzle. The first miner to find a valid solution earns the right to propose the next block and receives a **block reward**. The puzzle is hard to solve but easy to verify.

**The puzzle:** Find a nonce  $x$  such that:

$$\text{SHA-256}(\text{block\_header}||x) < \text{target}$$

**Key properties:**

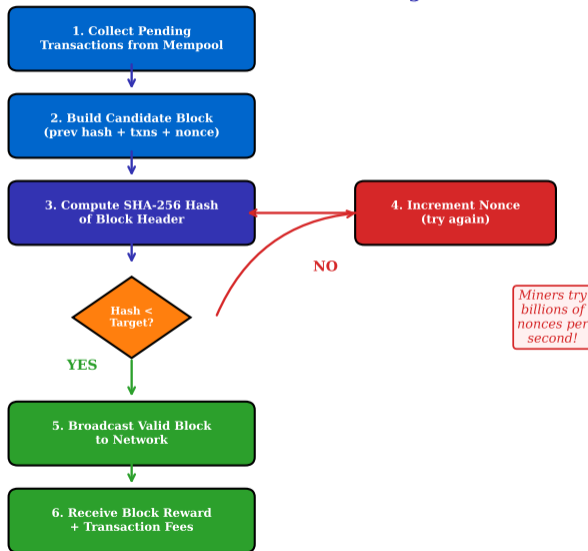
- **Expensive to compute:** Requires billions of hash attempts (brute force)
- **Trivial to verify:** Any node checks a single hash in microseconds
- **Asymmetry:** This “easy to check, hard to find” property is what makes PoW secure

**Analogy:** Finding a needle in a haystack is hard; confirming you hold a needle is instant.

---

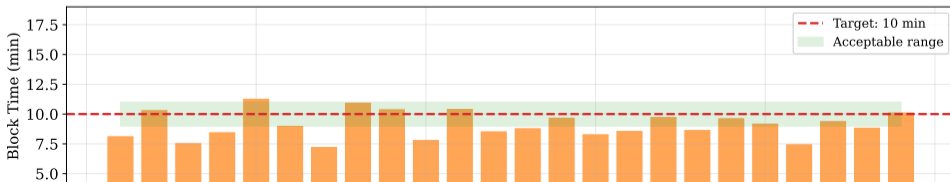
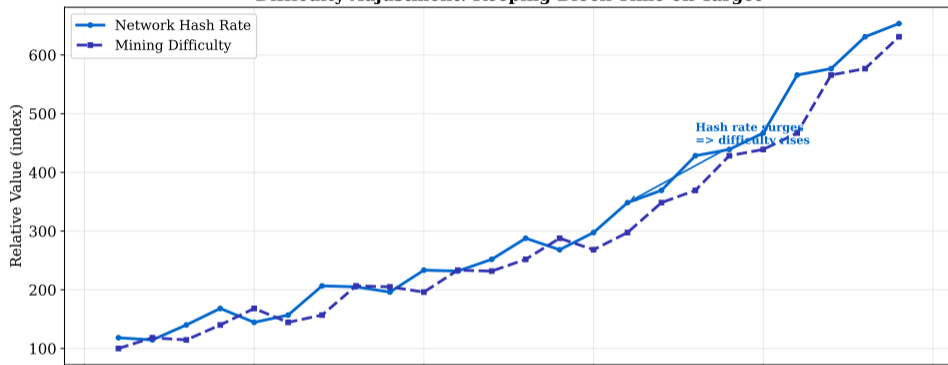
**Proof of Work converts electricity into consensus: the cost of cheating exceeds the cost of playing honestly.**

## Proof of Work: The Mining Process



# Difficulty Adjustment: Keeping Block Time Stable

## Difficulty Adjustment: Keeping Block Time on Target



## Miners are compensated with:

- 1 **Block reward:** Newly minted coins (currently 3.125 BTC per block after the April 2024 halving)
- 2 **Transaction fees:** Fees paid by users for inclusion in the block

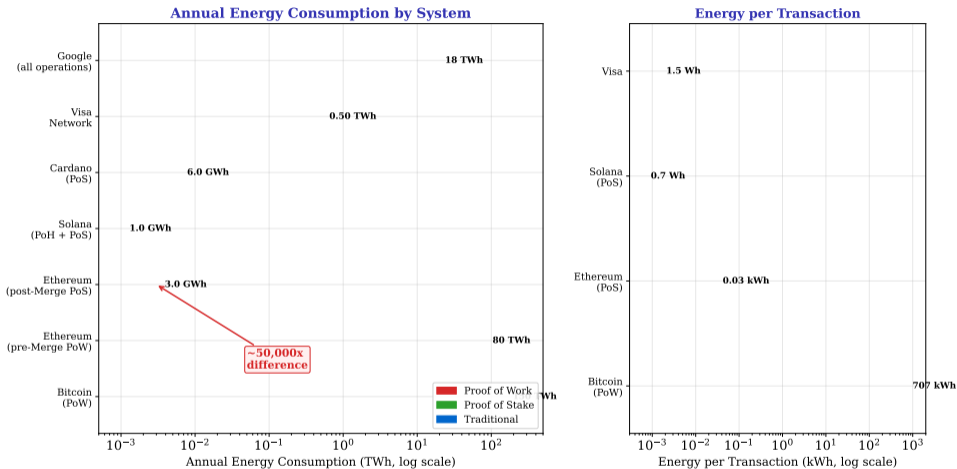
## Bitcoin's halving schedule:

Year	Block Reward	Total BTC Mined	Event
2009	50 BTC	0	Genesis block
2012	25 BTC	10.5M	1st halving
2016	12.5 BTC	15.75M	2nd halving
2020	6.25 BTC	18.375M	3rd halving
2024	3.125 BTC	19.6875M	4th halving
~2140	0 BTC	21M	Final coin mined

**Key insight:** As block rewards shrink, transaction fees must increasingly fund network security.

The halving creates a deflationary supply schedule: 21 million BTC is the hard cap, reached approximately in the year 2140.

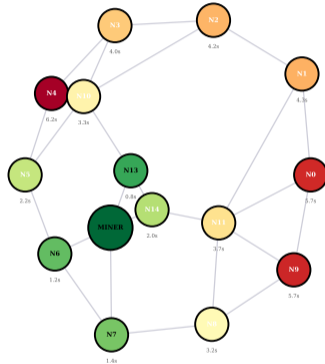
# PoW Energy Consumption: The Environmental Debate



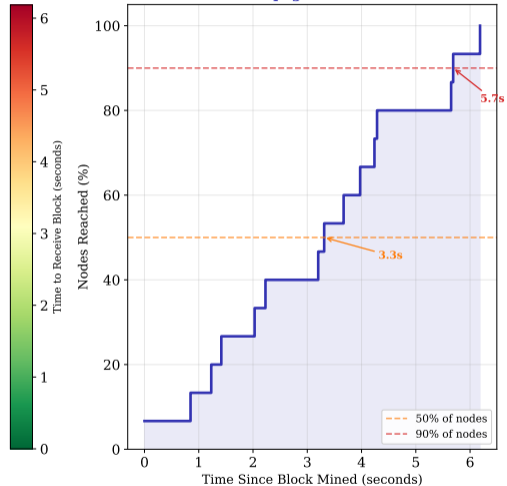
- **What you see:** Two log-scale panels comparing energy: left shows annual TWh for various systems; right shows energy per transaction
- **Key pattern:** Bitcoin uses 150 TWh/year, 50,000x more than Ethereum PoS; per-transaction cost is 707 kWh vs 0.03 kWh

# Block Propagation: How Blocks Spread Through the Network

## Block Propagation Across P2P Network

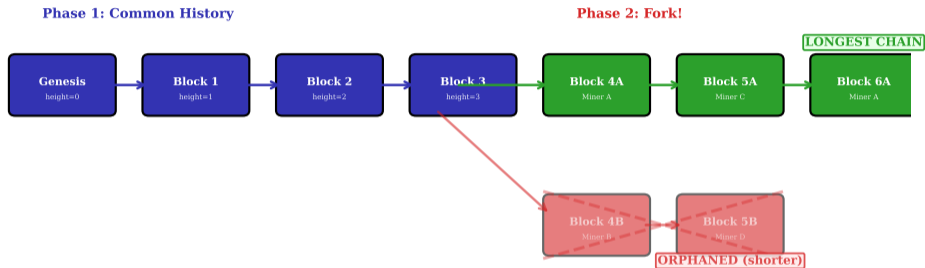


## Propagation CDF



- **What you see:** Left — P2P network graph with nodes colored by time-to-receive; right — CDF showing 50% reached in 4s, 90% in 7s

## Fork Resolution: The Longest Chain Rule



### Resolution Rule:

1. Two miners simultaneously find valid blocks => network temporarily splits
2. Both chains continue building; miners choose which chain to extend
3. Eventually one chain grows longer (more cumulative work)
4. All nodes switch to the longest chain; shorter chain is orphaned
5. Transactions in orphaned blocks return to the mempool

*In Bitcoin, 6 confirmations (blocks built on top) provide >99.9% finality assurance.*

## Definition: 51% Attack

If a single entity controls more than 50% of the network's total hash power, it can build a private chain faster than the honest network. The attacker can then: (1) **reverse transactions** by releasing the longer private chain, and (2) **double-spend** by spending coins on the public chain, then orphaning those blocks.

### What a 51% attacker CAN do:

- Reverse recent transactions (double-spend)
- Prevent specific transactions from being confirmed
- Prevent other miners from earning block rewards (selfish mining)

### What a 51% attacker CANNOT do:

- Steal coins from other people's wallets (still need private keys)
- Change the block reward amount (protocol rules enforced by all nodes)
- Create coins out of thin air

---

A 51% attack on Bitcoin would cost billions in hardware and electricity — making it economically irrational.

# What Is Proof of Stake?

## Definition: Proof of Stake (PoS)

A consensus mechanism where **validators** lock up (“stake”) cryptocurrency as collateral. The protocol selects a validator to propose the next block with probability proportional to their stake. Dishonest behavior results in **slashing** — the validator loses part or all of their staked funds.

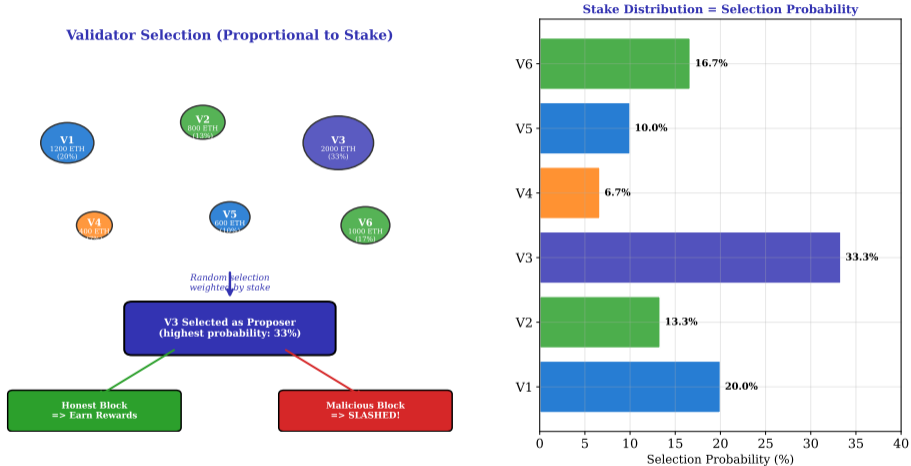
### The shift from PoW to PoS:

Property	PoW	PoS
Resource consumed	Electricity	Capital (staked coins)
Who participates?	Miners (hardware)	Validators (stakers)
Sybil resistance	Cost of computing	Cost of acquiring stake
Environmental cost	Very high	Negligible

**Core idea:** Replace “burn electricity to prove work” with “lock capital to prove commitment.”

PoS replaces computational competition with economic commitment: your stake is your skin in the game.

# Validator Selection: Stake-Weighted Randomness



- **What you see:** Left — validator pool with circles sized by stake (V3 has 33%); right — bar chart showing selection probability matches stake percentage
- **Key pattern:** Random selection weighted proportionally to stake; honest block earns rewards, malicious block triggers slashing

Slashing punishes validators who violate the protocol:

Violation	Description	Penalty
Double voting	Signing two different blocks at the same height	Severe slash (e.g., 1/32 of stake on Ethereum)
Surround voting	Creating a vote that contradicts a prior vote	Severe slash
Inactivity leak	Going offline for an extended period	Gradual stake reduction until $\frac{1}{3}$ threshold recovers

Why slashing works:

- A validator with 32 ETH staked risks real financial loss
- The expected cost of cheating exceeds the expected gain
- This creates **economic finality**: once enough validators attest to a block, reversing it would destroy their capital

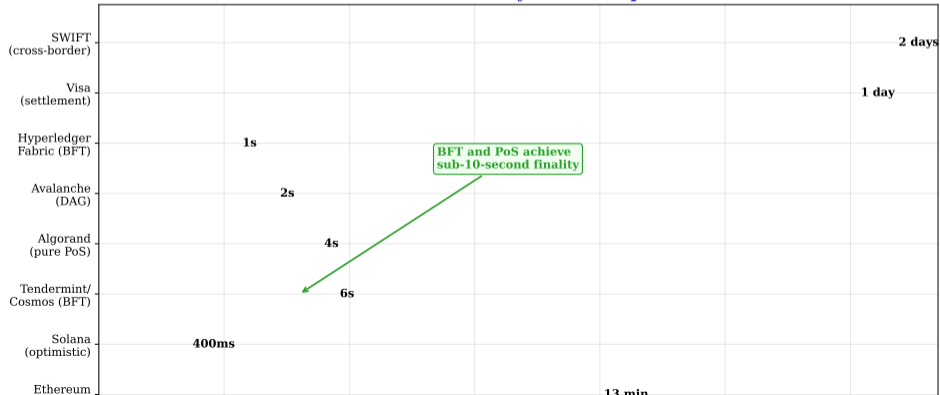
Slashing converts consensus from “majority of hash power” (PoW) to “majority of capital at risk” (PoS).

# Finality: When Is a Transaction Truly Final?

## Definition: Finality

A transaction has **finality** when it cannot be reversed, cancelled, or altered. In probabilistic finality (PoW), confidence increases with each confirmation. In deterministic finality (BFT/PoS), a transaction is final once a supermajority of validators attest to it.

### Transaction Finality Time Comparison



## Case Study: Ethereum's Transition from PoW to PoS

### “The Merge” — September 15, 2022:

- Ethereum switched from Proof of Work to Proof of Stake in a live network upgrade
- **Energy reduction:**  $\sim 99.95\%$  (from  $\sim 80$  TWh/year to  $\sim 0.003$  TWh/year)
- **No downtime:** The transition happened without pausing the network

### Ethereum PoS parameters:

Parameter	Value
Minimum stake	32 ETH per validator
Slot time	12 seconds
Epoch	32 slots (6.4 minutes)
Finality	2 epochs ( $\approx 12.8$ minutes)
Active validators	$\approx 1,000,000+$ (as of 2025)
Slashing penalty	1/32 of stake (minimum)

The Merge was the largest live migration in blockchain history, proving PoS viability at scale.

## Definition: PBFT

**Practical BFT** (Castro & Liskov, 1999) is a consensus algorithm where a designated leader proposes a block, and all validators exchange signed votes in three rounds: *pre-prepare*, *prepare*, and *commit*. A block is finalized when  $\frac{2}{3}$  of validators agree.

### The three-phase protocol:

- 1 **Pre-prepare:** Leader broadcasts proposed block to all validators
- 2 **Prepare:** Each validator checks the proposal and broadcasts a “prepare” vote
- 3 **Commit:** Once  $\frac{2}{3}$  prepare votes are received, validators broadcast “commit” votes
- 4 **Finalized:** Once  $\frac{2}{3}$  commit votes are received, the block is **final**

**Trade-off:** Deterministic finality, but  $O(n^2)$  message complexity limits scalability to  $\sim 100$  validators.

---

PBFT provides instant finality but does not scale to large open networks — it is used in permissioned blockchains.

**Tendermint (Cosmos):** A BFT consensus engine optimized for blockchain:

- Combines PBFT-style voting with **stake-weighted validators**
- Finality in  $\sim 6$  seconds (single round of voting)
- Tolerates up to  $\frac{1}{3}$  Byzantine validators
- Powers the Cosmos ecosystem (100+ interconnected blockchains)

**Other BFT variants in production:**

Algorithm	Used By	Key Feature
Tendermint	Cosmos	Instant finality, modular design
HotStuff	Aptos, formerly Libra	Linear message complexity $O(n)$
Istanbul BFT	Hyperledger Besu	Enterprise Ethereum variant
Federated BFT	Stellar	Trust-based quorum slices

Modern BFT algorithms reduce message complexity from  $O(n^2)$  to  $O(n)$ , enabling larger validator sets.

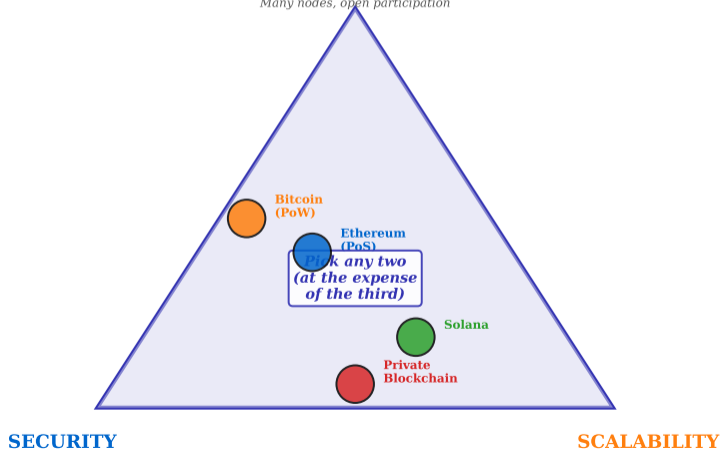
## Consensus Mechanism Comparison

Property	Proof of Work (PoW)	Proof of Stake (PoS)	BFT Variants (PBFT/Tendermint)
Resource consumed	Electricity (computational)	Staked capital (economic)	Network messages (communication)
Sybil resistance	Cost of hardware + electricity	Cost of acquiring stake	Permissioned membership
Fault tolerance	Up to 50% hash power	Up to 33% of stake	Up to 33% of nodes
Finality	Probabilistic (~6 blocks)	Faster probabilistic (~2 epochs)	Deterministic (immediate)
Energy use	Very High	Very Low	Very Low
Throughput (TPS)	~7 TPS	~30 TPS	~1,000+ TPS
Decentralization	High (permissionless)	High (permissionless)	Low (permissioned)
Example	Bitcoin	Ethereum (post-Merge)	Hyperledger Fabric

## The Blockchain Trilemma

### DECENTRALIZATION

*Many nodes, open participation*



System	Decentral.	Security	Scalability	Trade-off
Bitcoin	High	High	Low	7 TPS; no smart contracts
Ethereum L1	High	High	Medium	30 TPS; gas fees spike
Solana	Medium	Medium	High	65,000 TPS; requires powerful hardware
Hyperledger	Low	High	High	Permissioned; not open

### Emerging solutions to the trilemma:

- **Layer 2 scaling:** Move transactions off-chain (rollups, payment channels) while inheriting L1 security
- **Sharding:** Split the network into parallel chains that share security
- **Modular blockchains:** Separate execution, consensus, and data availability

The trilemma drives blockchain innovation: Layer 2 solutions and modular architectures aim to achieve all three properties.

Every Bitcoin block contains:

Field	Purpose
<b>Previous block hash</b>	Links this block to the chain (hash pointer from Lesson 3.1)
<b>Merkle root</b>	Single hash summarizing all transactions in the block
<b>Timestamp</b>	When the miner began working on the block
<b>Difficulty target</b>	How many leading zeros the block hash must have
<b>Nonce</b>	The variable miners iterate to find a valid hash
<b>Transaction list</b>	All transactions included in this block

Why each field matters:

- **Previous hash** → tamper-evident chain (change one block, invalidate all successors)
- **Merkle root** → efficient verification of any single transaction
- **Nonce + target** → Proof of Work consensus

---

The block header is only 80 bytes, yet it cryptographically commits to the entire block's contents and its position in the chain.

## From creation to finality in six steps:

- 1 **Create:** Alice signs a transaction (“Pay Bob 0.1 BTC”) with her private key
- 2 **Broadcast:** The signed transaction is sent to the P2P network
- 3 **Mempool:** Each node places the transaction in its “waiting room” (mempool)
- 4 **Mining:** A miner selects transactions from the mempool, builds a candidate block, and begins the PoW puzzle
- 5 **Validation:** When a valid block is found, it propagates across the network; each node independently verifies the PoW and all transaction signatures
- 6 **Confirmation:** Each subsequent block built on top of Alice’s block adds one “confirmation”; after 6 confirmations (~60 minutes), the transaction is considered final

**Key insight:** Every concept from Lesson 3.1 (hashing, Merkle trees, signatures) and this lesson (PoW, difficulty, longest chain) works together in this process.

---

A Bitcoin transaction touches every component we have studied: ECDSA signatures, SHA-256 hashing, Merkle trees, and Proof of Work.

## Pure PoS has unresolved theoretical and practical weaknesses:

- **Nothing-at-stake:** without slashing, validators can sign multiple competing chains for free — no cost to back every fork
- **Long-range attacks:** an attacker who acquires *old* validator keys can fork from genesis; PoW's cumulative work makes this infeasible
- **Weak subjectivity:** new nodes joining the network must trust a recent checkpoint — breaks pure permissionless decentralization

## Centralization in practice (Ethereum, post-Merge):

- Lido staking pool controls 32–35% of staked ETH (*Dune Analytics / Lido dashboards, 2024*)
- Vitalik Buterin: “Should Ethereum Fork if Lido Hits 50%?” (*vitalik.eth.limo, 2024*) — discusses social-layer response if a single staking entity dominates
- US-regulated exchanges (Coinbase + Kraken + Binance.US) jointly account for  $\approx 40\%$  of validator share (*Rated Network / Etherscan, 2024*)

---

Slashing makes nothing-at-stake economically irrational, but cartelization risk remains the open critique.

## The standard data source:

- Cambridge CCAF Bitcoin Electricity Consumption Index (*ccaf.io*, 2024) — bottom-up estimate from miner hardware mix and network hashrate

## Common misleading metric — and the counter-counter:

- “kWh per transaction” is misleading: Bitcoin secures the cumulative ledger value, not just the marginal transaction
- *But*: this argument doesn't justify the absolute magnitude — the network still consumes electricity comparable to a mid-sized country

## The mix is more sustainable than headlines suggest:

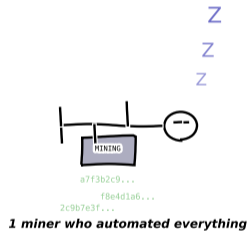
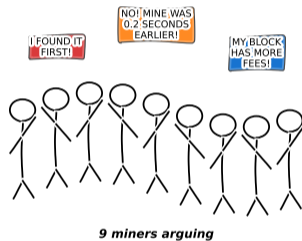
- ~50%+ sustainable energy share (*CCAF mining map*, 2024) (renewables + nuclear)
- Stranded-energy use cases: flared natural gas in the Texas Permian Basin, hydroelectric overflow in the Pacific Northwest — monetizing energy that would otherwise be wasted

**The honest bottom line:** absolute emissions remain significant ( $\approx 70\text{--}80$  Mt CO<sub>2</sub>/year (*CCAF / Digiconomist*, 2024)), comparable to a small industrialized country. PoW's security model has a real environmental cost that cannot be marketed away.

---

Engineering ethics: cite the methodology (CCAF), not the headline; acknowledge magnitude even when defending the design.

## Proof of Work: A Documentary



**The longest chain rule: Eventually one of them will mine the next block, and all the arguing becomes moot.**

Sometimes the best way to remember a concept is to laugh about it.

- 1 The **Byzantine Generals Problem** formalized the challenge of reaching agreement when some participants may be malicious; BFT requires  $n > 3f$
- 2 **Proof of Work** solves consensus through computational puzzles: miners burn electricity, making cheating economically irrational
- 3 **Difficulty adjustment** ensures a stable block production rate regardless of how much hash power enters or leaves the network
- 4 **Proof of Stake** replaces electricity with staked capital; validators are economically punished (slashed) for dishonest behavior
- 5 **BFT variants** (PBFT, Tendermint) achieve instant deterministic finality but trade off decentralization
- 6 The **blockchain trilemma** states that decentralization, security, and scalability cannot all be maximized simultaneously — every design makes trade-offs
- 7 **Bitcoin's architecture** combines hash pointers, Merkle trees, digital signatures, and Proof of Work into a single coherent system

---

Consensus mechanisms are the bridge between cryptographic primitives and trustless financial systems.

**This lesson:** We explored how distributed networks reach agreement — from the Byzantine Generals Problem through PoW, PoS, and BFT — and examined the trade-offs captured by the blockchain trilemma.

### Key vocabulary:

- Byzantine Generals Problem
- Byzantine Fault Tolerance
- Proof of Work (PoW)
- Nonce / difficulty target
- Block reward / halving
- 51% attack
- Proof of Stake (PoS)
- Validator / slashing
- Finality (probabilistic vs. deterministic)
- PBFT / Tendermint
- Blockchain trilemma
- Longest chain rule

**Next lesson (M3L3):** *Blockchain Architecture and Data Structures* — How are blocks actually structured? How do nodes synchronize? We build a complete picture of how the Bitcoin and Ethereum networks operate at the protocol level.

---

Review: Can you explain why PoW is secure against a 40%-attacker but not a 51%-attacker?

### Attempt these before turning the page.

- 1 [Understand] State the Byzantine Generals problem in one sentence. Why is it the foundational problem for distributed consensus?
- 2 [Apply] Ethereum PoS has  $\sim 1\text{M}$  validators, each staking 32 ETH ( $\sim \$3,200$  at  $\$100/\text{ETH}$ ). An attacker wants 33% to halt finality. Compute the dollar cost of attack. How does this compare to Bitcoin PoW's 51% attack cost (roughly  $\$50\text{B}$  mining hardware + electricity)?
- 3 [Evaluate] PoS critics argue “the rich get richer” (wealthier stakers earn more rewards). Is this different from PoW's “cheap-electricity regions dominate”? Argue one side with a criterion.

---

Solutions hidden unless `\solutionstrue` is set before compiling.