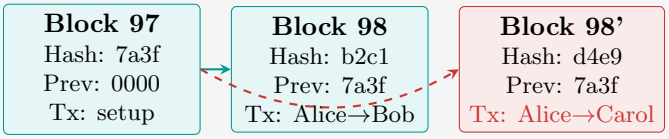
Pre-Class Discovery Handout: Blockchain Fundamentals

Activity 1: The Double-Spend Attack

Scenario: Alice has 1 BTC. She sends it to Bob AND to Carol simultaneously. Without a central authority, who gets paid?



Q1: Without a blockchain, who gets paid — Bob, Carol, or both? Explain why.

Q2: How does the blockchain resolve this conflict? (Hint: which transaction gets into a block first?)

Q3: Why does the hash chain make it impractical to rewrite history?

Activity 2: Hash It Out

Scenario: Define a simplified hash function: $H(\text{text}) = \left(\sum \text{ASCII values}\right) \bmod 256$.

ASCII reference: A=65, B=66, C=67, ..., K=75, L=76, O=79, 1=49, 2=50.

Q1: Compute $H(\text{“BLOCK1”}) = (66 + 76 + 79 + 67 + 75 + 49) \bmod 256 = ?$

Q2: Compute $H(\text{“BLOCK2”})$. How different is it from $H(\text{“BLOCK1”})$?

Q3: Change one letter: compute $H(\text{“ALOCK1”})$. Why does even a small change matter for blockchain immutability?

Activity 3: Consensus Comparison

Scenario: Two dominant consensus mechanisms secure blockchains today. Fill in the blank cells.

Dimension	Proof of Work	Proof of Stake
Energy consumption		
Security model		
Centralization risk		
Finality speed		
Barrier to entry		

Q1: Fill in the comparison table above with concise descriptions.

Q2: Which consensus mechanism is better for the environment? Why?

Q3: Can a consensus mechanism be both fast and fully decentralized? Explain the tradeoff.

Solutions

Activity 1: The Double-Spend Attack

A1: Without a blockchain, both Bob and Carol could believe they were paid, because there is no shared ledger to detect the duplicate spend. A central intermediary (like a bank) normally prevents this, but in a peer-to-peer system the conflict goes unresolved.

A2: The blockchain resolves the conflict through mining: only one transaction can be included in the next valid block. Whichever transaction a miner includes first becomes the canonical record; the other is rejected as a double-spend attempt. The longest-chain rule ensures network-wide agreement.

A3: Each block's hash depends on the previous block's hash, creating a cryptographic chain. Changing a transaction in Block 98 would alter its hash, which would invalidate Block 99's "Prev" pointer, and every subsequent block. An attacker would need to re-mine the entire chain from that point, which requires more computational power than the rest of the network combined.

Activity 2: Hash It Out

A1: $H(\text{“BLOCK1”}) = 66 + 76 + 79 + 67 + 75 + 49 = 412$. Then $412 \bmod 256 = 156$.

A2: $H(\text{“BLOCK2”}) = 66 + 76 + 79 + 67 + 75 + 50 = 413$. Then $413 \bmod 256 = 157$. The difference is only 1, but in real cryptographic hash functions (SHA-256), changing a single character produces a completely different output — the avalanche effect.

A3: $H(\text{“ALOCK1”}) = 65 + 76 + 79 + 67 + 75 + 49 = 411$. Then $411 \bmod 256 = 155$. Even a one-character change alters the hash. In a blockchain, this means any tampering with a block's contents changes its hash, breaking the chain of "Prev" pointers and making forgery immediately detectable.

Activity 3: Consensus Comparison

A1: Energy consumption: PoW is extremely high (mining hardware runs continuously); PoS is minimal (no intensive computation). Security model: PoW relies on computational cost (51% of hash power to attack); PoS relies on economic stake (51% of staked tokens). Centralization risk: PoW trends toward mining pools with cheap electricity; PoS trends toward large token holders. Finality speed: PoW takes minutes (e.g. Bitcoin ∼10 min); PoS can achieve finality in seconds. Barrier to entry: PoW requires expensive hardware (ASICs); PoS requires purchasing and locking tokens.

A2: Proof of Stake is far better for the environment. PoW consensus (Bitcoin) consumes as much electricity as some countries, whereas PoS (Ethereum post-Merge) reduced energy use by ∼99.95% because validators do not perform energy-intensive computation.

A3: This is the blockchain trilemma: a system can optimize for at most two of decentralization, security, and scalability (speed). Fast finality typically requires fewer validators or a delegated model, which sacrifices full decentralization. Conversely, maximum decentralization (thousands of independent validators) slows consensus because more nodes must agree.

