Blockchain

Problem: We want a distributed transaction ledger so that **no single entity has control over the ledger**

- Global transaction ledger:
  - Contains all transactions between all participants
  - Distributed to all participants (=> open)
  - Can be sent by anyone, and therefore requires some verifiability
- Solution: Use cryptographic techniques so that participants can verify the ledger

Global Transaction Ledger

Sender	Receiver	Bitcoin amount	
Alice	Bob	0.31	Block 1
Carol	Bob	1.21	
Alice	Bob	0.4	
Bob	Alice	0.532	Block 2
Carol	Bob	0.01	Block 4
Bob	Carol	0.01	
...	...	...	
...	...	…	

- Everyone agrees on the same ledger
  - Update each other on new transactions
- Divided into blocks, 1 block every 10 minutes
- Can someone lie?
- e.g. “Alice: Block 1 has no transactions!”

Global Transaction Ledger

Three types of threats against the ledger's integrity:

Add: People refuse to add a real transaction to the ledger

Modify: Someone modifies a transaction in the ledger

- This is easiest to fix: Simply have the participants sign all transactions, and include the signature in the ledger

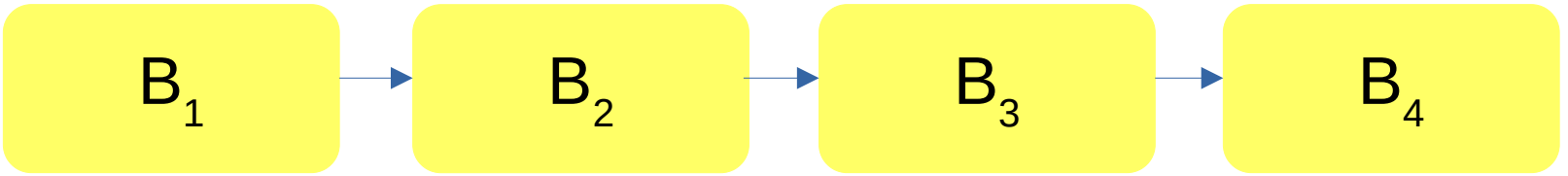
Delete: Someone removes a transaction from the ledger

- To prevent those, we use a *proof of work* to safeguard blocks

Two types of participants in a blockchain system:

- Users: Signs their transactions and announces them
- Miners: Helps add the transactions to the ledger by generating the proof of work (technically, miners can also be users)

Blockchain



$$B_2 = <T_2, h(B_1), P_2(h(B_1))>$$

$$B_3 = <T_3, h(B_2), P_3(h(B_2))>$$

$$B_4 = <T_4, h(B_3), P_4(h(B_3))>$$

Transactions in this block

Proof of work, based on hash and very hard to compute

Attacker claim: actually, B_1 should be B_1'

- The attacker needs to change B_1, which changes B_2, which...
- The attacker needs to generate 3 new proofs of work, because the network accepts the longest chain
- Trying to generate those essentially triggers a race: other miners are generating P_5, P_6, P_7...

Proof of Work

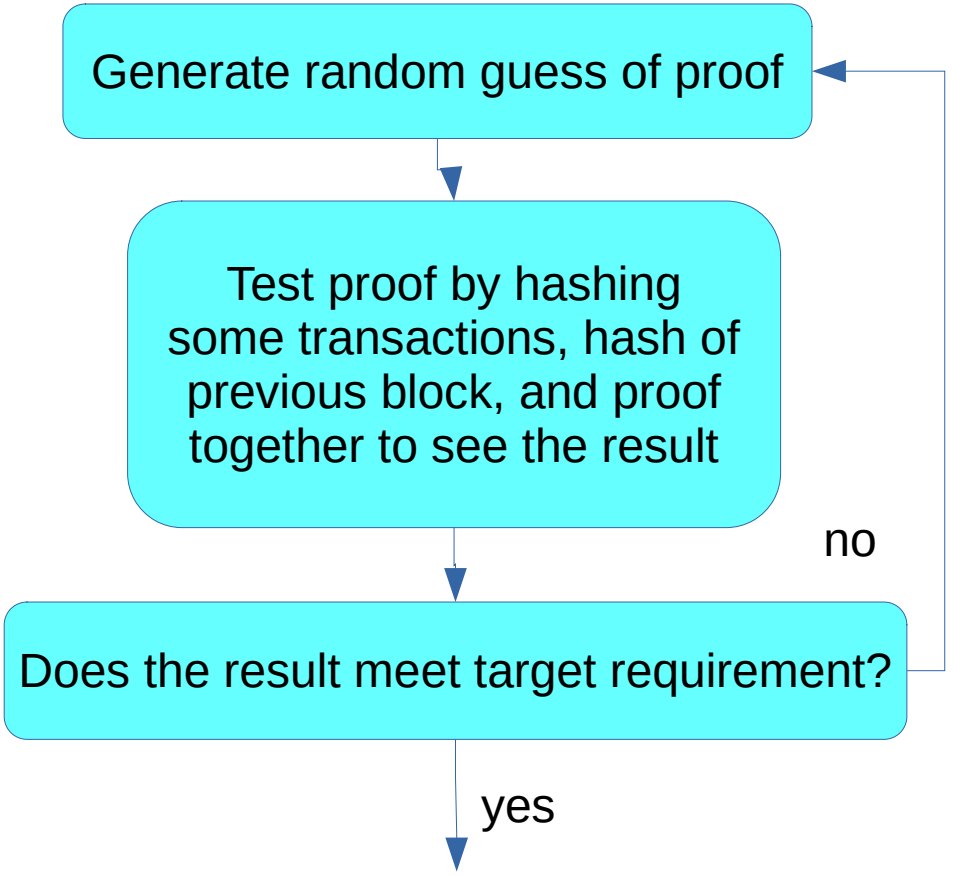
- Challenge: Given C, how can we find P such that

h(C||P) *ends with 32 zero bits?*

- Cryptographic hash: Best way = Brute force (one success per 2^{32} hashes on average)
- C is the Content and P is the Proof of Work: If Alice sends C to Miner and Miner sends P back to Alice, Miner has “proven” that they did the work of many hashes

Bitcoin

Miner algorithm:



Show it; the system automatically credits the miner with a reward

Bitcoin

- Reward
  - On average, 1 proof (= 1 block) will be found every 10 minutes – the proof difficulty automatically adjusts
  - However, these proofs can correspond to empty transaction blocks...
  - To prevent this, transactions attach a transaction fee, which is a payment to whichever miner includes that transaction into a block

Bitcoin

- Scaling issues
  - New miners need to download full (growing) global transaction ledger
  - Each block can only store about 6,000 transactions (=10 transactions per second)
  - Changes require soft/hard fork
- Transaction delay
  - Need to wait for blocks (several to be safe)
  - Each block is 10 minutes

Ethereum

- Can be thought of as “Bitcoin+ with smart contracts” – transaction delay is reduced, more transactions can be supported
- Smart contracts are programs that Ethereum nodes execute in the same way

Attacks galore

- Writing a smart contract correctly is very difficult
- DAO hack ($150 mil): the vulnerable program was “If you want to empty your account, I will check your account balance, then give you money, then set your account to 0” -> vulnerable to a TOCTTOU-like attack
- Axie Infinity hack ($615 mil): PDF file containing trojans
- Key-stealing attacks, “Key-stealing attacks”
- Malware that targets cryptocurrency software specifically
- Attraction to attacks due to being completely irreversible (except that one time)