Goals for today:

(1) block-chains: key insights / reusable design techniques

(2) analyze the system from the perspective of the of various topics we've studied this term

- Centralized vs. Decentralized
- Distributed Agreement (e.g., Paxos vs. BitCoin's heuristic on who will extend the chain)
- Consistency Models / CAP
- Incremental Scalability (vs. blockchain difficulty)
- Hash functions and Merkel Trees

… generous topic we will run out of time

Block-chains:
a technological solution to aggregate a *public tamper-proof ledger* form untrusted sources

Perhaps with ability to enable users to selectively reveal information

What matters?

- System resilience: fault-tolerance / robustness to attacks
- Type of guarantees: 100% certainty vs. probabilistic
- Ability to provide (some form of) privacy / anonymity
- *Bootstrapping the system*
- Costs: transactions, operating the system

Annualized Total Bitcoin Footprints

Carbon Footprint	Electrical Energy	Electronic Waste
114.06 Mt CO2	204.50 TWh	33.70 kt
Comparable to the carbon footprint of **Czech Republic**.	Comparable to the power consumption of **Thailand**.	Comparable to the small IT equipment waste of **the Netherlands**.

$20 billion / year !

Single Bitcoin Transaction Footprints

Carbon Footprint	Electrical Energy	Electronic Waste
1185.15 kgCO2	2124.84 kWh	350.10 grams
Equivalent to the carbon footprint of **2,626,704** VISA transactions or **197,525** hours of watching Youtube.	Equivalent to the power consumption of an average U.S. household over **72.83** days.	Equivalent to the weight of **2.13** iPhones 12 or **0.71** iPads. (Find more info on e-waste here.)

Source: https://digiconomist.net/bitcoin-energy-consumption/, April 2022

… depends on the application

- Ledger of financial transactions (*coin)
- Medical records
- Land records
- Certify provenance – goods, green electricity

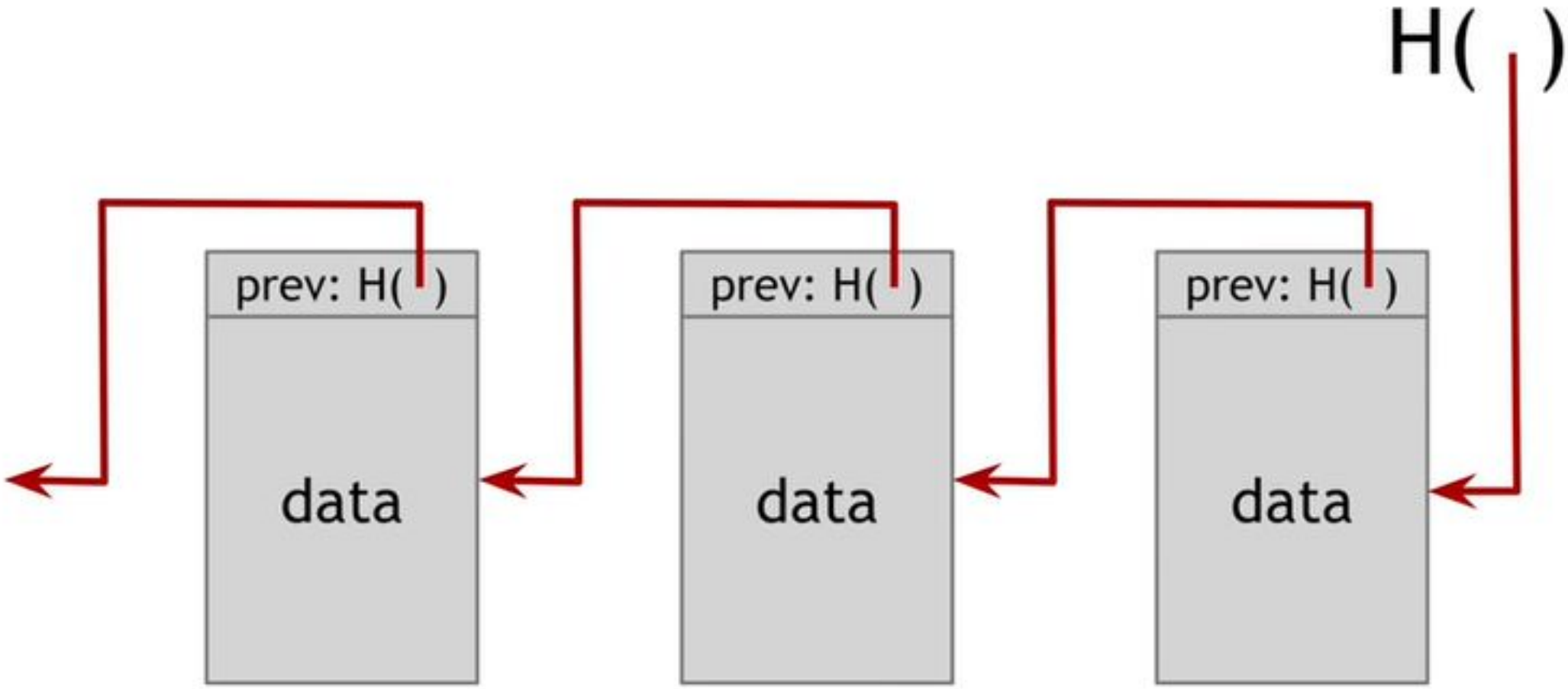
What matters?

- System resilience
- Type of guarantees
- Cost
- Privacy / anonymity
- Bootstrapping

The ‘sauce’

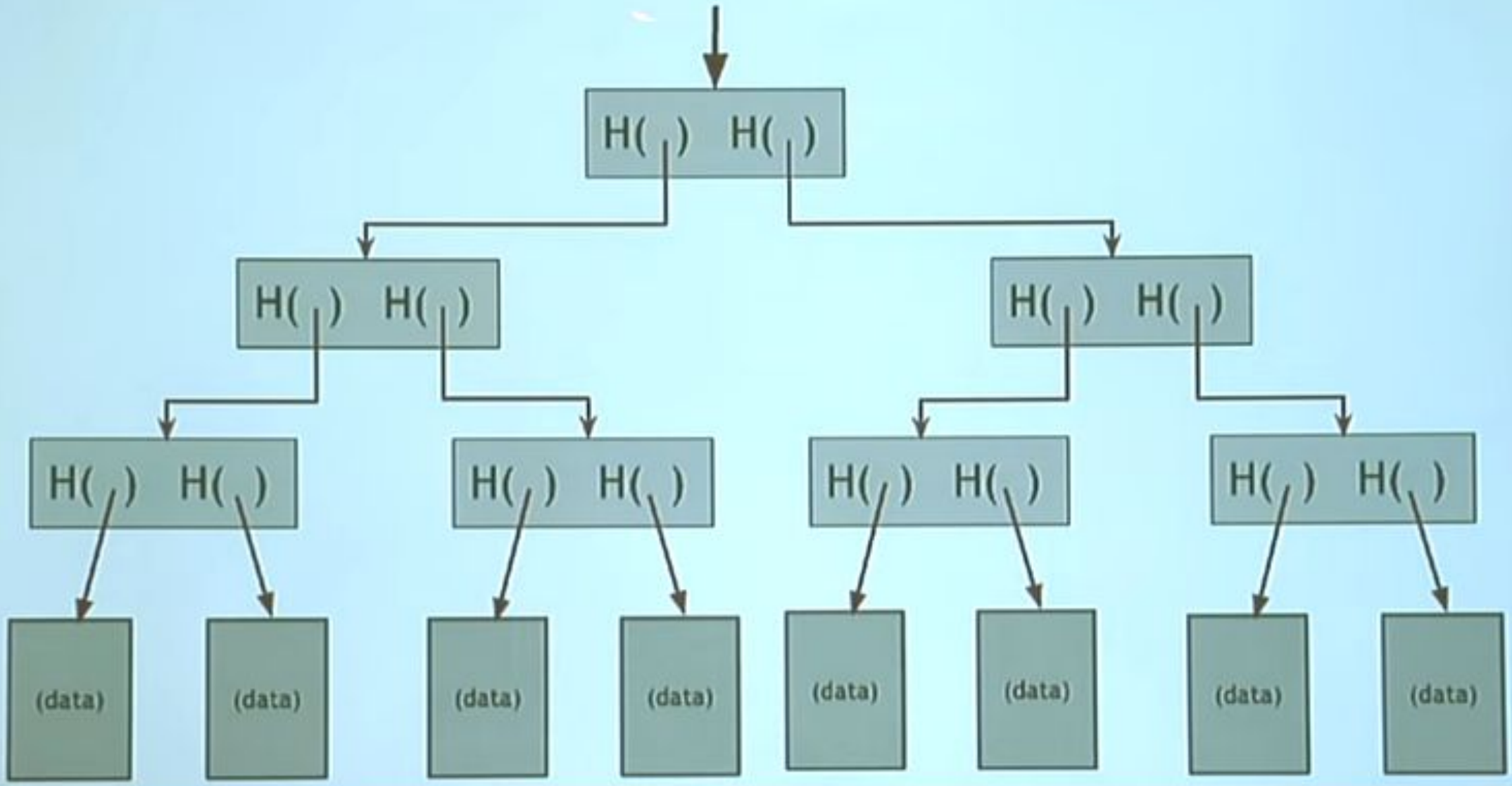
Tamper-evident [append-only] data structures

Tamper-evident [append-only] data structure



use case: tamper-evident log

binary tree with hash pointers = “Merkle tree”



Get rich in 3 simple steps with GoofyCoin! ☺️

GoofyCoin

Goofy can create new coins

signed by pk_{Goofy}

CreateCoin [uniqueCoinID]

Goofy's 3-step scheme to get rich:

1.) Mint one million coins

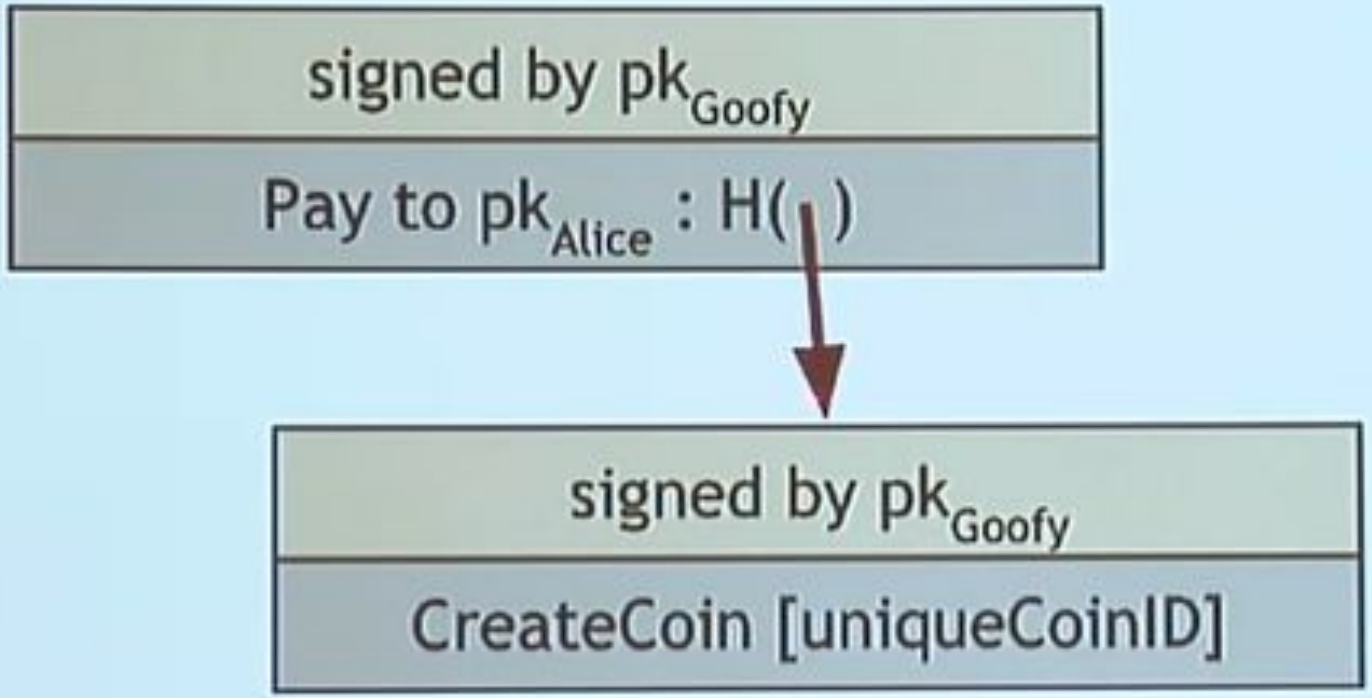
2.) Give away 10,000 of them to friends for free.

3.) Buy 1,000 of them back for $1 each

What's Goofy's net worth now?

$999,000 (in coins)

A coin's owner can spend it.



The recipient can pass on the coin again.

signed by pk_{Alice}

Pay to pk_{Bob} : H( )

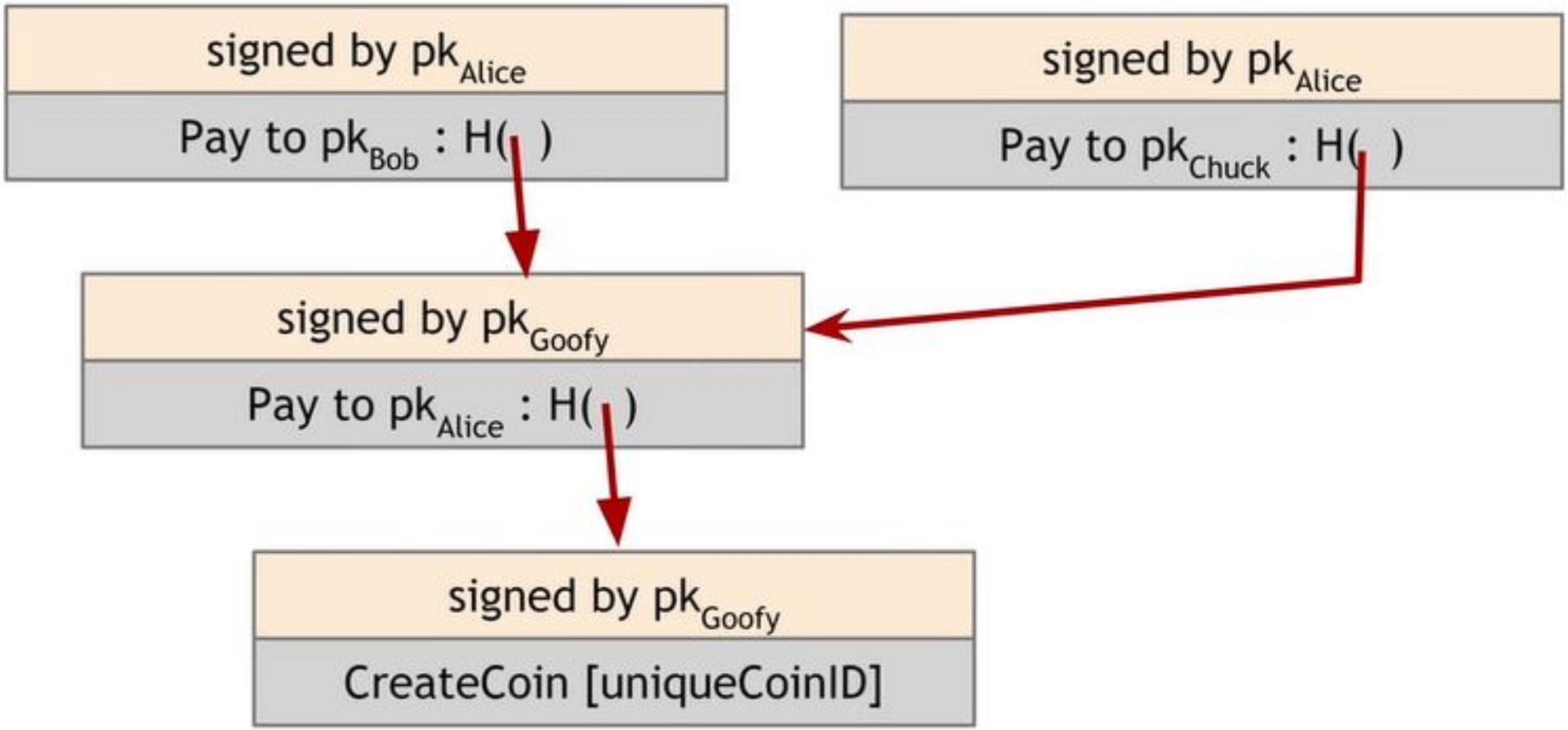
signed by pk_{Goofy}

Pay to pk_{Alice} : H( )

signed by pk_{Goofy}

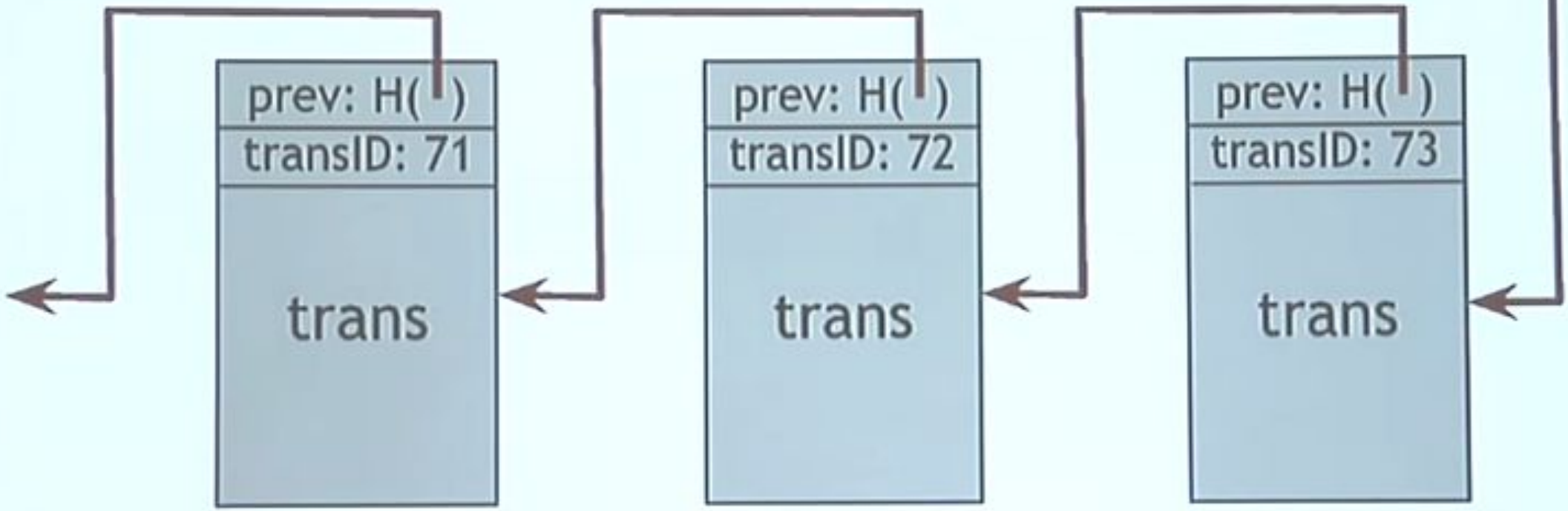
CreateCoin [uniqueCoinID]

double-spending attack



Scrooge publishes a history of all transactions
(a block chain, signed by Scrooge)

H( )



optimization: put multiple transactions in the same block

PayCoins transaction consumes (and destroys) some coins, and creates new coins of the same total value

transID: 73 type:PayCoins

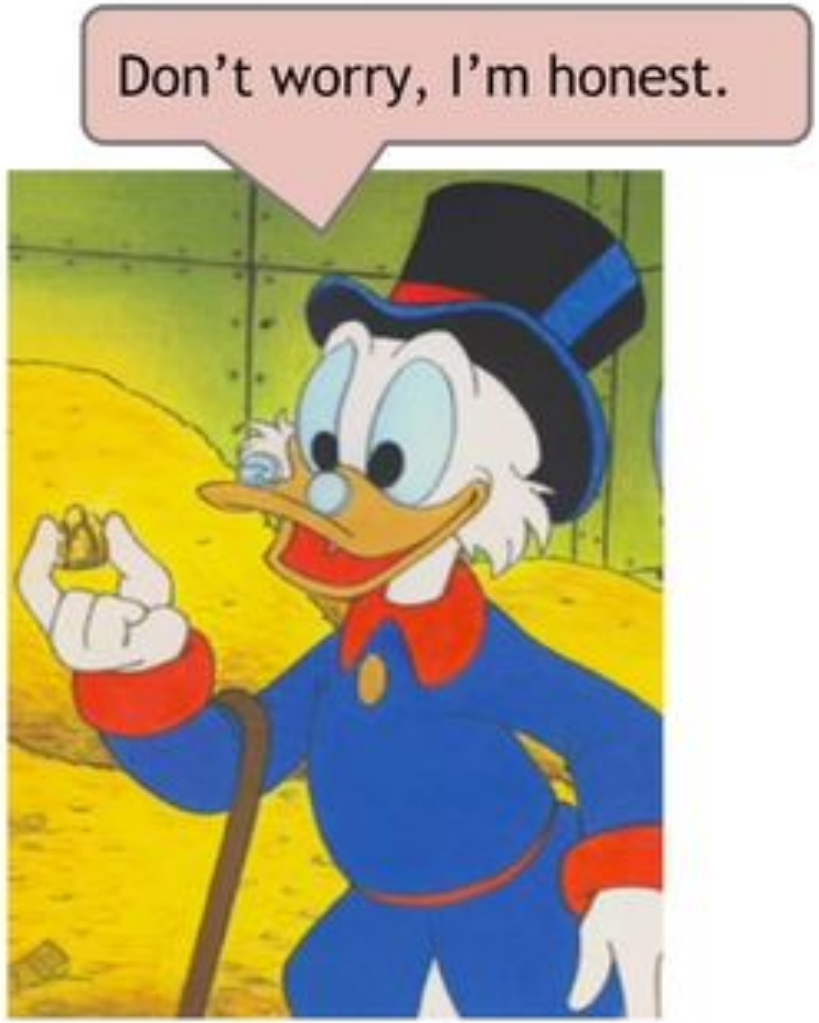
consumed coinIDs:
68(1), 42(0), 72(3)

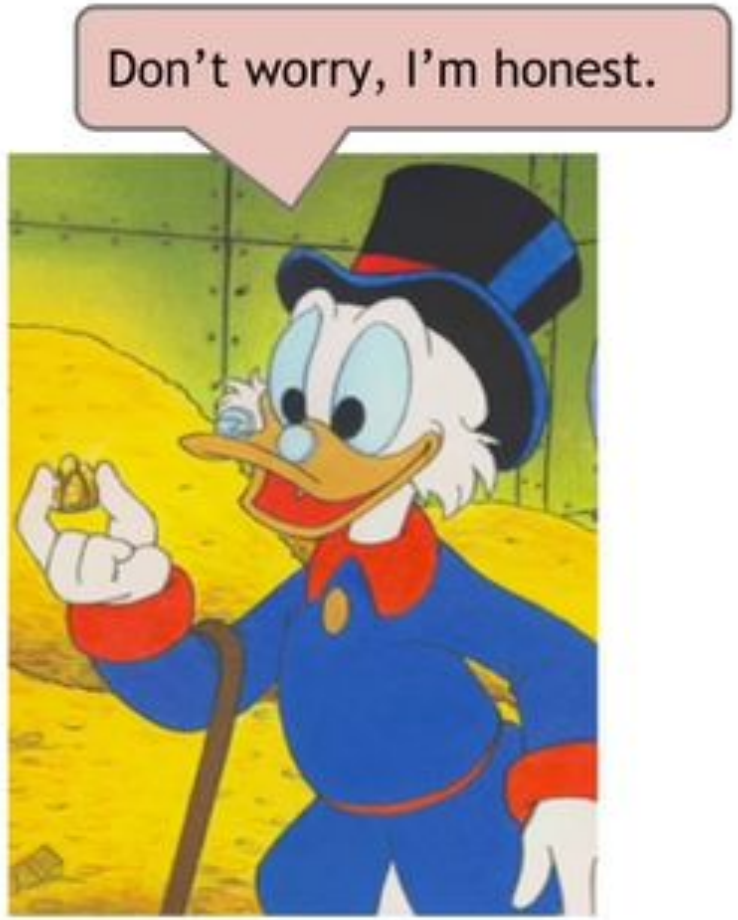
coins created

num	*value*	*recipient*
0	3.2	0x...
1	1.4	0x...
2	7.1	0x...

signatures

Valid if:
-- consumed coins valid,
-- not already consumed,
-- total value out = total value in, and
-- signed by owners of all consumed coins


Don’t worry, I’m honest.



Crucial question:

Can we descroogify the currency, and operate without any central, trusted party?

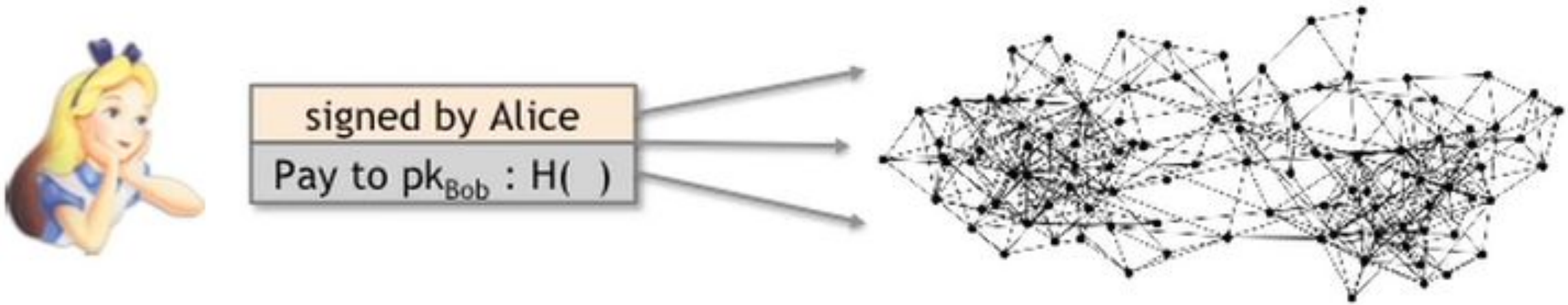
Key challenge (for any distributed e-cash system)

Distributed consensus

Next: how to decentralize ScroodgeCoin

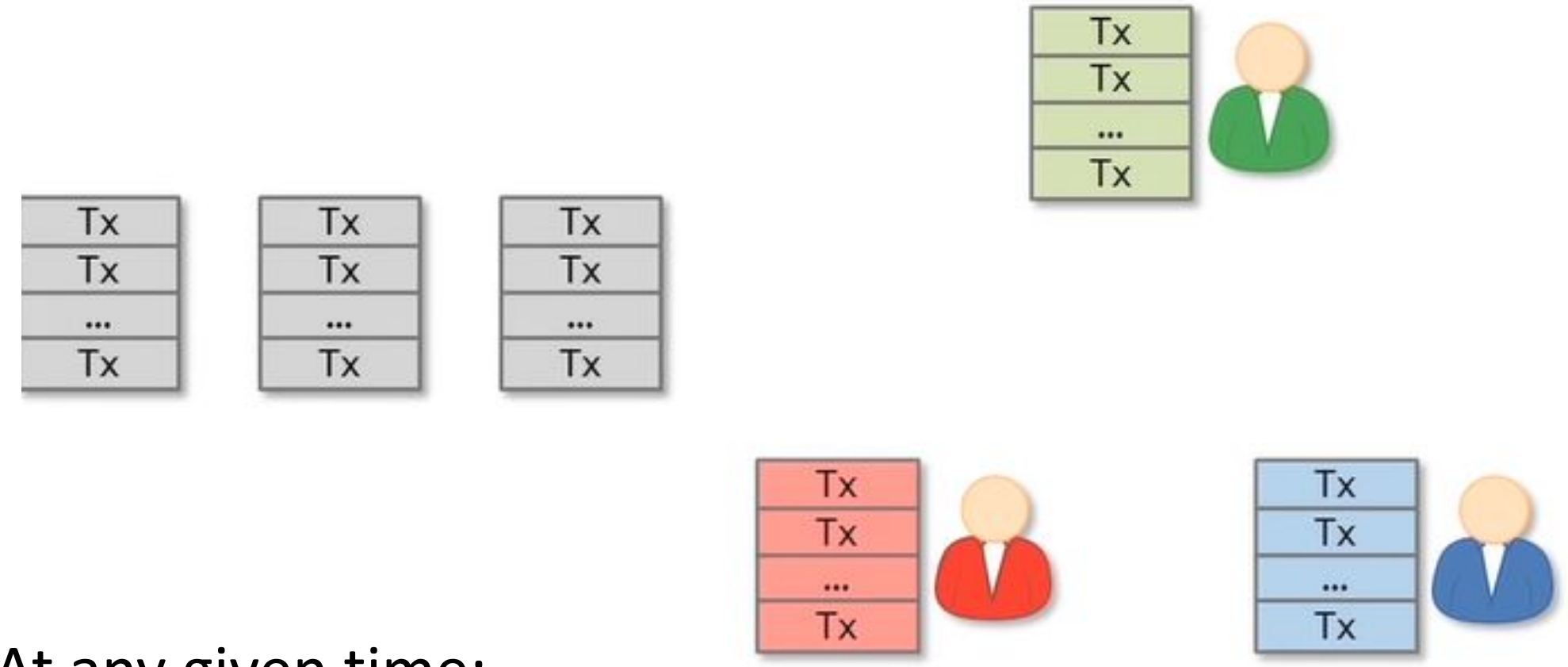
Bitcoin is a peer-to-peer system

When Alice wants to pay Bob:
she broadcasts the transaction to all Bitcoin nodes



Note: Bob's computer is not in the picture

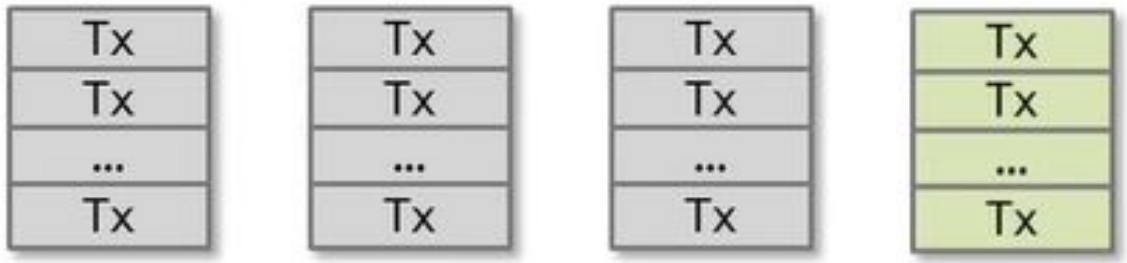
How consensus could work in Bitcoin



At any given time:

- all nodes have a sequence of (blocks of) transactions they have reached agreement on
- each node has a set of outstanding transactions if has heard about

How consensus could work in Bitcoin



OK to select any valid block, even if proposed by only one node

Could run any of the distributed consensus algorithms we have discussed!

Consensus: Impossibility result

Fischer-Lynch-Paterson: consensus impossible to guarantee even with a single faulty node in an asynchronous system.

What's possible though? Paxos/RAFT

correct (safe) and finish most of the time (mostly live)

transactions are final (i.e., durable)

Byzantine failures: 3F+1 nodes needed to support F faulty nodes (B-PAXOS)
Fail-stop / fail-restart: 2F+1 nodes needed to support F faulty nodes (PAXOS, RAFT)

Consensus in BitCoin - different model

probabilistic transaction finality (system converges)

Traditional consensus protocols require identities

- Pragmatic reasons
- Security: assume less than ⅓ (B-PAXOS) are malicious

Bitcoin context is different

- P2P network: open to anyone / low barrier to enter
- Pseudo anonymity is a goal

Consequence

- difficult to enforce/use identities (Sybil attack)
- hard-identities are a non-goal

Aspects of decentralization in Bitcoin

Peer-to-peer network:

open to anyone, low barrier to entry

Mining:

open to anyone, but inevitable concentration of power

often seen as undesirable

Updates to software:

core developers trusted by community, have great power

Weaker assumption: select random node

Analogy: lottery or raffle

When tracking & verifying identities is hard, we give people tokens, tickets, etc.

Now we can pick a random ID & select that node

Key idea: implicit consensus

In each round, random node is picked

This node proposes the next block in the chain

Other nodes implicitly accept/reject this block

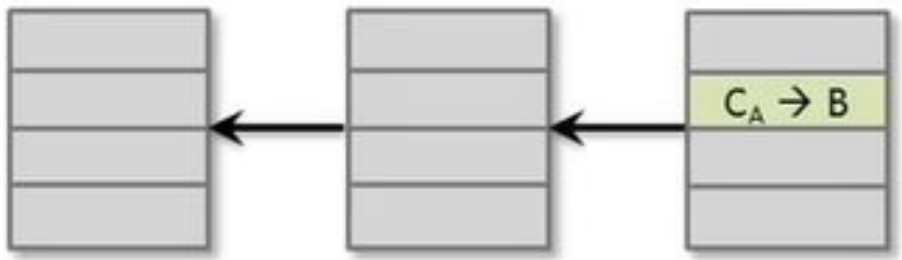
- by either extending it
- or ignoring it and extending chain from earlier block

Every block contains hash of the block it extends

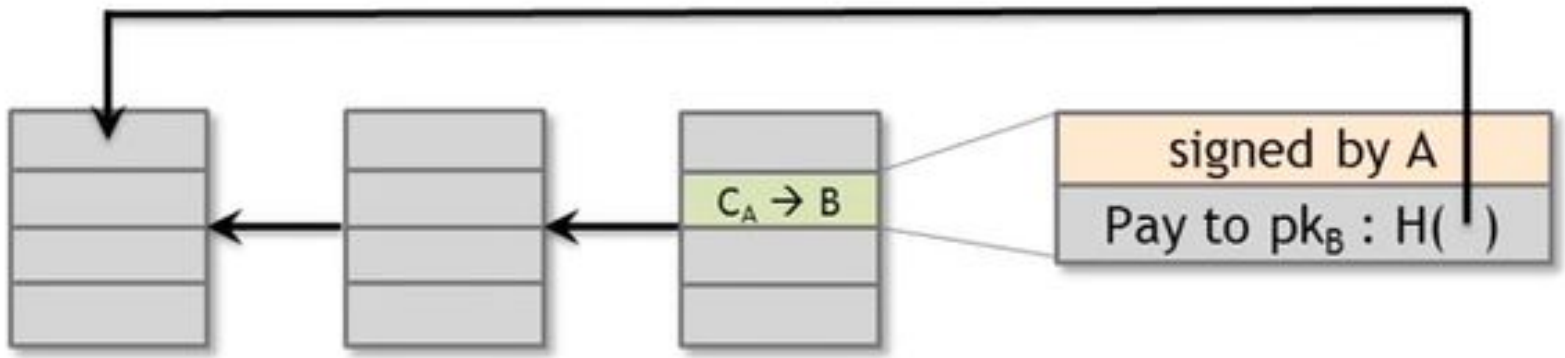
Consensus algorithm (simplified)

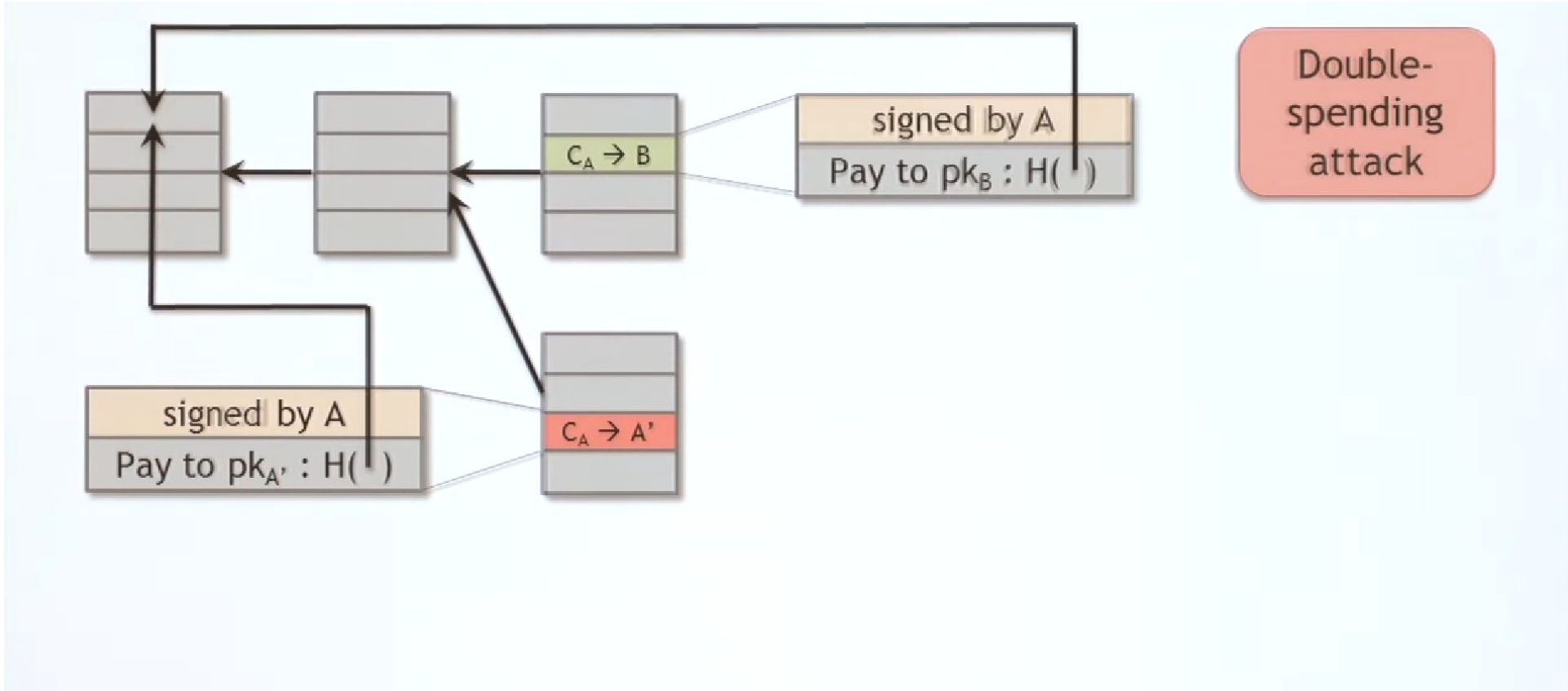
1. New transactions are broadcast to all nodes
2. Each node collects new transactions into a block
3. In each round a random node gets to broadcast its block
4. Other nodes accept the block only if all transactions in it are valid (unspent, valid signatures)
5. Nodes express their acceptance of the block by including its hash in the next block they create

What can a malicious node do?

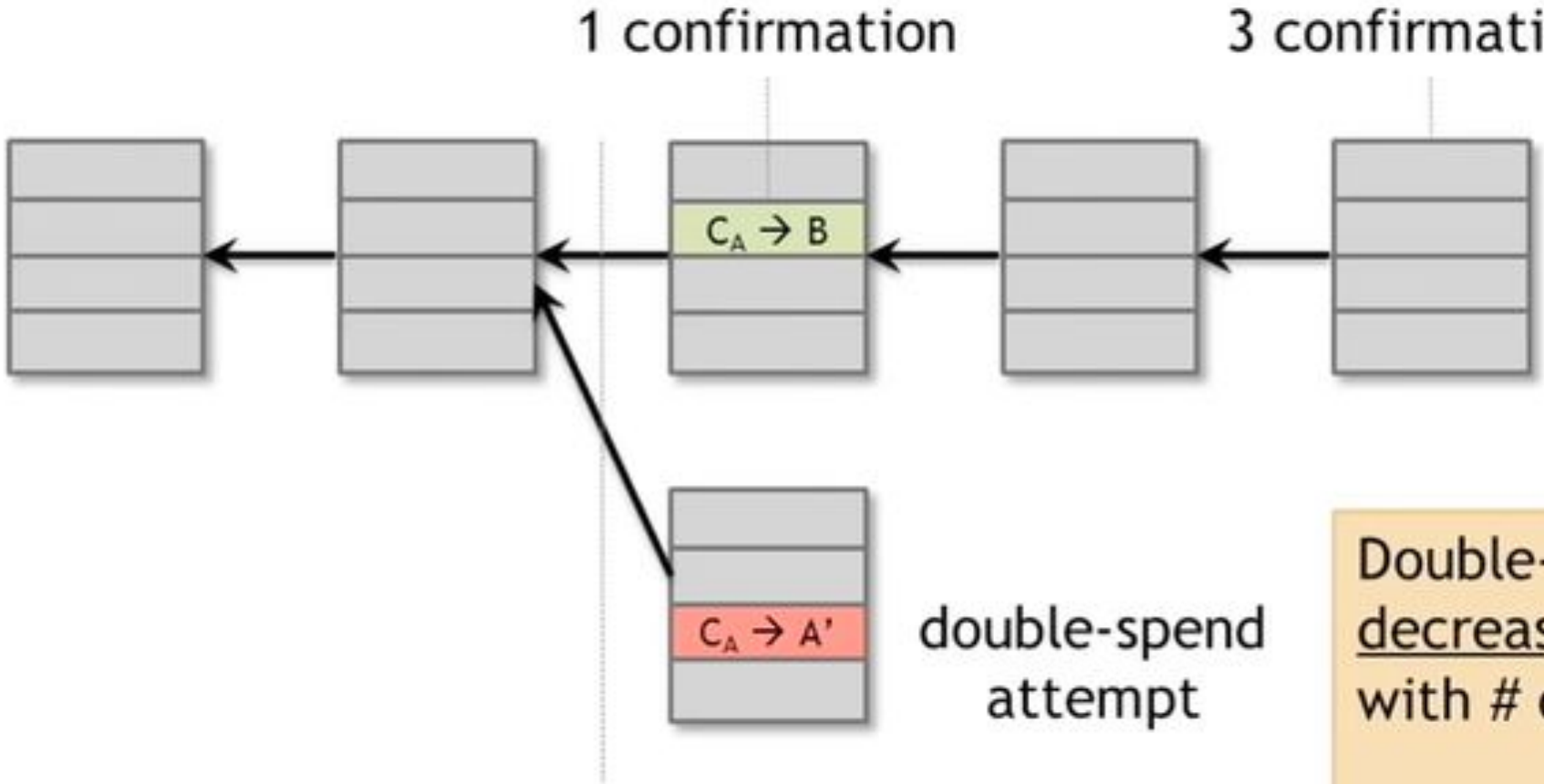


What can a malicious node do?




signed by A
Pay to pk_B : H( )
C_A → B
signed by A
Pay to pk_A' : H( )
C_A → A'
Double-spending attack

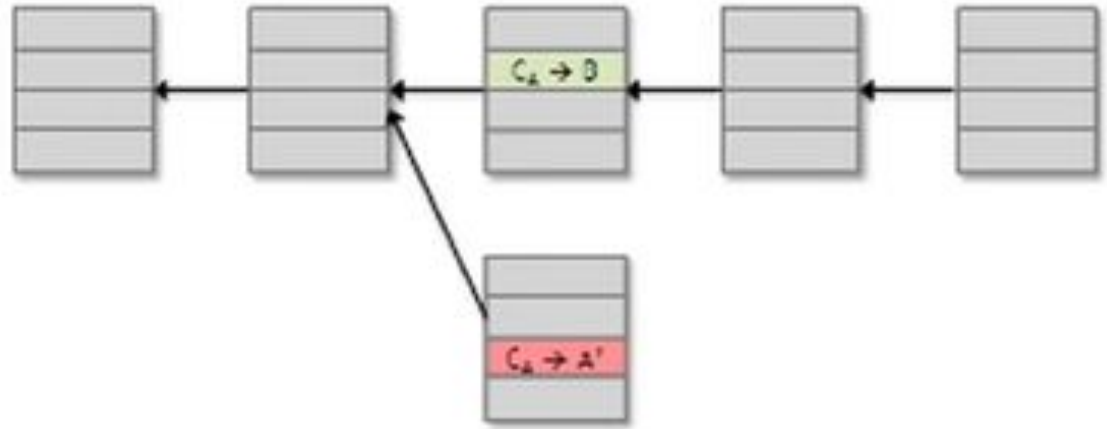
From Bob the merchant's point of view



Double-spend probability decreases exponentially with # of confirmations

Most common heuristic:
6 confirmations

Recap



Protection against invalid transactions is cryptographic, but enforced by consensus

Protection against double-spending is purely by consensus

You're never 100% sure a transaction is in consensus branch. Guarantee is probabilistic

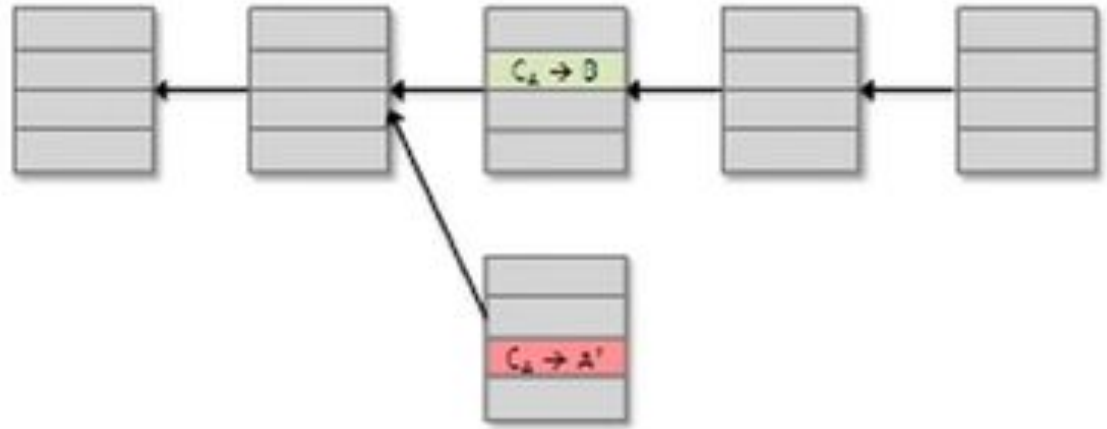
Goals:

- block-chains: key insights / reusable design techniques
- analyze system from the perspective of the of various topics we've studied this term
  - Centralized vs. Decentralized Distributed System
  - Distributed Agreement (e.g., Two Phase Commit vs Proof of Work for double spending)
  - Byzantine Fault-Tolerance
  - Consistency Models
  - CAP Theory
  - Incremental Scalability (vs. blockchain difficulty)
  - Node Join/Leave
  - Hash functions and Merkel Trees
- … generous topic we will run out of time

Goals:

- block-chains: key insights / reusable design techniques
- analyze system from the perspective of the of various topics we've studied this term
  - Centralized vs. Decentralized Distributed System
  - Distributed Agreement (e.g., Two Phase Commit vs Proof of Work for double spending)
  - Byzantine Fault-Tolerance
  - Consistency Models
  - CAP Theory
  - Incremental Scalability (vs. blockchain difficulty)
  - Node Join/Leave
  - Hash functions and Merkel Trees
- … generous topic we will run out of time

Recap



Protection against invalid transactions is cryptographic, but enforced by consensus

Protection against double-spending is purely by consensus

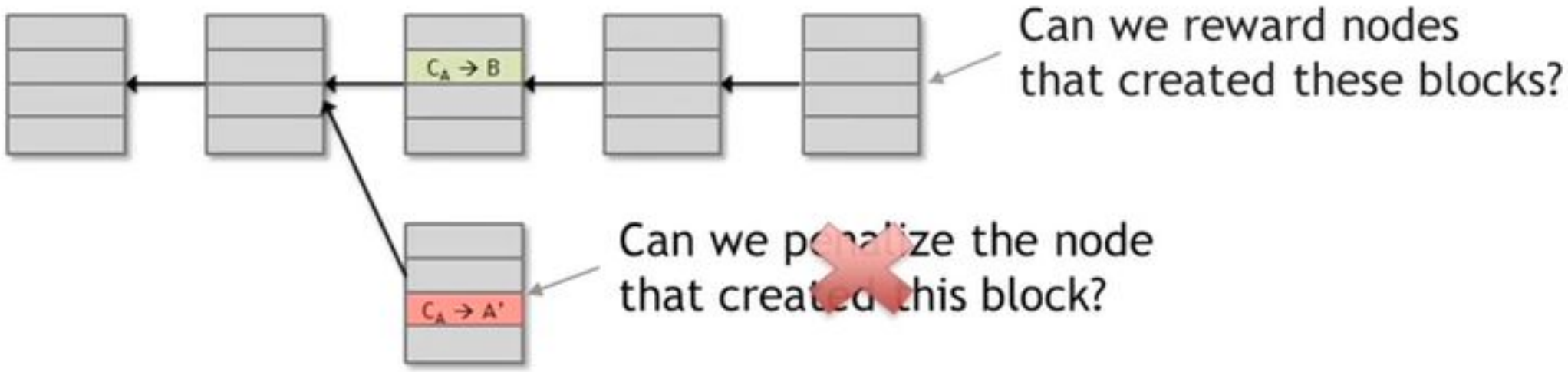
You're never 100% sure a transaction is in consensus branch. Guarantee is probabilistic

Assumption of honesty is problematic

Can we give nodes incentives for behaving honestly?

Assumption of honesty is problematic

Can we give nodes incentives for behaving honestly?



Incentive 1: block reward

Creator of block gets to

- include special coin-creation transaction in the block
- choose recipient address of this transaction

Value is fixed: currently 25 BTC, halves every 4 years

Incentive 2: transaction fees

Creator of transaction can choose to make output value less than input value

Remainder is a transaction fee and goes to block creator

Purely voluntary, like a tip

Remaining problems

1. How to pick a random node?
2. How to avoid a free-for-all due to rewards?
3. How to prevent Sybil attacks?

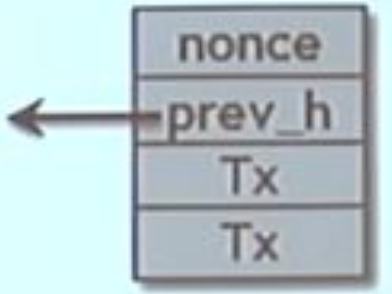
Proof of work

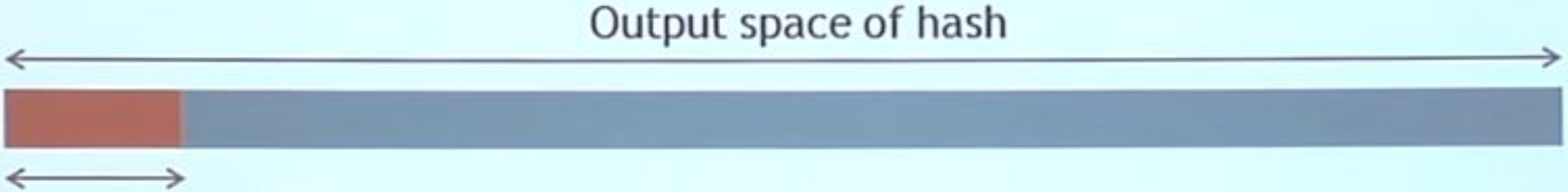
To approximate selecting a random node:
select nodes in proportion to a resource
that no one can monopolize (we hope)

- In proportion to computing power: proof-of-work
- In proportion to ownership: proof-of-stake

Hash puzzles

To create block, find nonce s.t.
H(nonce ‖ prev_hash ‖ tx ‖ … ‖ tx) is very small





If hash function is secure:
only way to succeed is to try enough nonces until you get lucky

What can a “51% attacker” do?

Steal coins from existing address?

Suppress some transactions?

- From the block chain
- From the P2P network

Change the block reward?

Destroy confidence in Bitcoin?

Bitcoin Stories: Bitcoin Cash (Fork)

- Updates to software
  - **Core developers trusted by the community, have great power!**
  - Fork on Aug. 1, 2017
- Block Size
  - 1MB vs. 32MB
  - ~7 TPS vs. ~100 TPS
- Impact?
  - Higher Transaction Speed?
  - Now, which is the “real” Bitcoin?
  - Transactions before Aug. 1, 2017?
  - Transaction after Aug. 1, 2017?
  - To fork or not to fork: balancing trust and evolution?

Beyond Bitcoin: Ethereum

Bitcoin

- 2009
- Proof-of-Work
- Coins (Bitcoin)
- Currency / Store of Value
- ~ Distributed Transaction Ledger
- Deflationary (Fixed Supply / Halved Reward)
- ASIC Mining
- 10 minutes (block time)
- ~7 TPS
- Miners collect Transaction Fees

Ethereum

- 2015
- Proof-of-Stake (originally PoW)
- Tokens (Ether)
- Smart Contracts / Currency
- ~ Distributed Computer
- Inflationary (Unlimited Supply / Fixed Reward)
- GPU Mining
- 12 seconds (block time)
- ~15 TPS
- Miners collect Gas Fees

Ethereum Stories: The DAO Hack (1/2)

- Updates to software
  - **Core developers trusted by the community, have great power!**
  - Fork on July 20th, 2016

- A typical **DAO** (Decentralized Autonomous Organization)
  - Smart contract developed
  - Initial Coin Offering (**ICO**)
  - After funding, DAO operates
  - Members vote on how DAO spends money

Ethereum Stories: The DAO Hack (2/2)

- Smart Contracts: **“Code is law”**
  - What about *bugs*? :)
  - Fork: Ethereum Classic vs Ethereum

- What happened?
  - The DAO launched on 30th April 2016 with a 28-day funding window.
  - The DAO was the largest crowdfunding in history: $150M from more than 11,000 members.
  - The DAO contained roughly 15% of all ether at the time
  - Saturday, 18th June, an attacker “steals” more than 3.6M ether **(> $70M)**
  - The price of ether dropped from over $20 to under $13.
  - Ethereum forked: 2 opposing “sides”

- Impact?
  - **“Code is law”**?
  - Eth. (April 12, 2023): **~$1,900**
  - Eth. Classic (April 12, 2023): **~$22**