Blockchains & Cryptocurrencies

Consensus & Towards Bitcoin

Instructor: Matthew Green
Johns Hopkins University - Fall 2020

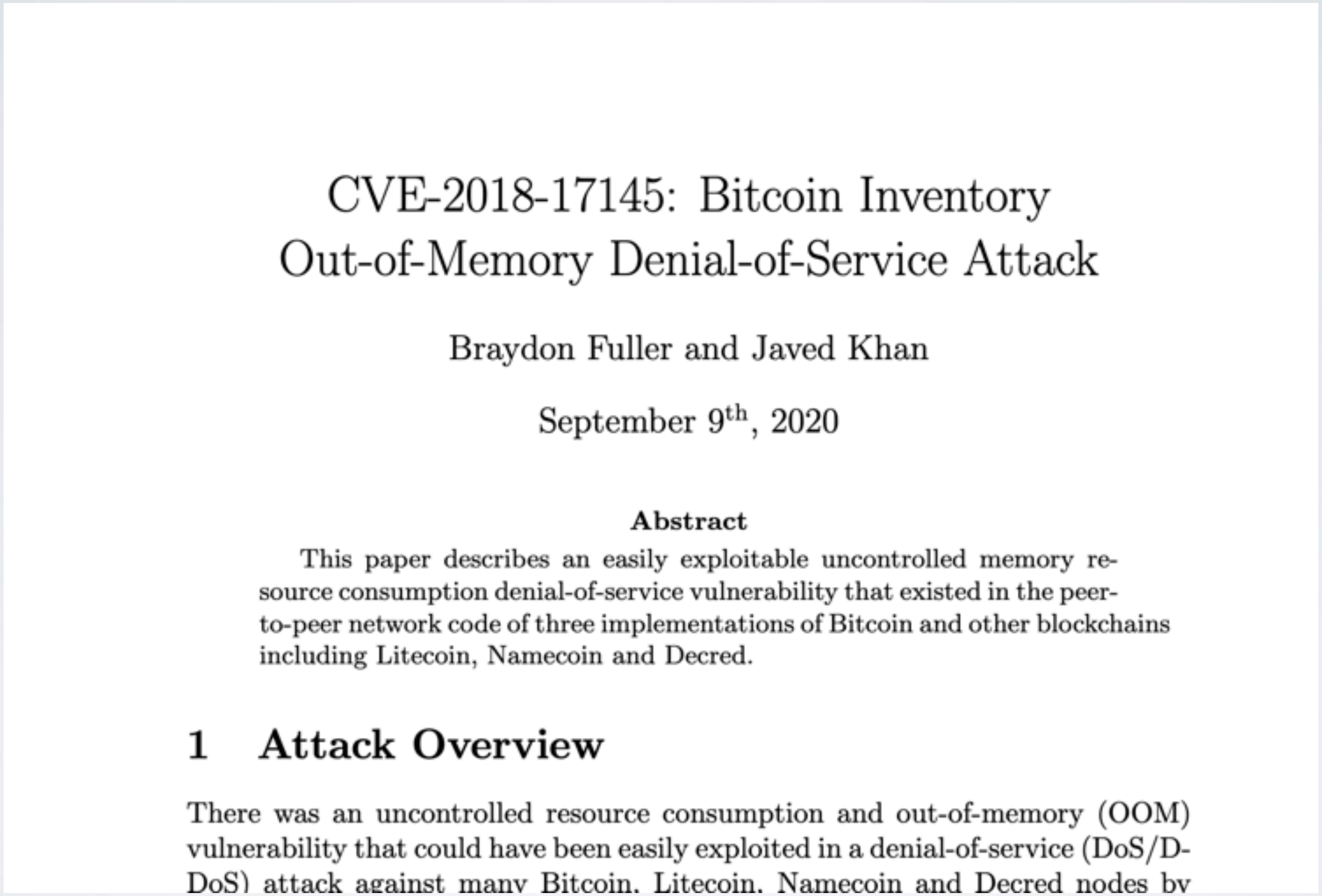
Many slides based on NBFMG

Housekeeping

- Readings: due <u>today</u> Bitcoin and Hashcash
- For next lecture: Lamport, P2P blog
- A1 is still due 9/24 11:59pm Baltimore time
- **Project groups and proposal is due 10/8 end of day**

News?

News?



CVE-2018-17145: Bitcoin Inventory Out-of-Memory Denial-of-Service Attack

Braydon Fuller and Javed Khan

September 9th, 2020

Abstract

This paper describes an easily exploitable uncontrolled memory resource consumption denial-of-service vulnerability that existed in the peer-to-peer network code of three implementations of Bitcoin and other blockchains including Litecoin, Namecoin and Decred.

1 Attack Overview

There was an uncontrolled resource consumption and out-of-memory (OOM) vulnerability that could have been easily exploited in a denial-of-service (DoS/D-DoS) attack against many Bitcoin, Litecoin, Namecoin and Decred nodes by

Today

- We're going to talk about "consensus"
- What the heck is consensus, how do you accomplish it, what's the point?
- This is all in preparation for next time, when we'll actually talk about Bitcoin

Review: digital signatures

Digital signatures

Security parameter

- (sk, pk) ← keygen(1^k)

  sk: secret signing key

  pk: public verification key

- sig ← sign(sk, message)

- isValid ← verify(pk, message, sig)

$\}$ randomized algorithm

$\}$ Typically randomized

Requirements for signatures

- Correctness: “valid signatures verify”
    - verify(pk, message, sign(sk, message)) == true
- Unforgeability under chosen-message attacks (UF-CMA): “can’t forge signatures”
    - adversary who knows pk, and gets to see signatures on messages of his choice, can’t produce a verifiable signature on another message

Review: cash problems

- **Double spending**
  - To capture double spending you need an online (networked) party that must be trusted
- **Authentication / Authentication**
  - How do I prove that I am the owner of currency & thus authorized to transact with it?
- **Origin/Issuance**
  - How is new currency created?

Partial approach

- Let’s not dispense with our centralized approach (just yet)
  - We will, however, reduce the number of our assumptions
  - Our new assumption is that there is a **centralized** party that can maintain a ledger
  - This centralized party also can create (“mint”) new currency and assign it to be owned by users
  - But other than that, they have no power
  - Key requirement: <u>digital signature</u>

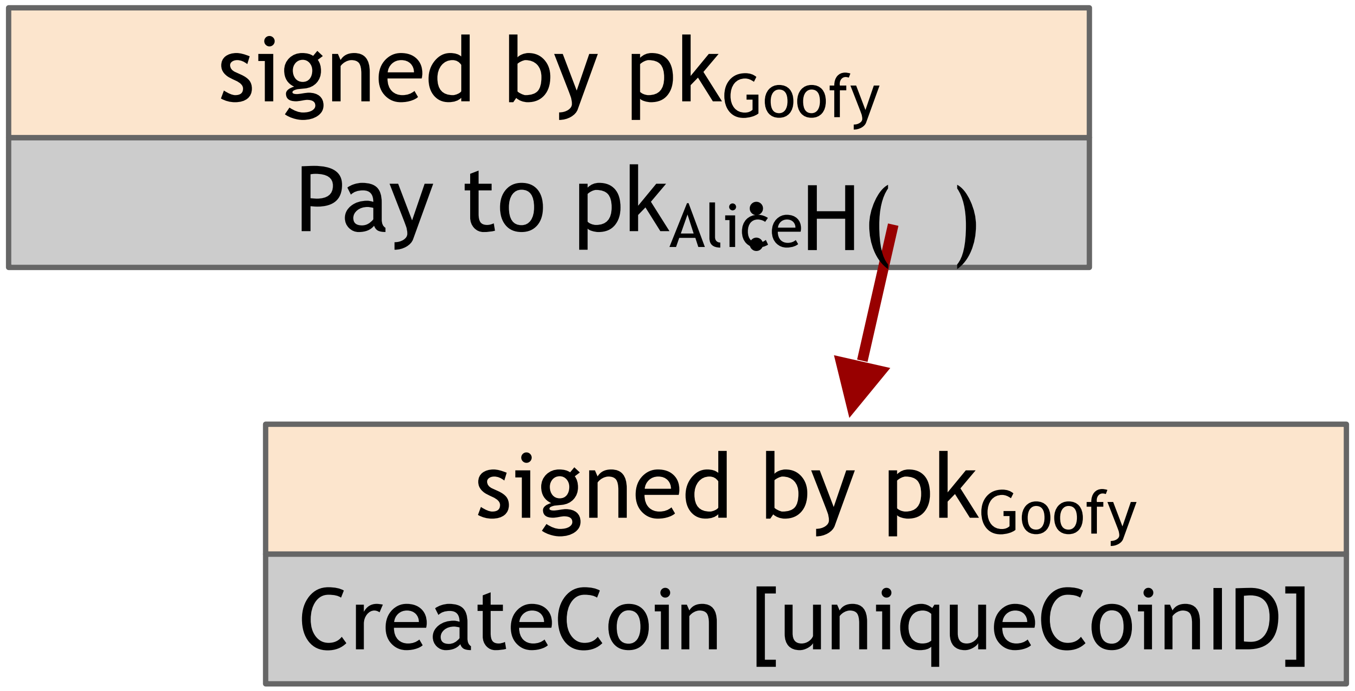
GoofyCoin

Goofy can create new coins

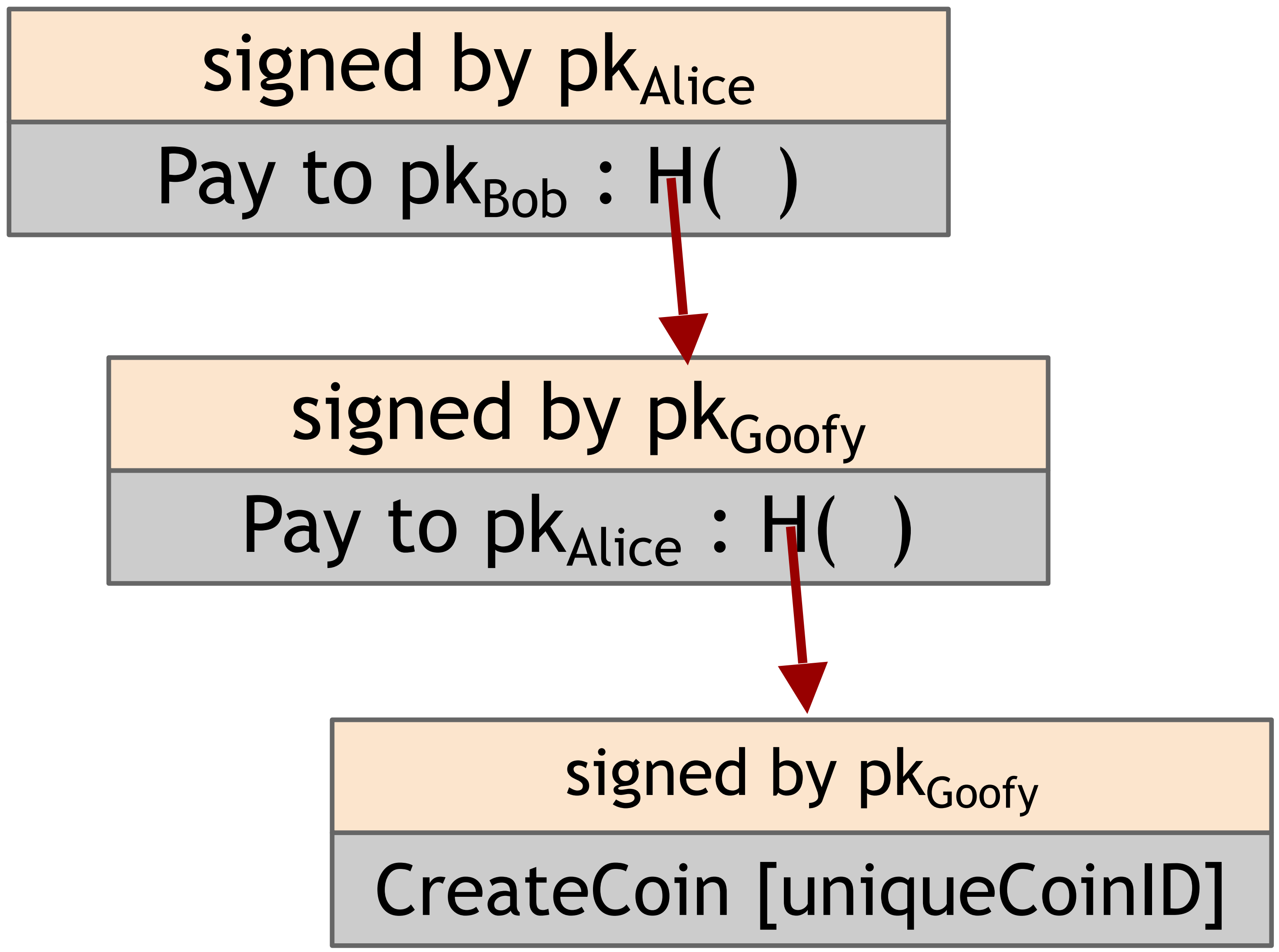
signed by pk_{Goofy}

CreateCoin [uniqueCoinID]

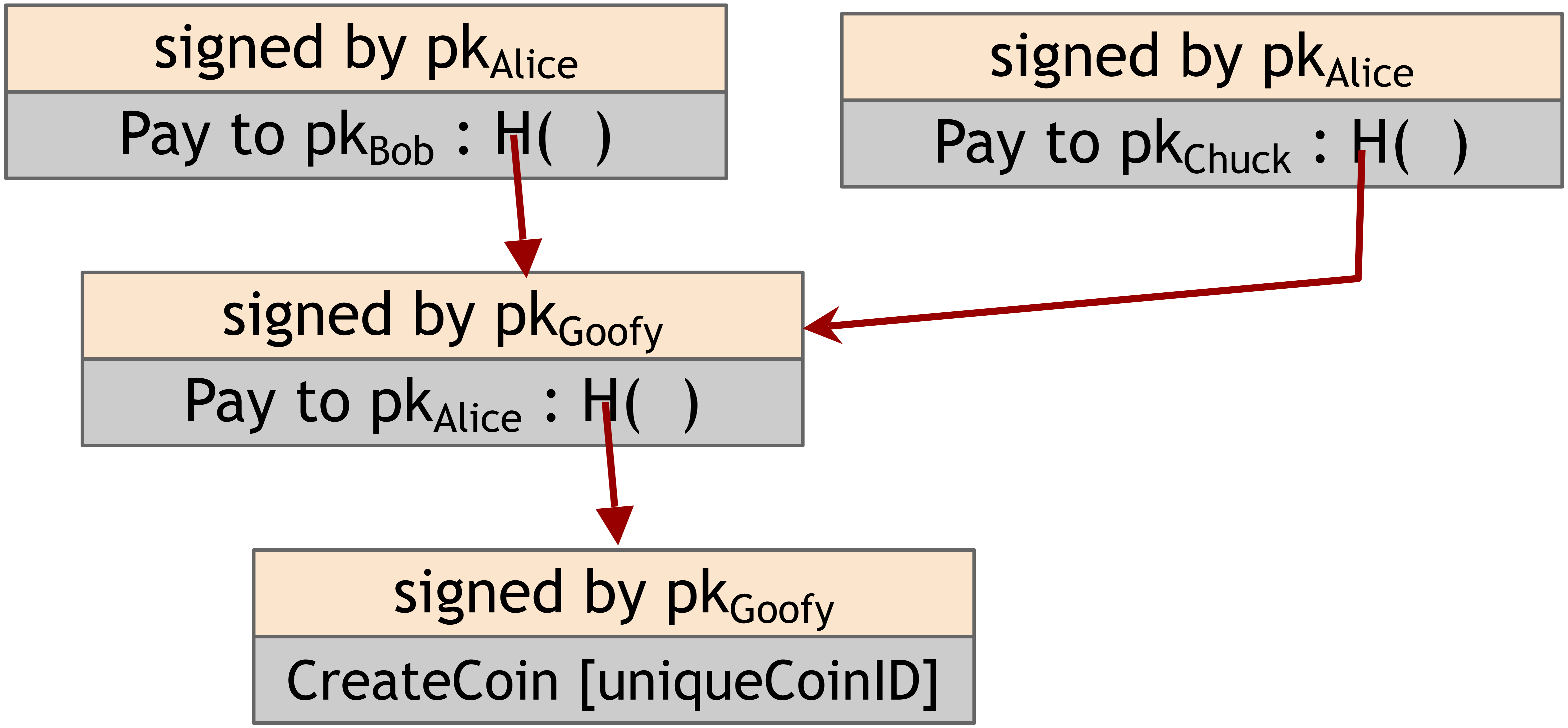
A coin's owner can spend it.



The recipient can pass on the coin again.



double-spending attack



double-spending attack

This is the main design challenge in digital currency

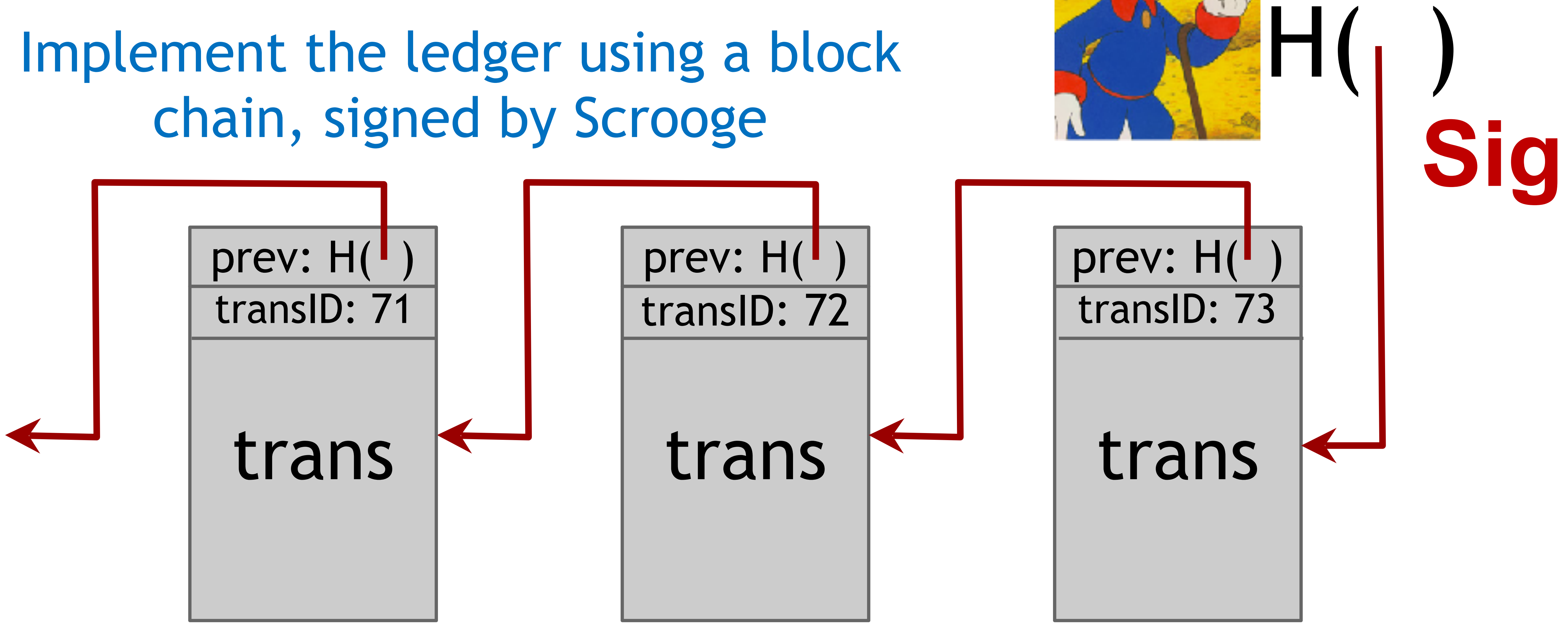
How do we solve this?

- Simplest answer: send all transactions to an atomic, append-only centralized ledger
- Have the ledger provide a definite ordering for transactions
  - If two transactions conflict, simply disallow the later one
- No TX is valid unless the ledger has "approved" and ordered it

ScroogeCoin

Scrooge publishes a history of all transactions in an "append-only" ledger

Implement the ledger using a block chain, signed by Scrooge



optimization: put multiple transactions in the same block

CreateCoins transaction creates new coins

transID: 73	type:CreateCoins	
coins created		
num	*value*	*recipient*
0	3.2	0x...
1	1.4	0x...
2	7.1	0x...

coinID 73(0)
coinID 73(1)
coinID 73(2)

signature

Valid, because I said so.

CreateCoins transaction creates new coins

transID: 73	type:CreateCoins	
coins created		
num	*value*	*recipient*
0	3.2	0x...
1	1.4	0x...
2	7.1	0x...

coinID 73(0)

coinID 73(1)

coinID 73(2)

signature

These are public keys!

Valid, because I said so.

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

One signature for each consumed coin

Immutable coins

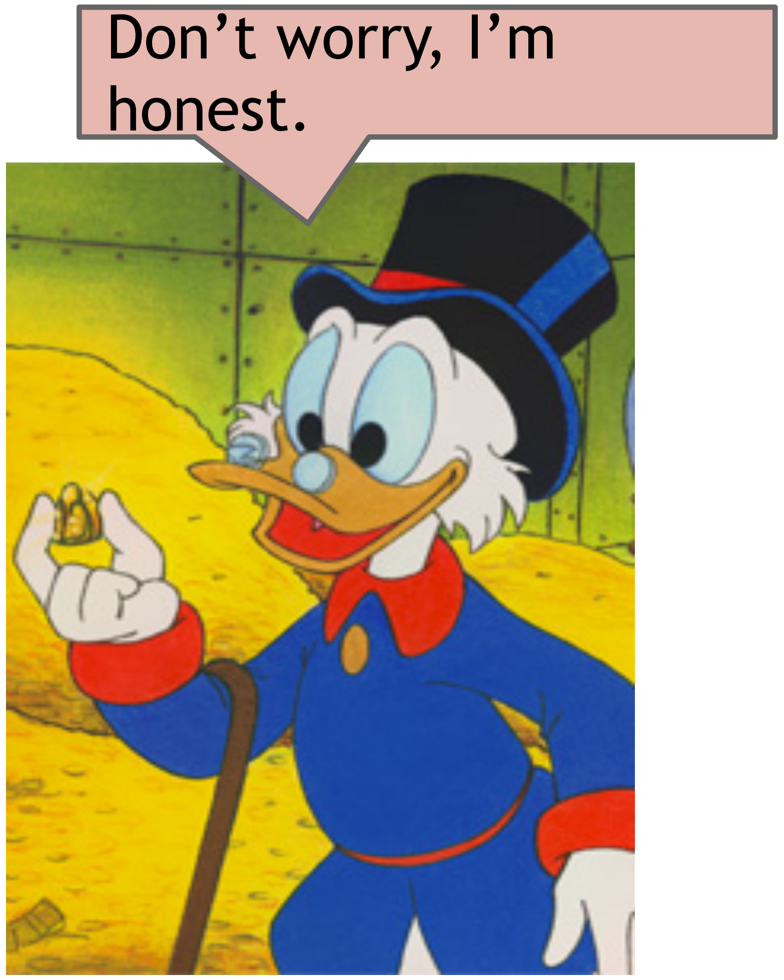
Coins can’t be transferred, subdivided, or combined.

But: you can get the same effect by using transactions
- to subdivide: create new transaction
  - consume your coin
    - pay out two new coins to yourself

Don’t worry, I’m honest.

Crucial question:

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



Crucial question:

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

Related question:

Why do we <u>need</u> to do this?

Centralization vs. Decentralization

- **Competing paradigms that underlie many technologies**
- Decentralized != Distributed
  (as in distributed system) but we'll often use them as synonyms

Centralization vs. Decentralization

- Examples:
  - email?
  - WWW?
  - DNS?
- What about software development?

Aspects of decentralization in Bitcoin

1. Who maintains the ledger?
2. Who has authority over which transactions are valid?
3. Who creates (and obtains) new bitcoins?
4. Who determines how the rules change?
5. How do these coins acquire monetary value?

Aspects of decentralization in Bitcoin

Peer-to-peer network:

open to anyone, low barrier to entry
high node churn (nodes can come and go)

Mining:

open to anyone, but inevitable concentration of power
often seen as undesirable

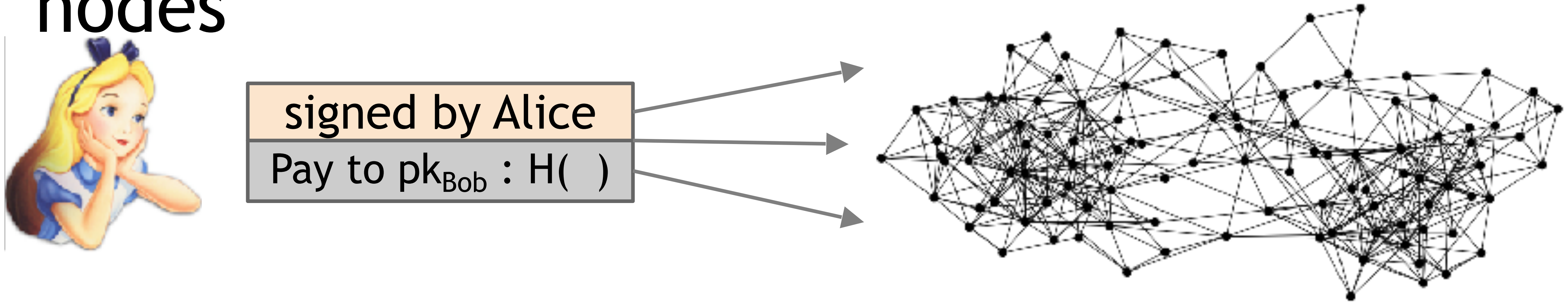
Updates to software:

core developers trusted by community, have great power

Distributed consensus

Bitcoin is a peer-to-peer system

When Alice wants to pay Bob:
she <u>broadcasts the transaction</u> to all Bitcoin nodes

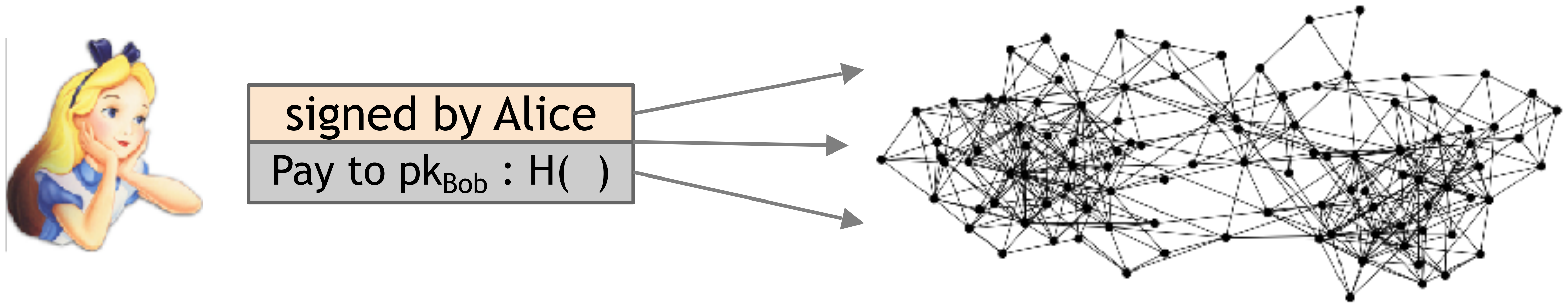


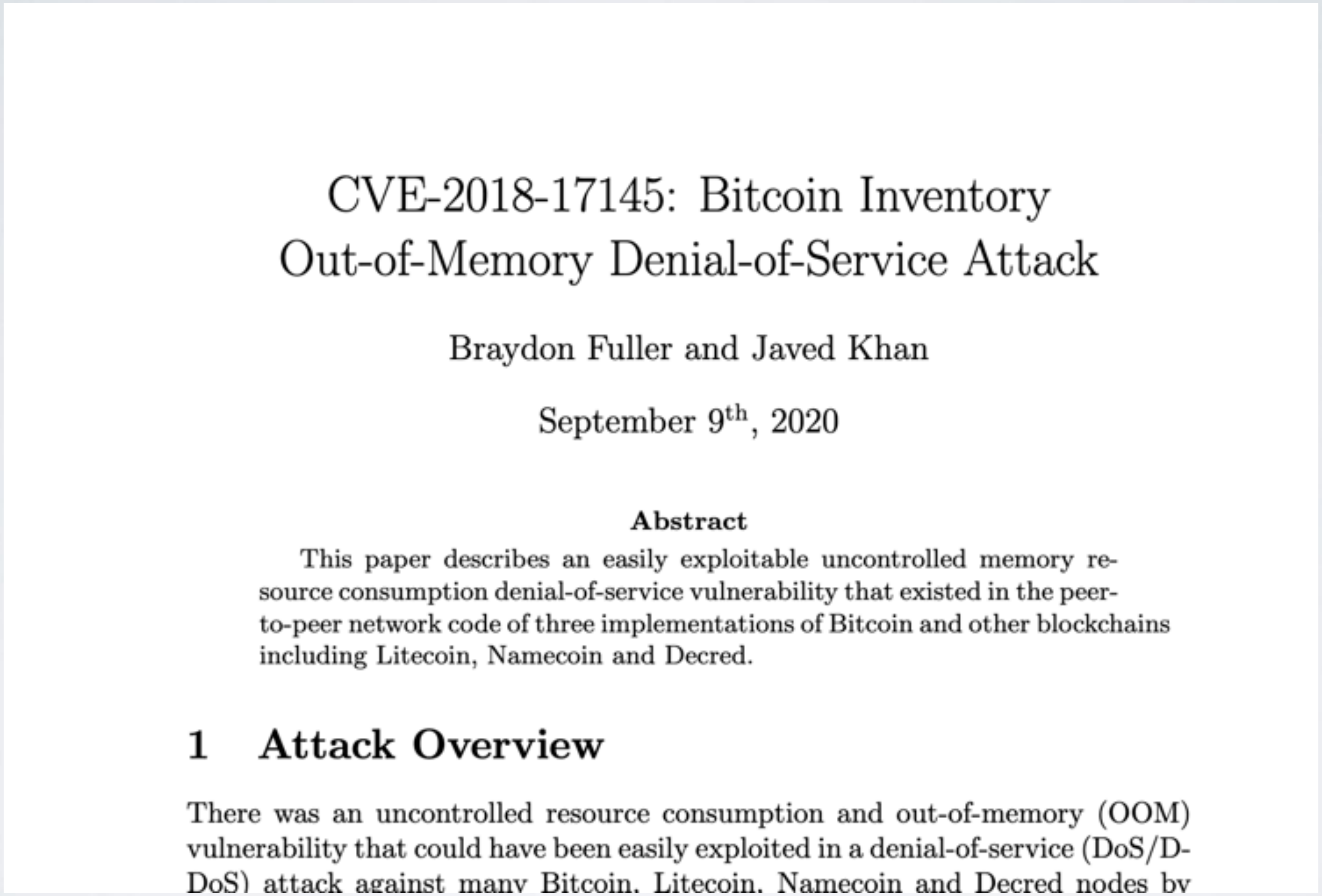
Note: Bob’s computer is not in the picture

Bitcoin is a peer-to-peer system

This network is a fill/flood style P2P network: all nodes perform basic validation, then relay to their peers

This introduces bootstrapping, spam and DoS problems, which are dealt with through "seeders" and "reputation" scores





CVE-2018-17145: Bitcoin Inventory Out-of-Memory Denial-of-Service Attack

Braydon Fuller and Javed Khan

September 9th, 2020

Abstract

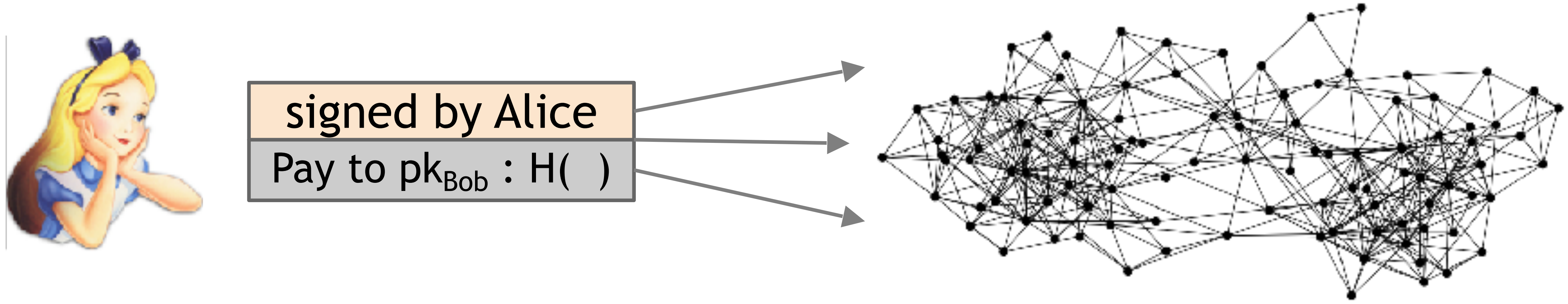
This paper describes an easily exploitable uncontrolled memory resource consumption denial-of-service vulnerability that existed in the peer-to-peer network code of three implementations of Bitcoin and other blockchains including Litecoin, Namecoin and Decred.

1 Attack Overview

There was an uncontrolled resource consumption and out-of-memory (OOM) vulnerability that could have been easily exploited in a denial-of-service (DoS/D-DoS) attack against many Bitcoin, Litecoin, Namecoin and Decred nodes by

Why aren’t we done here?

Why can’t we just trust this system to eliminate invalid blocks, and give everyone a robust view of the Tx history?



Bitcoin's key challenge

Key technical challenge of decentralized e-cash: <u>distributed consensus</u>

or: how do all of these nodes agree on an <u>ordered</u> history of transactions?

Defining distributed consensus

The protocol terminates and all honest nodes decide on the same **value**

This value must have been proposed by some honest node

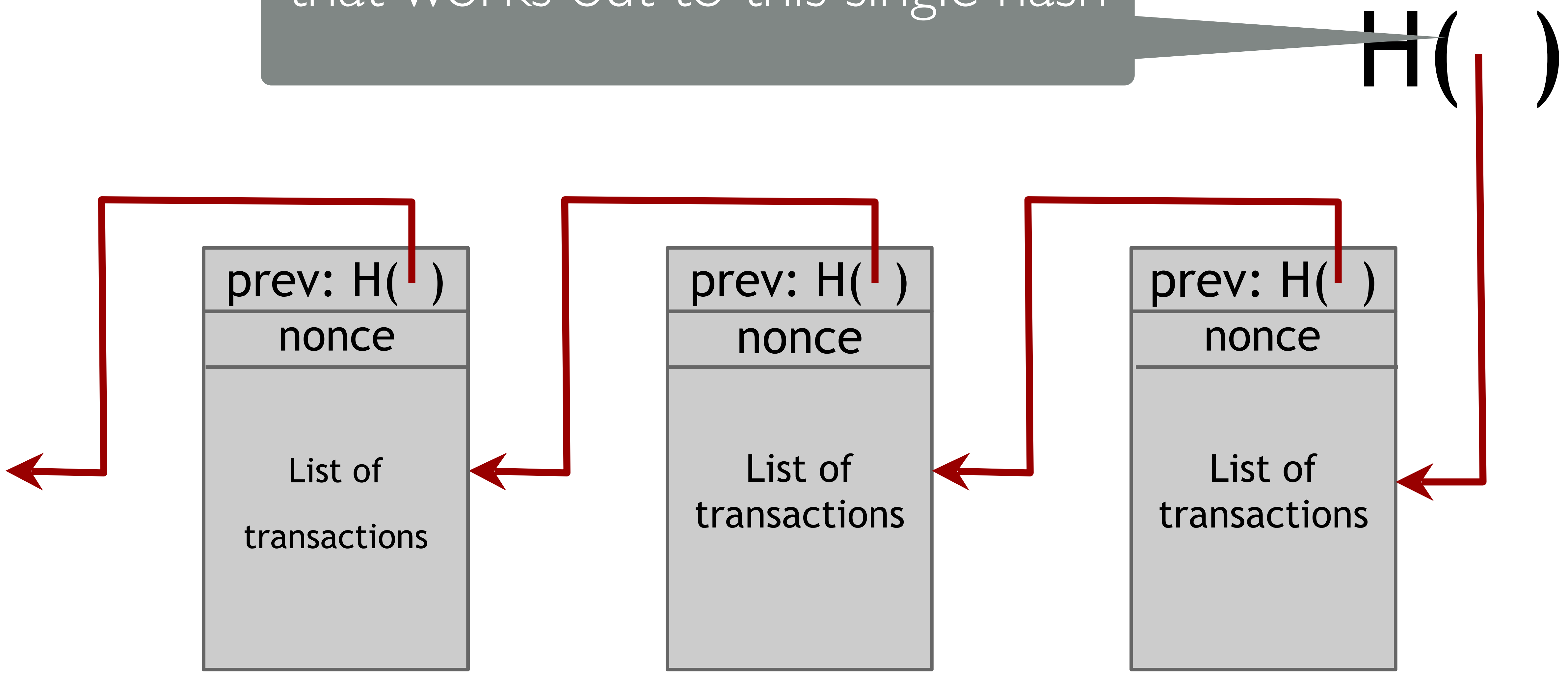
Defining distributed consensus

Q: What is this "value" in Bitcoin?

The protocol terminates and all honest nodes decide on the same **value**

This value must have been proposed by some honest node

A: In Bitcoin, the value we want to agree on is the current state of the ledger. If we use a blockchain, that works out to this single hash

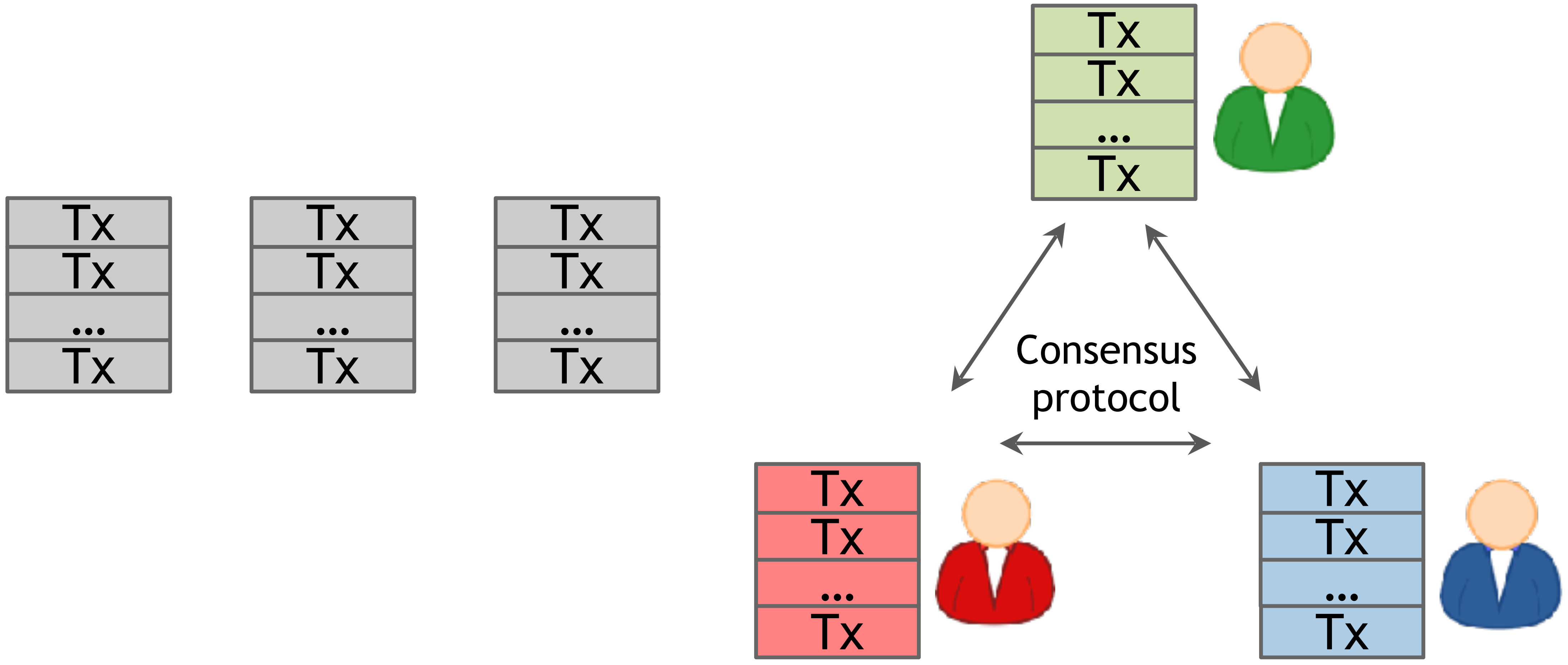


How consensus could work in Bitcoin

At any given time:

- All nodes have a sequence of blocks of transactions they've reached consensus on
- (Blocks are also distributed via p2p network)
- Each node has a set of outstanding transactions it's heard about

How consensus could work in Bitcoin



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

Why consensus is hard

Nodes may crash
Nodes may be malicious

Network is imperfect

- Not all pairs of nodes connected
- Faults in network ("partitioning")
- Latency

No notion of global time

Defining distributed consensus

The protocol terminates and all honest nodes decide on the same value (history)

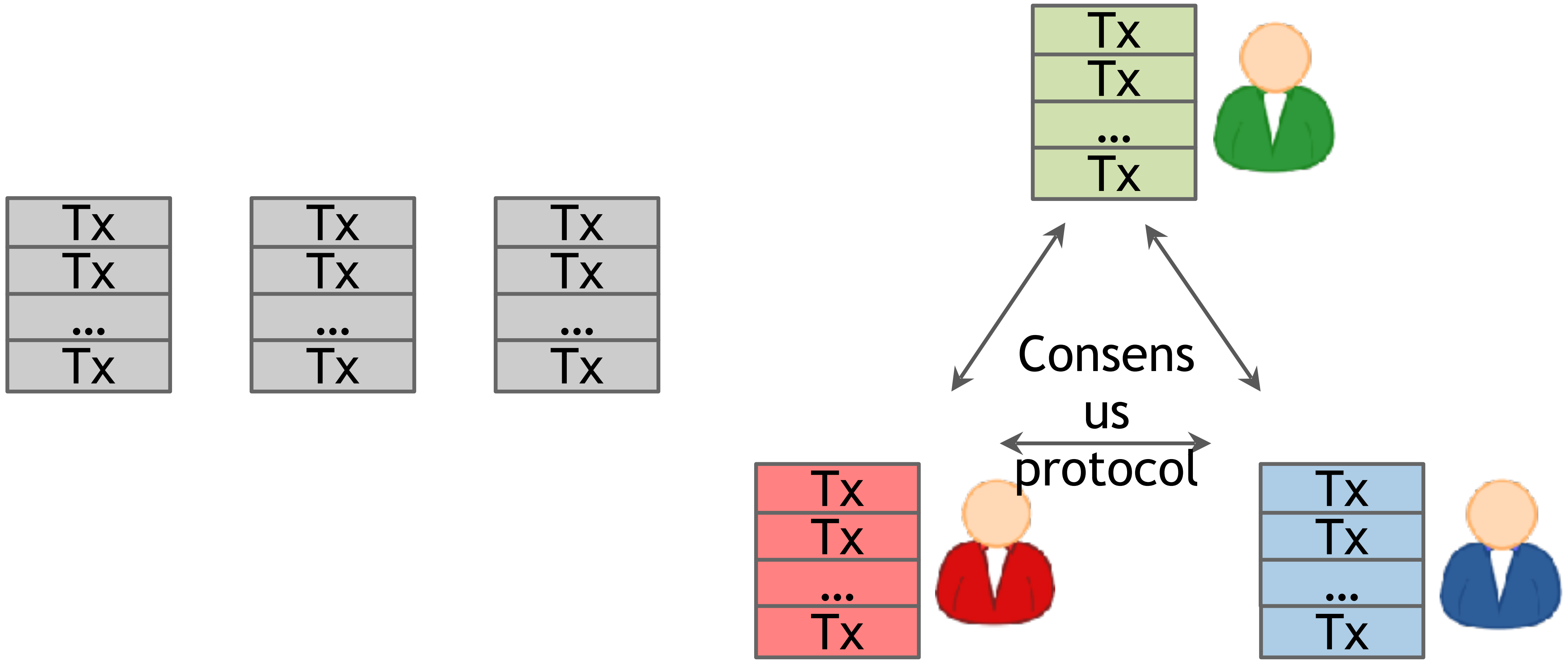
This value must have been proposed by some honest node

How consensus could work in Bitcoin

At any given time:

- All nodes have a sequence of blocks of transactions they’ve reached consensus on
- Each node has a set of outstanding transactions it’s heard about

How consensus could work in Bitcoin



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

Why consensus is hard

Nodes may crash
Nodes may be malicious

Network is imperfect

- Not all pairs of nodes connected
- Faults in network
- Latency

↳ No notion of global time

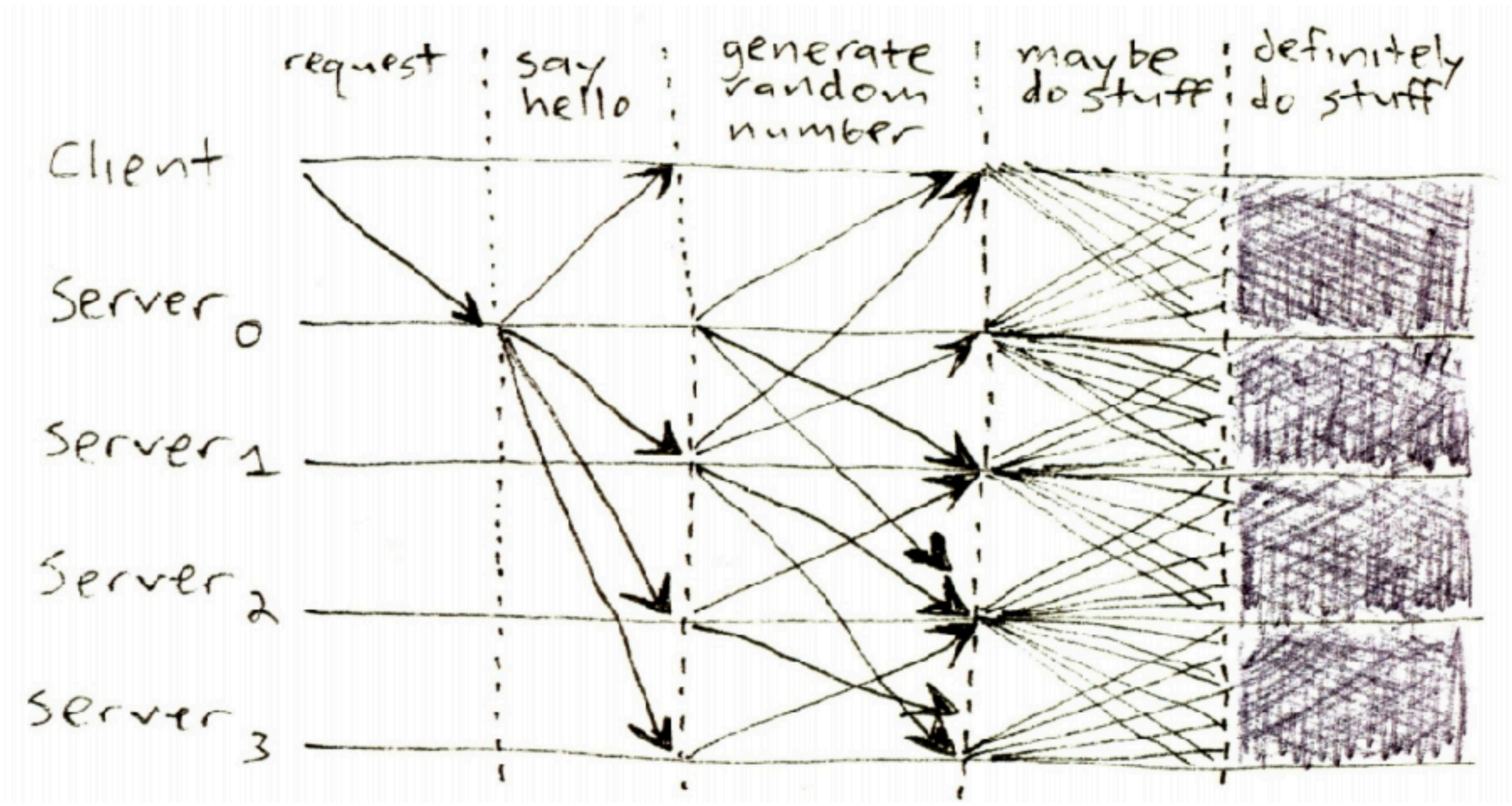
Many impossibility results

- Impossible without 2/3 honest majority [Pease, Shostak, Lamport’80]
- Impossible with a single faulty node, in the fully asynchronous setting, with deterministic nodes [Fischer-Lynch-Paterson’85]

Why do these results matter?

- Because without node identities,
  an attacker could easily crash these networks by impersonating many nodes
  (“Sybil attack”)

- Because synchronicity is hard


request
say hello
generate random number
maybe do stuff
definitely do stuff
Client
Server 0
Server 1
Server 2
Server 3

Some positive results

Example: Paxos [Lamport]

Never produces inconsistent result, but can (rarely) get stuck

Understanding impossibility results

These results say more about the model than about the problem

The models were developed to study systems like distributed databases

Bitcoin consensus: theory & practice

- Bitcoin consensus: initially, seemed to work better in practice than in theory

- Theory has been steadily catching up to explain why Bitcoin consensus works [e.g., Garay-Kiayias-Leonardos’15,Pass-Shelat-Shi’17,Garay-Kiayias-Leonardos’17,…]

- Theory is important, can help predict unforeseen attacks

Some things Bitcoin does differently

Introduces incentives

- Possible only because it’s a currency!

Embraces randomness

- Does away with the notion of a specific end-point
- Consensus happens over long time scales — about 1 hour

Consensus without identity: the blockchain

Why identity?

Pragmatic: some protocols need node IDs

Security: assume less than 50% malicious

Why don’t Bitcoin nodes have identities?

Identity is hard in a P2P system — Sybil attack

Pseudonymity is a goal of Bitcoin

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

<u>Every block contains hash of the block it extends</u>

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 <u>random</u> node gets to broadcast its block
4. Other nodes accept the block only if all transactions in it are valid (unspent, valid signatures) and it builds on a chain they already accept
5. Nodes express their acceptance of the block by including its hash in the next block they create

So how do we pick a random node?

Resources & Consensus

- One computer can easily pretend to be many "nodes", so simple random voting != good
- But an observation: resources (e.g., hardware, storage, CPU, GPU, etc.) are much harder to fake
- Idea: make your probability of winning the vote proportional to your overall resources