Lecture 3

Introduction to Cryptocurrencies

Public Keys as Identities

public key := an identity

- if you see *sig* such that *verify(pk, msg, sig)=true*, think of it as:

  pk says, “*[msg]*”

- to “speak for” *pk*, you must know matching secret key *sk*

How to make a new identity

- create a new, random key pair *(sk, pk)*
  - *pk* is the public “name”
  - *sk* lets you “speak for” the identity

- you control the identity, because only you know *sk*
- if *pk* “looks random”, nobody needs to know who you are

Decentralized identity management

- anybody can make a new identity at any time
- make as many as you want!
- no central point of coordination

These identities are called “addresses” in Bitcoin

Privacy

- Addresses not directly connected to real-world identity
- But observer can link together an address's activity over time, make inferences
- Later: privacy issues in Bitcoin, and cryptocurrencies with provable anonymity

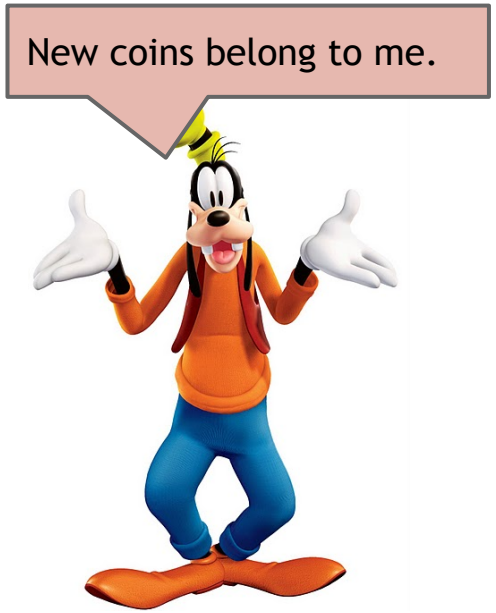
Simple Cryptocurrencies

GoofyCoin

Goofy can create new coins

signed by pk_{Goofy}

CreateCoin [uniqueCoinID]



A coin's owner can spend it.

signed by pk_{Goofy}

Pay to pk_{Alice} : H( )

signed by pk_{Goofy}

CreateCoin [uniqueCoinID]

The recipient can pass on the coin again.

signed by pk_{Alice}

Pay to pk_{Bob} : H( )

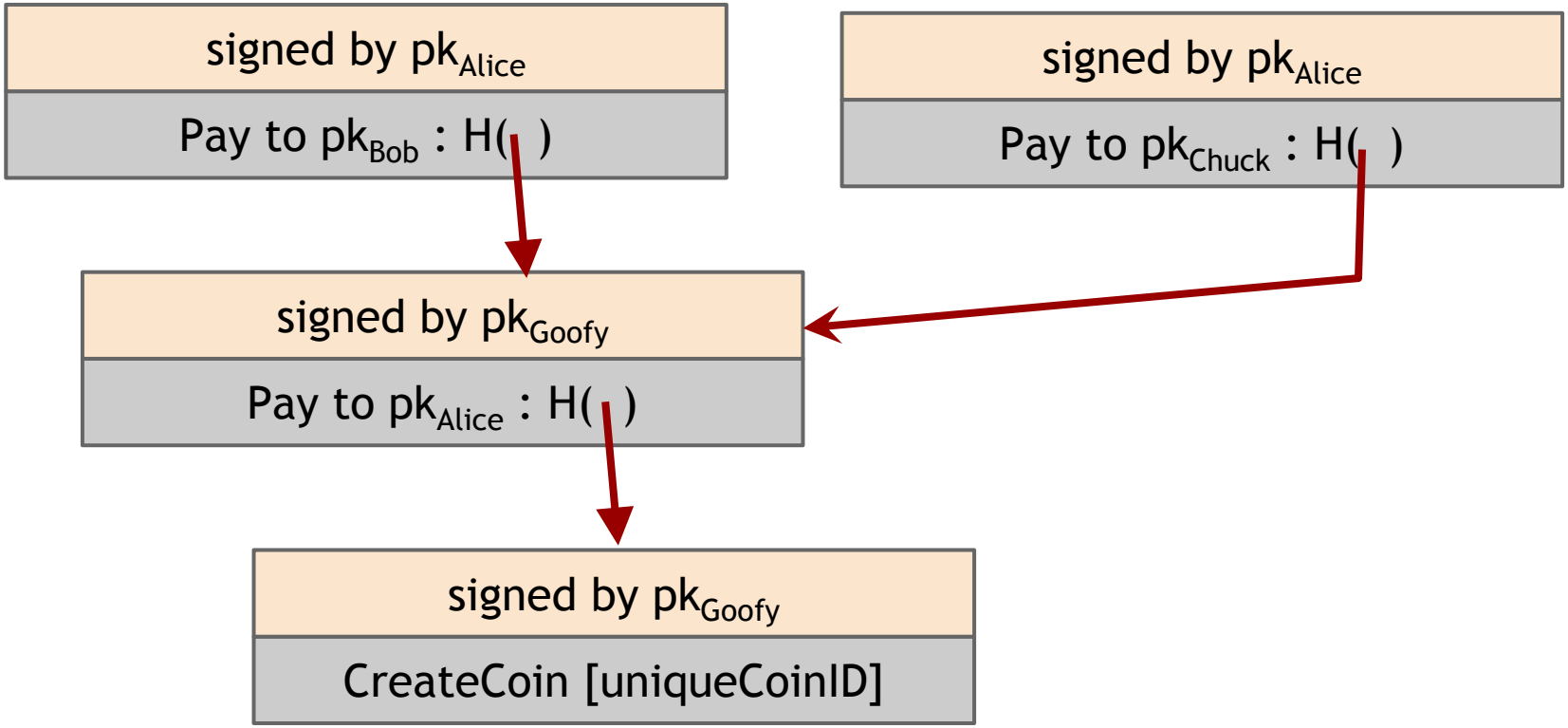
signed by pk_{Goofy}

Pay to pk_{Alice} : H( )

signed by pk_{Goofy}

CreateCoin [uniqueCoinID]

double-spending attack



double-spending attack

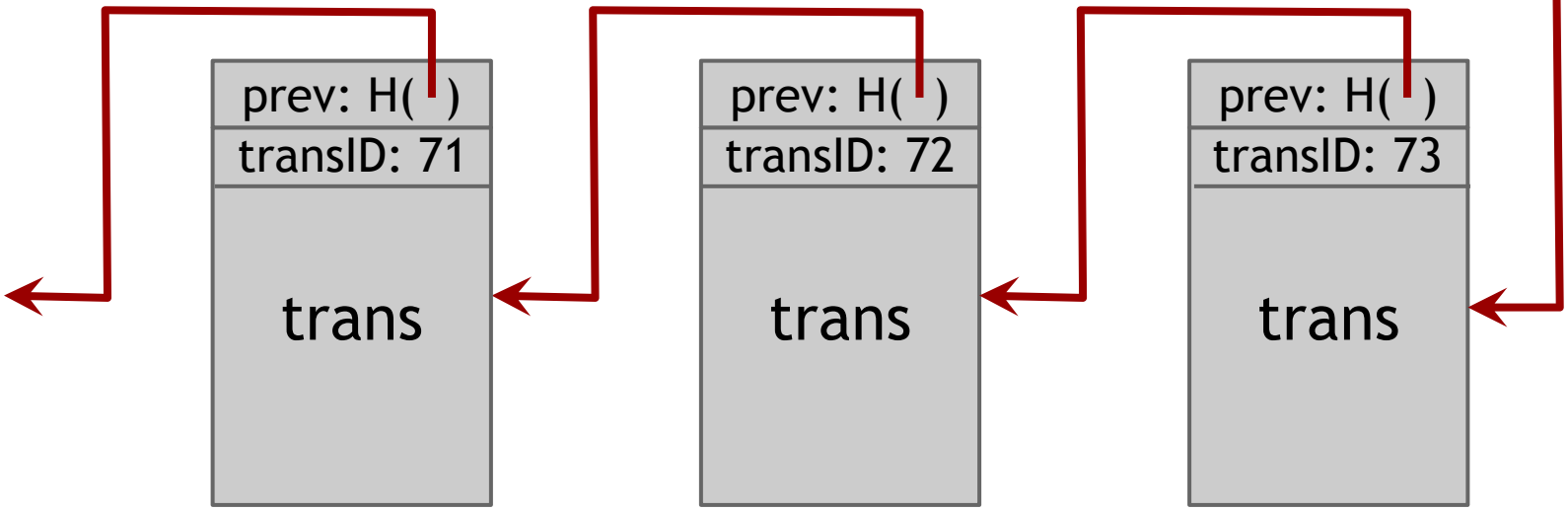
the main design challenge in digital currency

ScroogeCoin

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

H( ), Sig

Implement the ledger using a block chain, signed by Scrooge



optimization: put multiple transactions in the same block

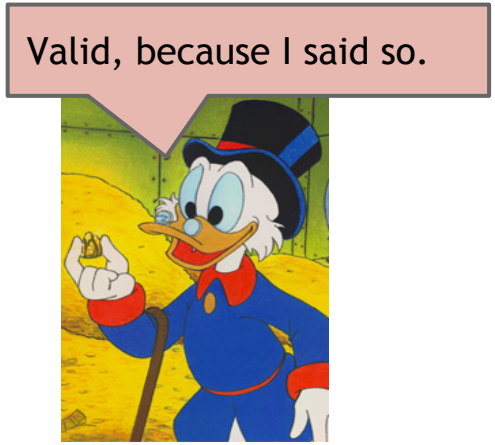
CreateCoins transaction creates new coins

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

coinID 73(0)

coinID 73(1)

coinID 73(2)



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

Immutable coins

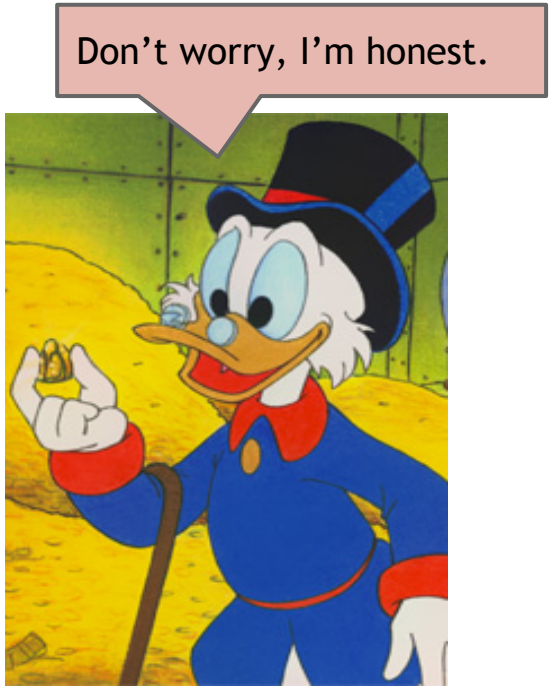
Coins can’t be transferred, subdivided, or combined.

But: you can get the same effect by using transactions

to subdivide: create new trans

consume your coin

pay out two new coins to yourself



Crucial question:

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

How Bitcoin achieves Decentralization

Centralization vs. decentralization

Competing paradigms that underlie many digital technologies

Decentralization is not all-or-nothing

E-mail:

decentralized protocol, but dominated by centralized webmail services

Aspects of decentralization in Bitcoin

1. Who maintains the ledger?
2. Who has authority over which transactions are valid?
3. Who creates new bitcoins?
4. Who determines how the rules of the system change?
5. How do bitcoins acquire exchange value?

Beyond the protocol:
exchanges, wallet software, service providers...

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

Distributed consensus

Bitcoin's key challenge

Key technical challenge of decentralized e-cash: distributed consensus

or: how to decentralize ScroogeCoin

Why consensus protocols?

Traditional motivation: reliability in distributed systems

Distributed key-value store enables various applications:
DNS, public key directory, stock trades …

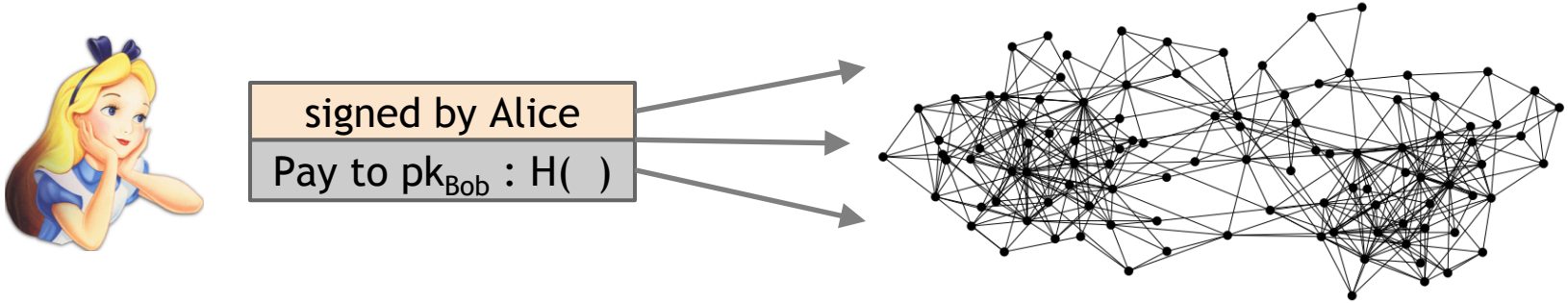
Defining distributed consensus

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

This value must have been proposed by some honest node

Bitcoin is a peer-to-peer system

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



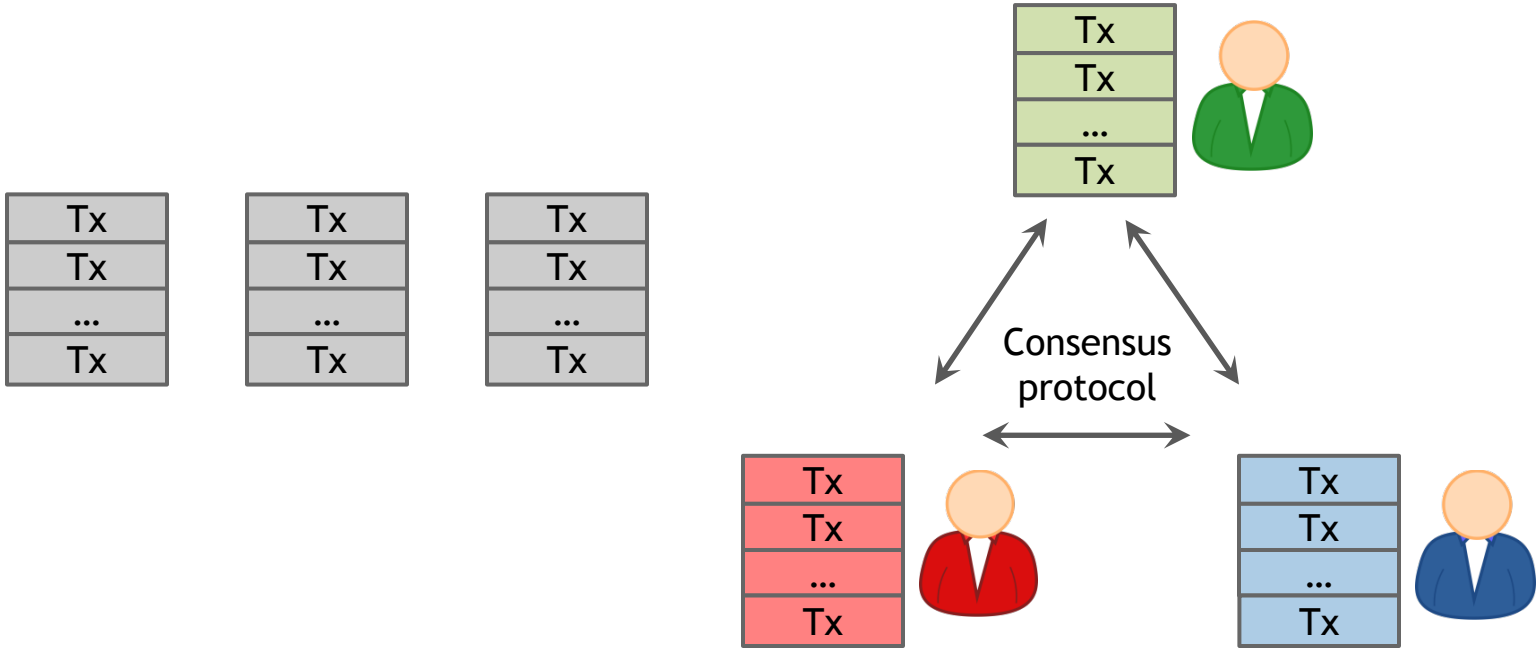
Note: Bob's computer is not in the picture

How consensus could work in Bitcoin

At any given time:

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

How consensus could work in Bitcoin



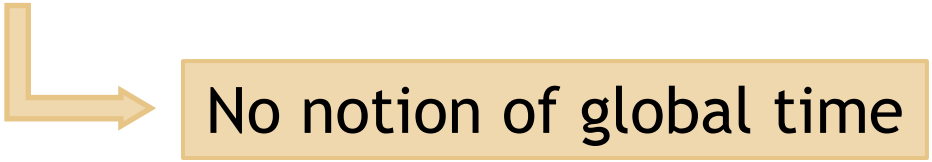
OK to select any valid block, even if proposed by only one node

Why consensus is hard

Nodes may crash
Nodes may be malicious

Network is imperfect

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



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]

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

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