Cryptography Application Goals

RECAP

Confidentiality: only intended recipient can read a message

Integrity: received message is unchanged from sender

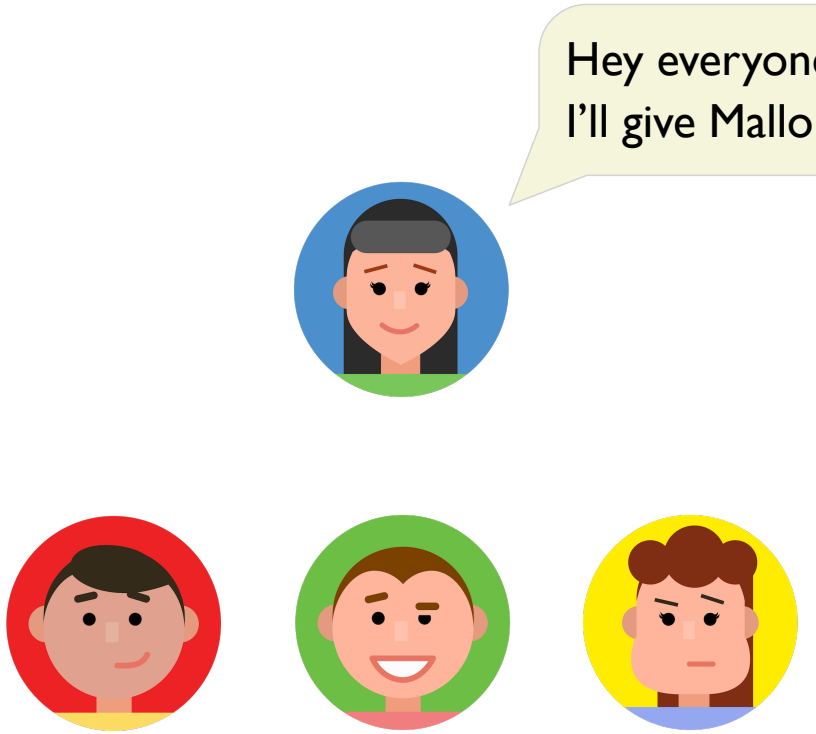
Authenticity: identity of each communicating party can be confirmed

and sometimes

Non-repudiation: parties cannot deny previous commitments

We still haven't covered this one

Commitment via Public Declarations



Commitment via Public Declarations

It's clear who is making a promise

Commitment via Public Declarations

We know how to handle **authentication**

It's clear who is making a promise

Everyone must pay attention and remember correctly

Can write things down, but need a way to make sure that the record is accurate and never rewritten

This is a question of **integrity,** and we know how to handle that, too

Commitment via Public Declarations

We know how to handle **authentication**

It's clear who is making a promise

Everyone must pay attention and remember correctly

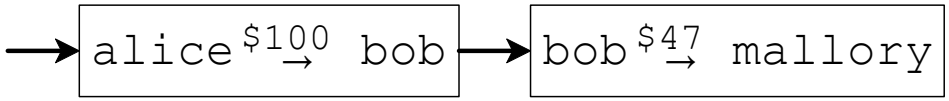
Can write things down, but need a way to make sure that the record is accurate and never rewritten

This is a question of **integrity,** and we know how to handle that, too

Needs to scale to lots of people with many promises

A Chain of Blocks

Define a **ledger** as a sequence of blocks

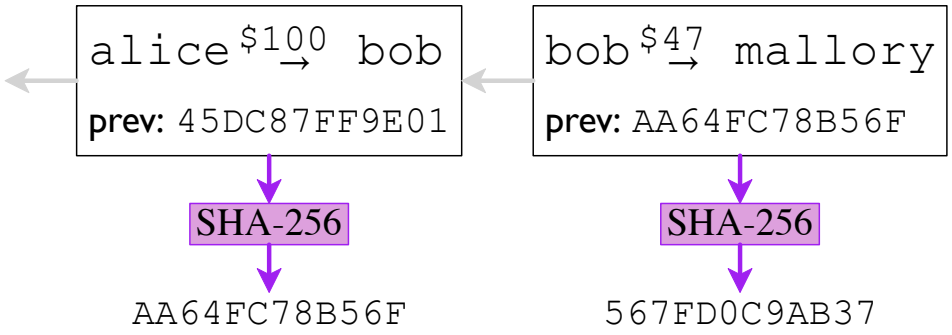


A Chain of Blocks

Define a **ledger** as a sequence of blocks

Define each **block** to include data plus the hash of the next-older block in the sequence

⇒ a **chain** via hashes

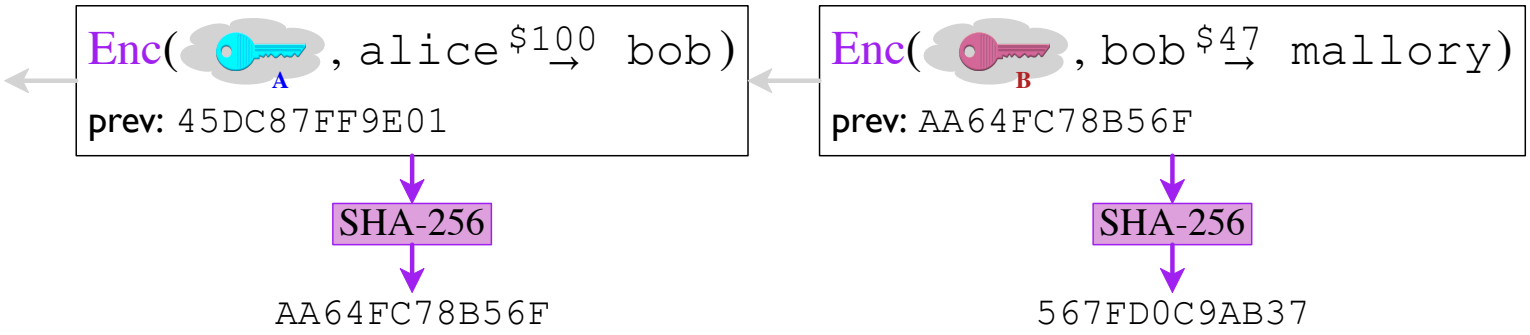


A Chain of Blocks

Define a **ledger** as a sequence of blocks

Define each **block** to include data plus the hash of the next-older block in the sequence

⇒ a **chain** via hashes

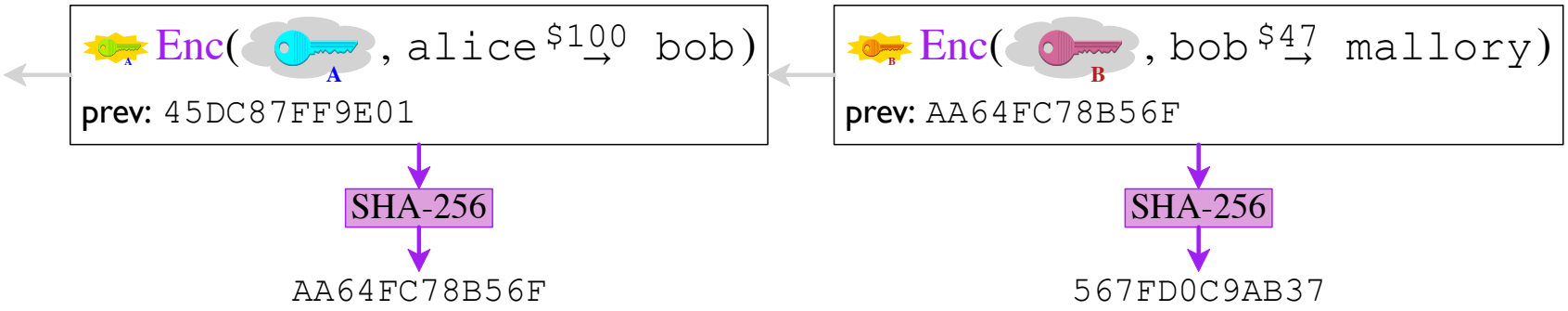


A Chain of Blocks

Define a **ledger** as a sequence of blocks

Define each **block** to include data plus the hash of the next-older block in the sequence

⇒ a **chain** via hashes

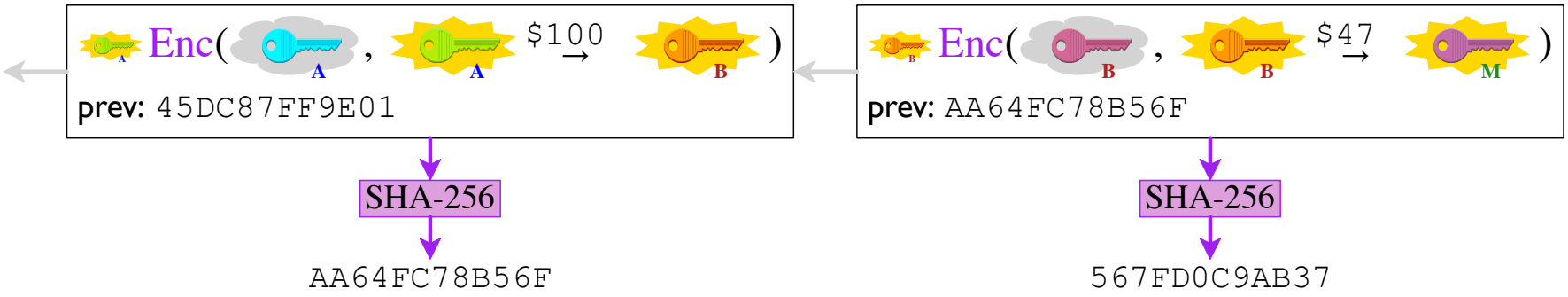


A Chain of Blocks

Define a **ledger** as a sequence of blocks

Define each **block** to include data plus the hash of the next-older block in the sequence

⇒ a **chain** via hashes



As long as two entities agree on the current block (or even just its hash), they agree on the whole sequence of blocks

Trusting a Blockkeeper

Can we avoid having a trusted keeper of blocks that everyone will agree on?

hash code

evidence of

Idea 1: blockkeeper publishes blocks regularly to some third party

The company Surety has been doing this since 1995, using the *New York Times* classifieds as the third party

Idea 2: every participant keeps all blocks and agree — without a designated leader — through a **consensus** algorithm

Avoids a gatekeeper for adding the ledger

The term **blockchain** may imply Idea 2

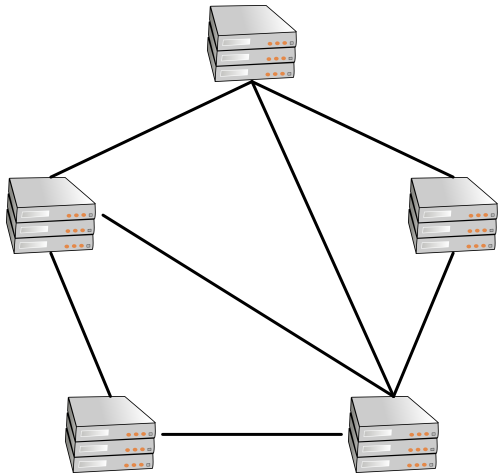
Blockchain Uses

There are many potential uses of blockchains

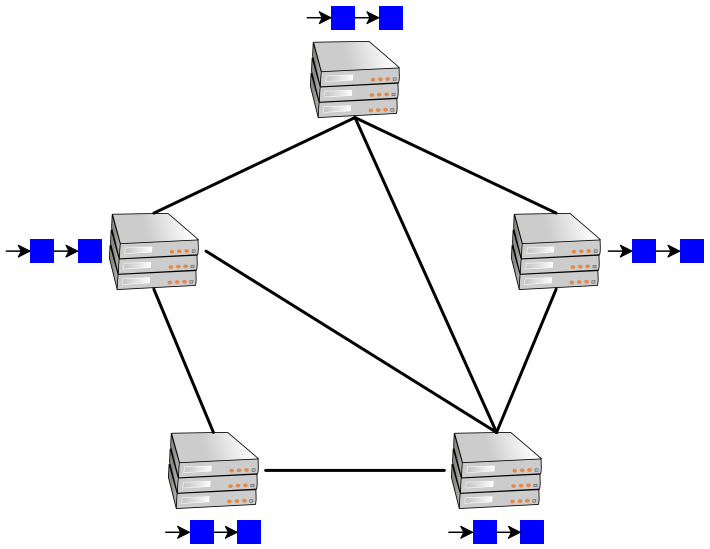
Cryptocurrency is one of the main uses

- Bitcoin
- Etherium
- many others

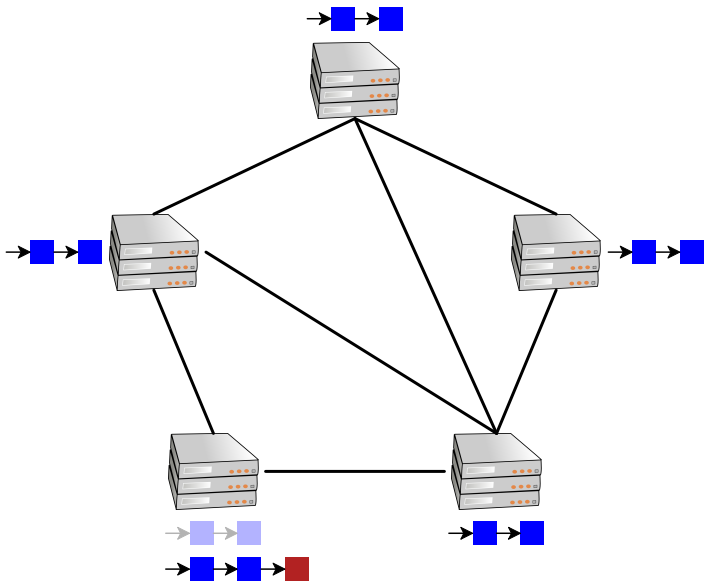
Consensus



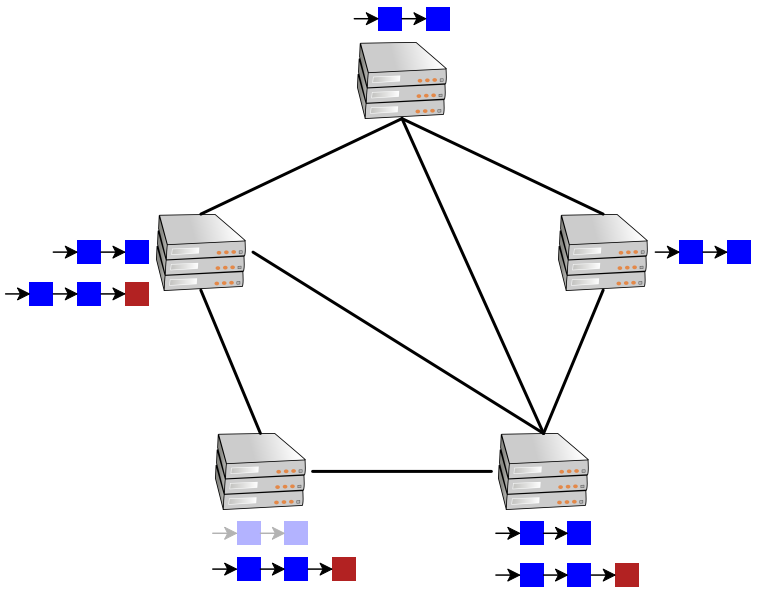
Consensus



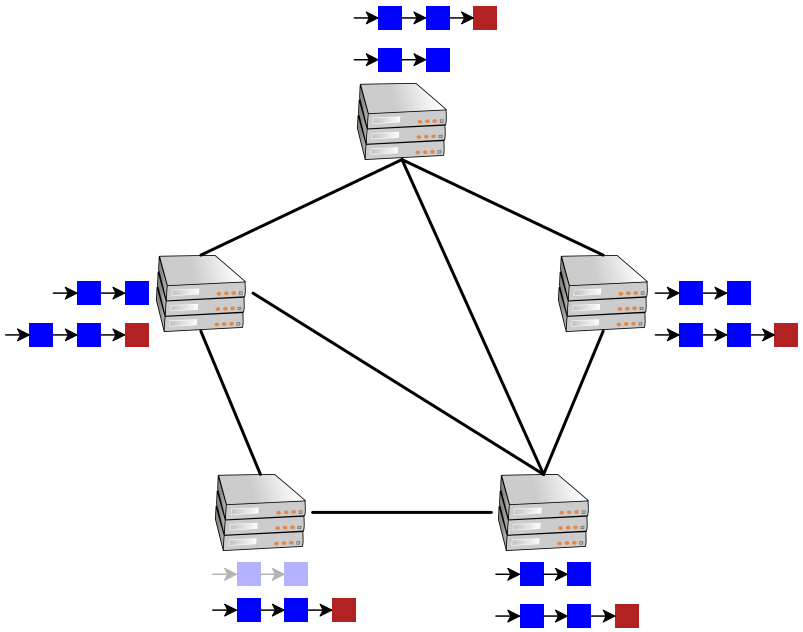
Consensus



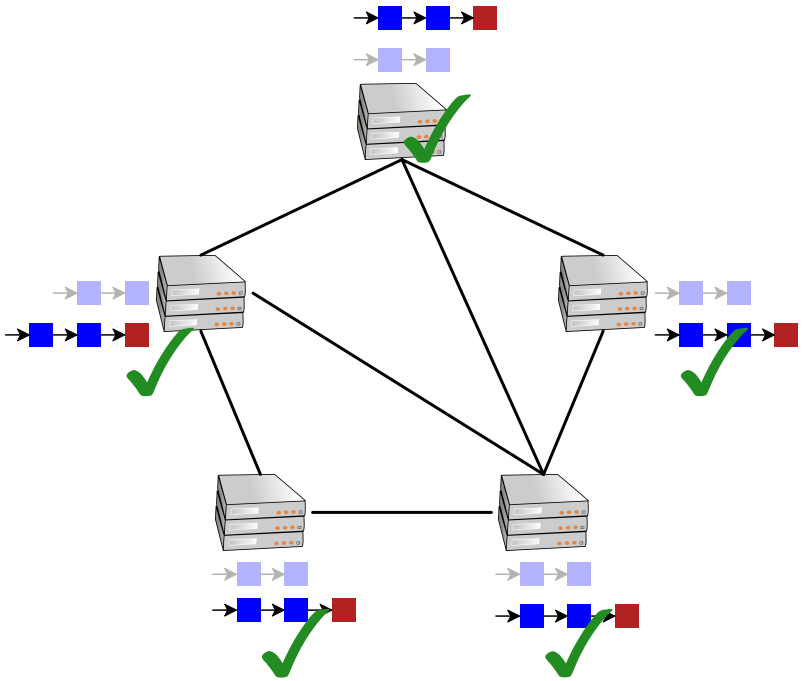
Consensus



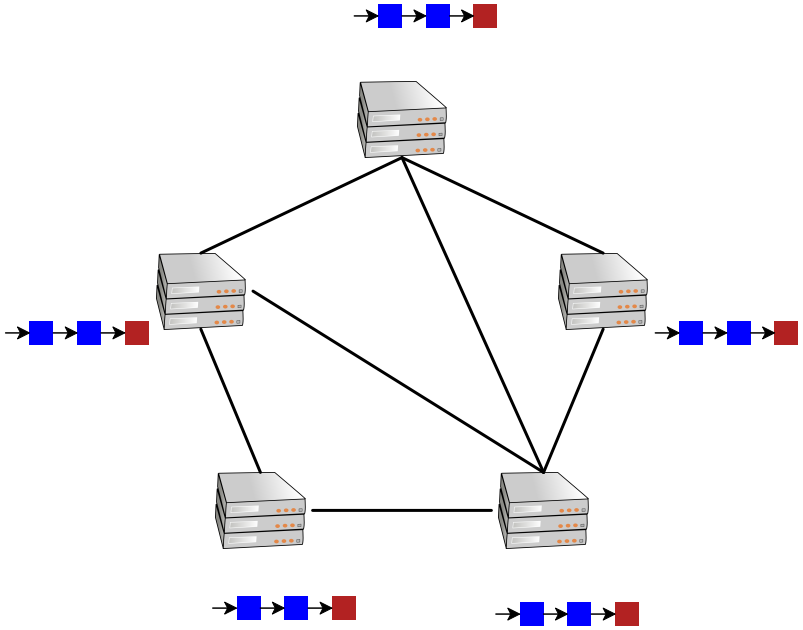
Consensus



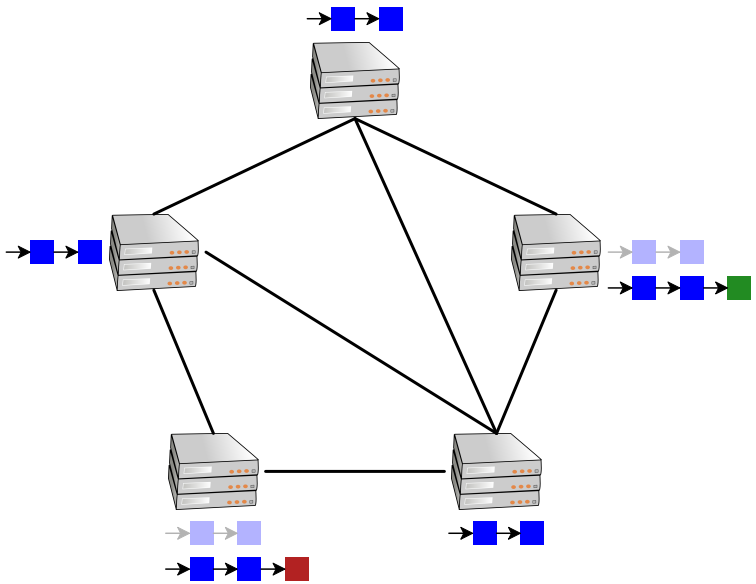
Consensus



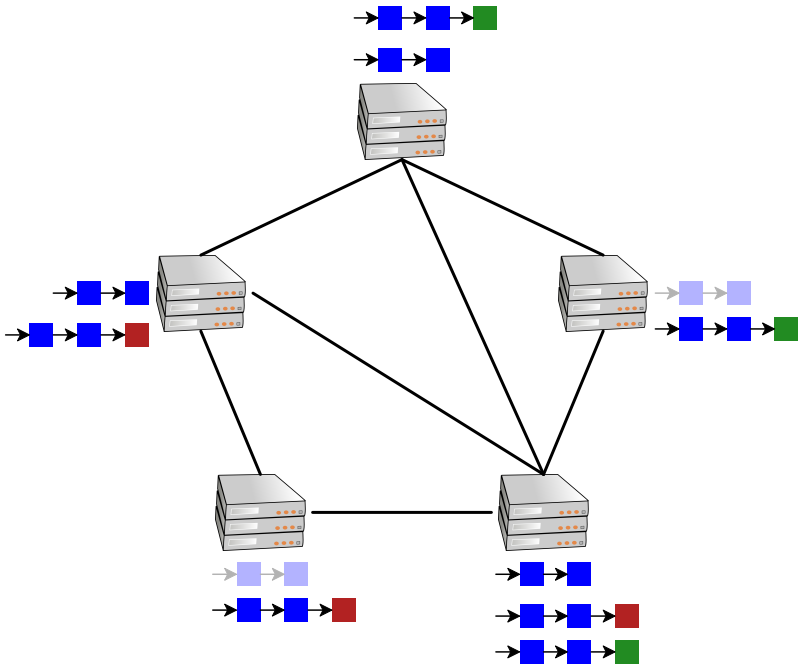
Consensus



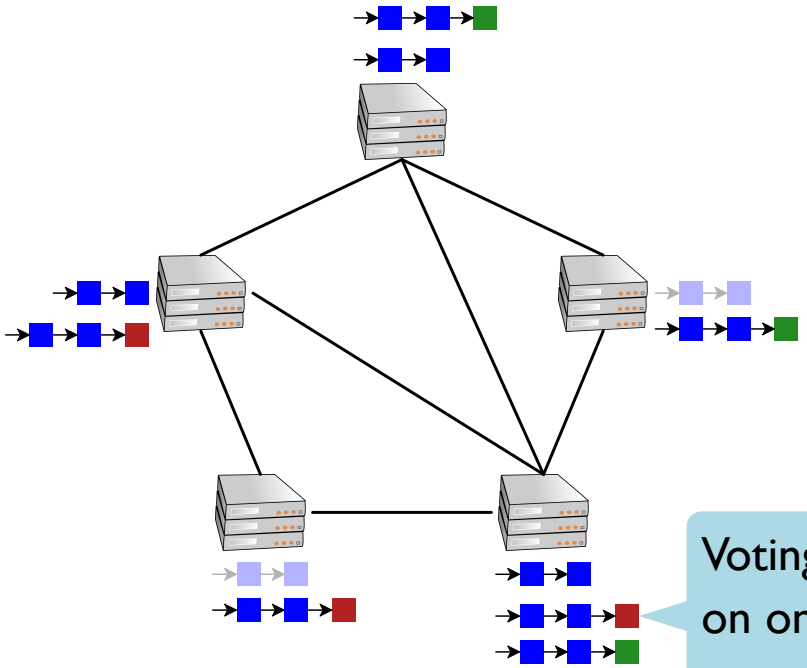
Consensus



Consensus

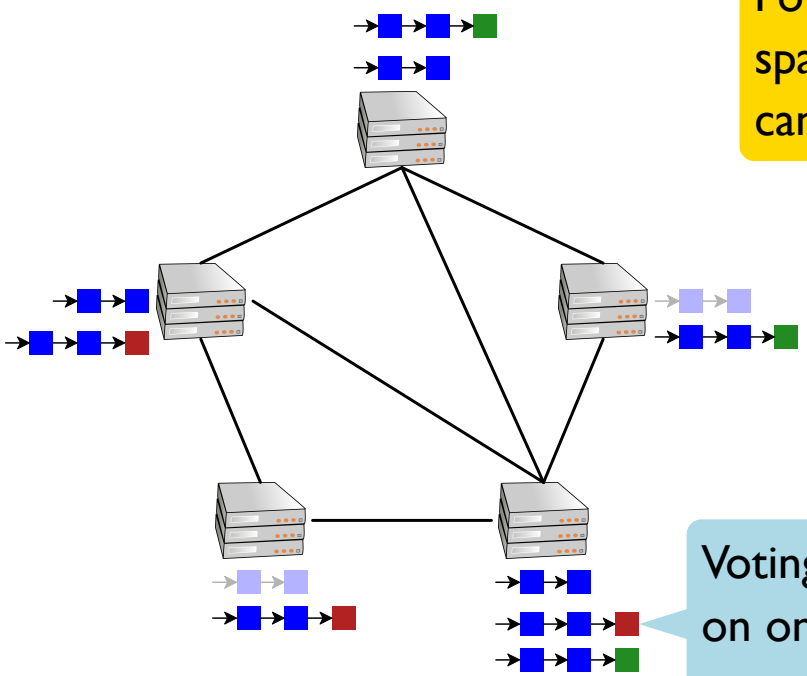


Consensus



Voting leads to agreement on one chain or the other, eventually

Consensus



Potential DoS attack: spam network with candidates and votes

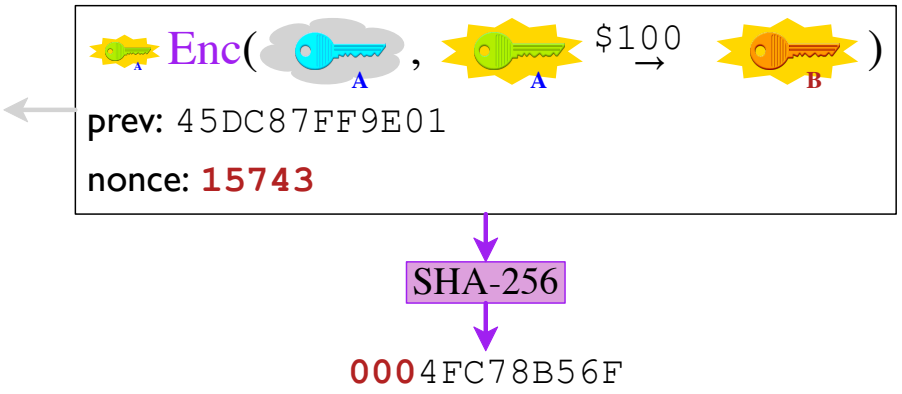
Voting leads to agreement on one chain or the other, eventually

Proof of Work

To defend against DoS attacks, make them prohibitively expensive

Vote only on candidates that provide evidence that substantial resources have been invested in the candidate, i.e., **proof of work**

In Bitcoin, proof of work takes the form of finding a nonce to include such that the resulting hash code starts with some number of 0s:



Scaling Up

To scale to a large number of transactions:

- Use a better data structure than a linked list

  ⇒ a **Merkle tree**

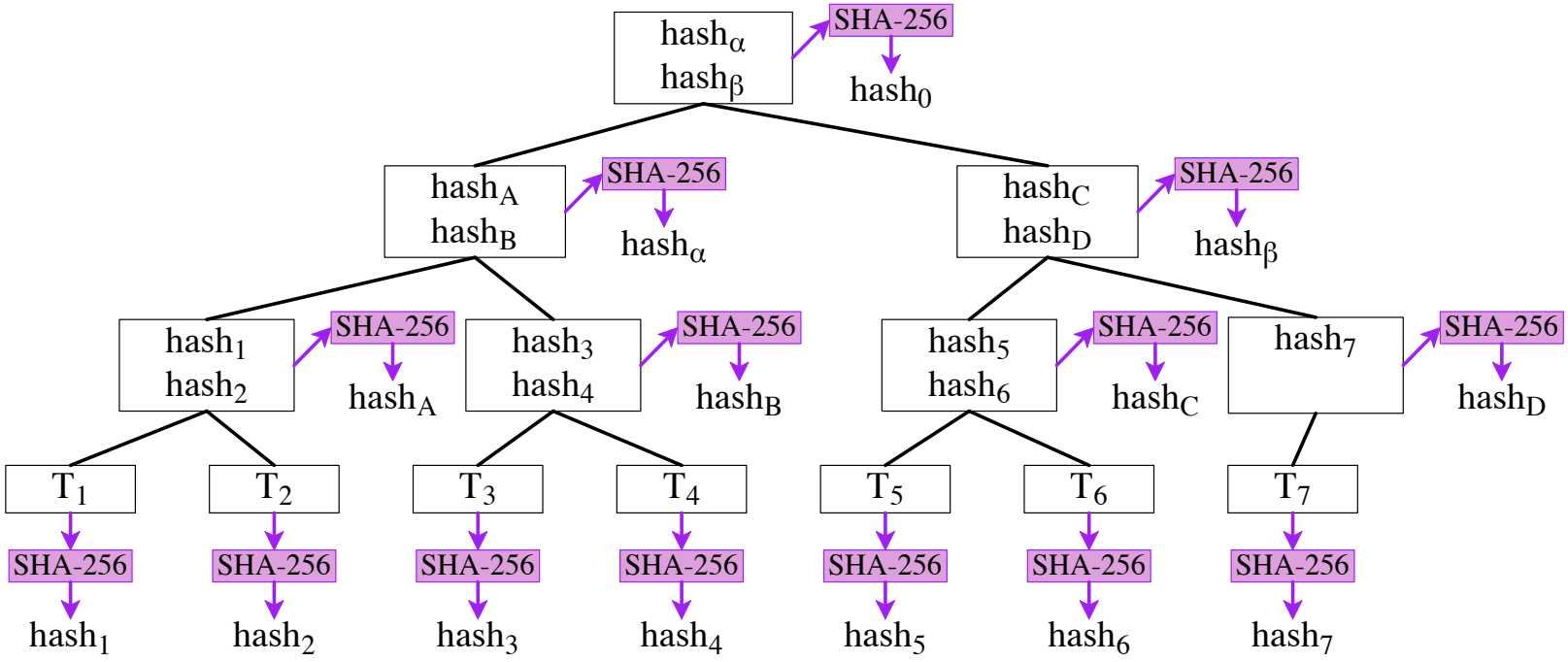
- Group multiple transactions into a single blockchain extension

  ⇒ incentives to perform transaction work

  via mining rewards and transaction fees

Merkle Tree

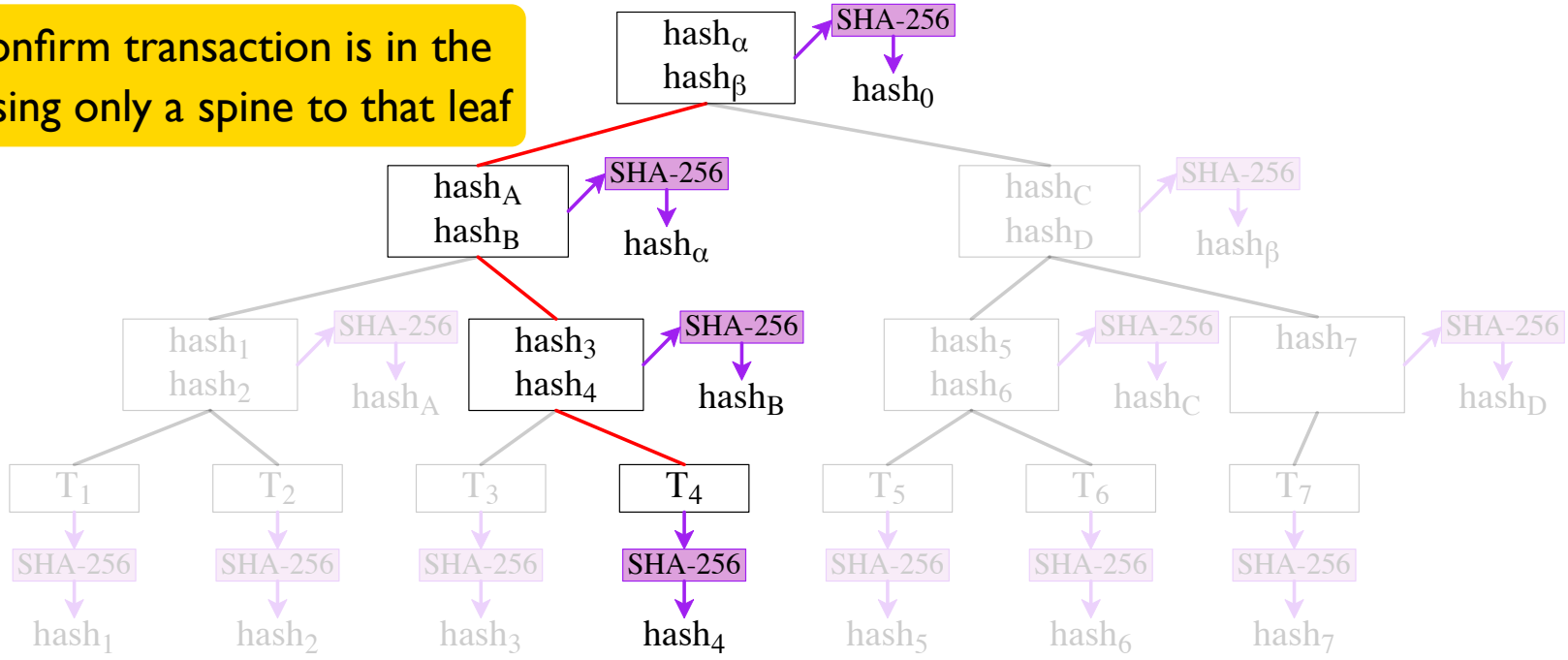
Instead of adding to the front of a linked list, add to the right of a balanced binary tree



Merkle Tree

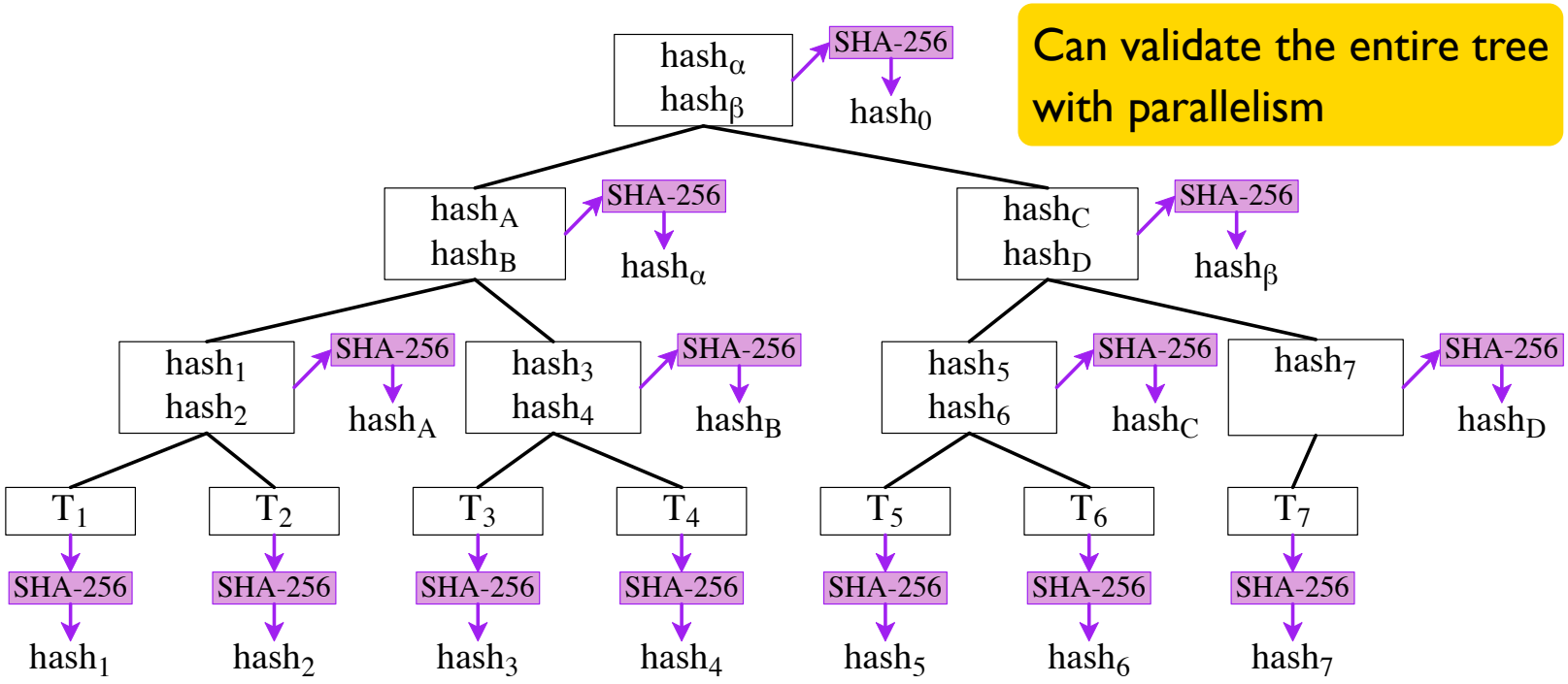
Instead of adding to the front of a linked list, add to the right of a balanced binary tree

Can confirm transaction is in the tree using only a spine to that leaf



Merkle Tree

Instead of adding to the front of a linked list, add to the right of a balanced binary tree



Transaction Processing Incentives

Batching multiple transactions reduces consensus traffic

... but why do the work (and prove it) for someone else's transactions?

Successful addition to the blockchain is rewarded in two ways:

- Transaction fees: paid by transaction parties
- Mining: compensated via newly minted money

Bitcoin: mining reward decays until the year 2140

Proof of Stake

A problem with **proof of work**: it's literally busy work

Energy expended for Bitcoin mining = medium-sized country

Proof of stake is an alternative where voting is modulated not by how much work you do, but how invested you are

prevents DoS, because participants want system to work

Etherium switched from proof of work to proof of stake

A problem with **proof of stake**: the rich get richer

Blockchain Expressiveness

A Bitcoin transaction is a simple script

An Etherium transaction is a program in a Turing-complete language

- supports **smart contracts** whose effect depends on future activity
- each transaction specifies **gas** to be used, which affects its cost

Summary

A **blockchain** implements non-repudiation via a public record of authorized transactions

A **consensus** protocol is needed to maintain the record in a distributed way, which avoids the need for a central trusted party

A **Merkle tree** provides integrity with log-time access

Two mechanisms to prevent DoS: **proof of work** and **proof of stake**