Spring 2022

区块链技术
Blockchain Technologies

密码学基础
Intro to cryptography

What is a blockchain?

Abstract answer: a blockchain provides

- coordination between many parties,
- when there is no single trusted party

if trusted party exists $\Rightarrow$ no need for a blockchain

[financial systems: often no trusted party]

What is all the excitement about?

(1) Basic application: a digital currency (stored value)

- Current largest: Bitcoin (2009), Ethereum (2015)
- Global: accessible to anyone with an Internet connection

Opinion

The New York Times

Bitcoin Has Saved My Family

"Borderless money" is more than a buzzword when you live in a collapsing economy and a collapsing dictatorship.

By Carlos Hernández

Mr. Hernández is a Venezuelan economist.

Feb. 23, 2019

What is all the excitement about?

(1) Basic application: a digital currency (stored value)

- Current largest: Bitcoin (2009), Ethereum (2015)
- Global: accessible to anyone with an Internet connection

(2) Beyond stored value: **decentralized applications (DAPPs)**

- DeFi: financial applications managed by <u>public</u> programs
  - examples: stablecoins, lending, exchanges, ….
- Asset management (e.g., art, domain names, games).
- Decentralized organizations (DAOs)
  - DAOs for 投资、捐赠、艺术品收藏…

(3) New programming model: writing decentralized programs

Transaction volume

(Sep, 2020)

	24h Volume
Bitcoin	$70,163,302,153
Ethereum	$62,307,903,847
Tether	$52,715,790,830
XRP	$1,724,384,881

Central Bank Digital Currency (CBDC)

What is a blockchain?

Layer 3: **user facing tools** (cloud servers)

Layer 2: **applications** (DAPPs, smart contracts)

Layer 1.5: **compute layer** (blockchain computer)

Layer 1: **consensus layer**

Consensus layer (informal)

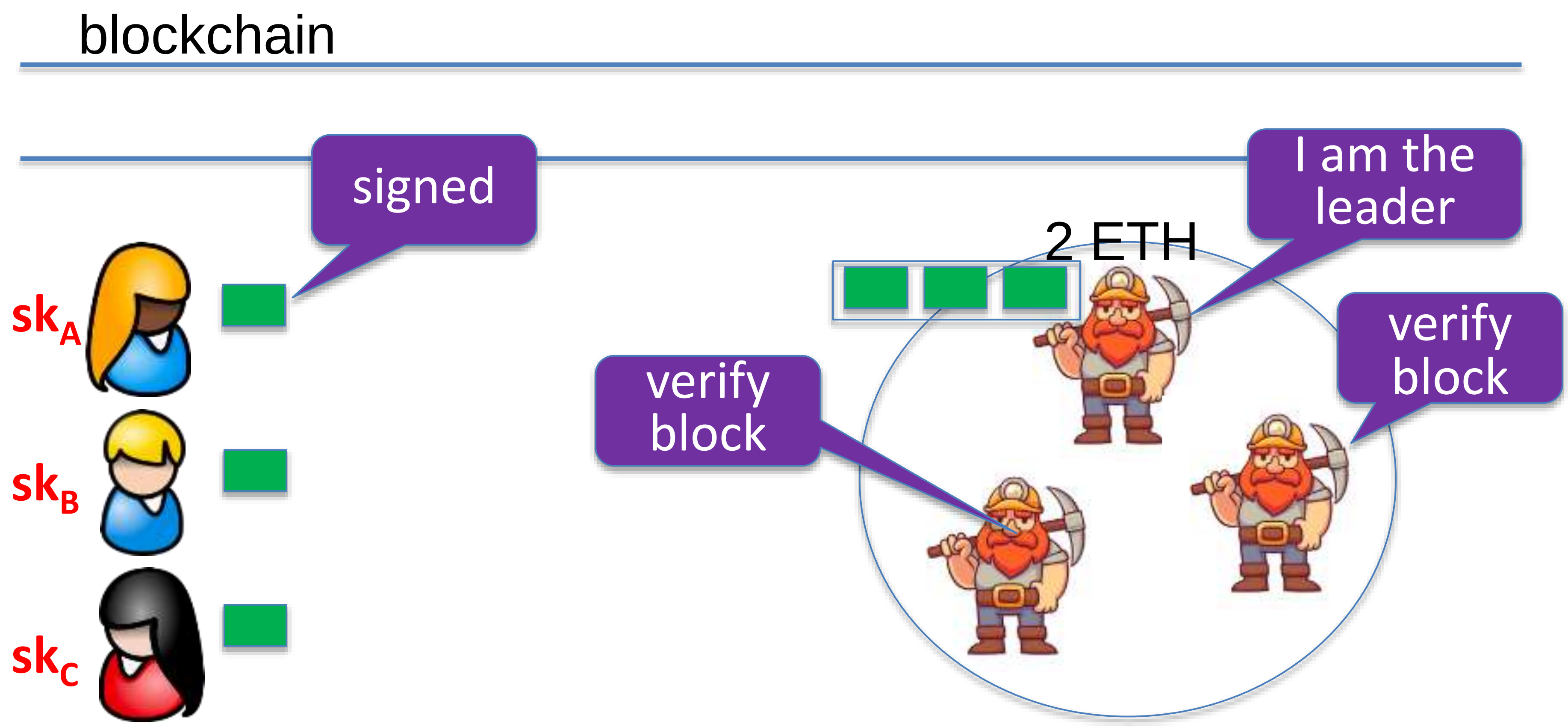
achieved by replication

A **public** append-only data structure:

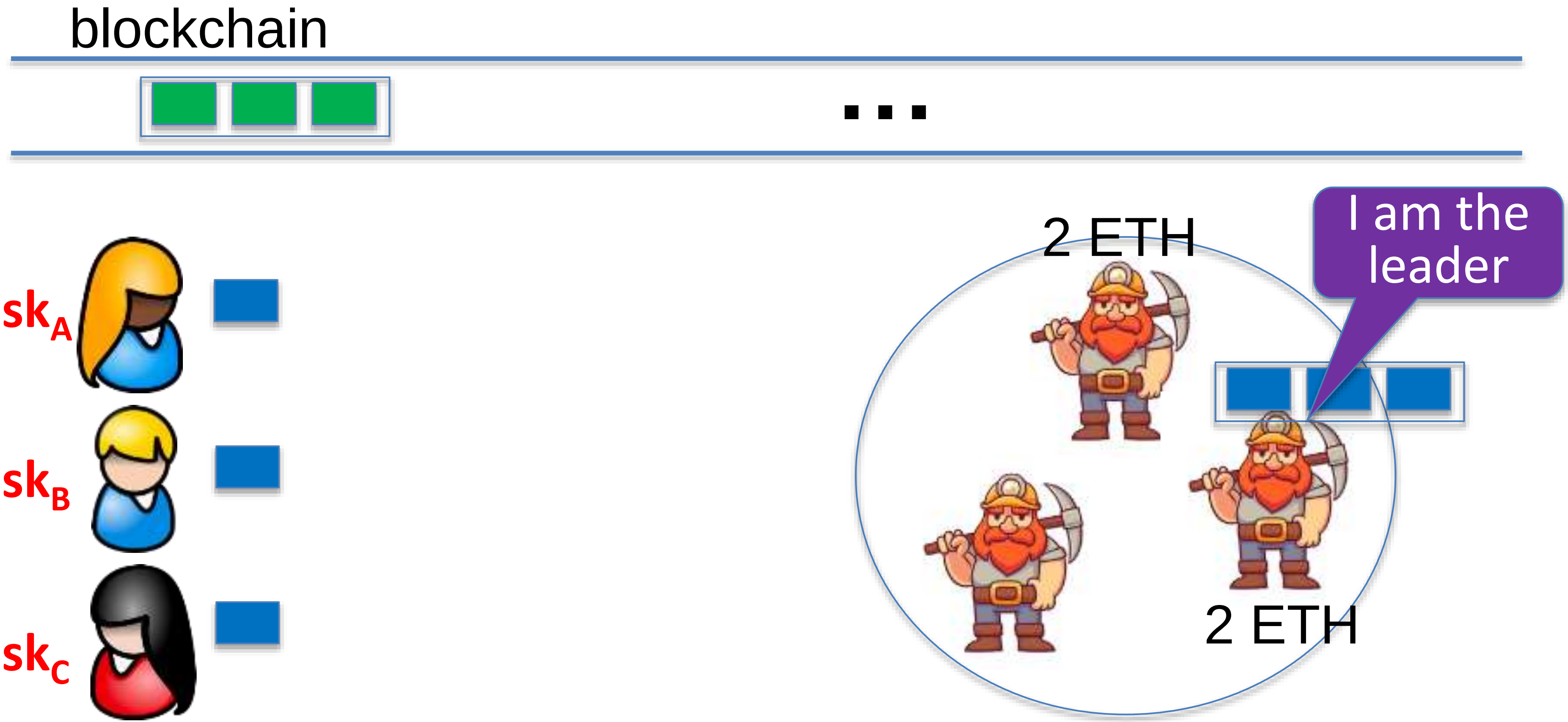
- **Persistence**: once added, data can never be removed*
- **Consensus**: all honest participants have the same data**
- **Liveness:** honest participants can add new transactions
- **Open(?)**: anyone can add data (no authentication)

Layer 1: consensus layer

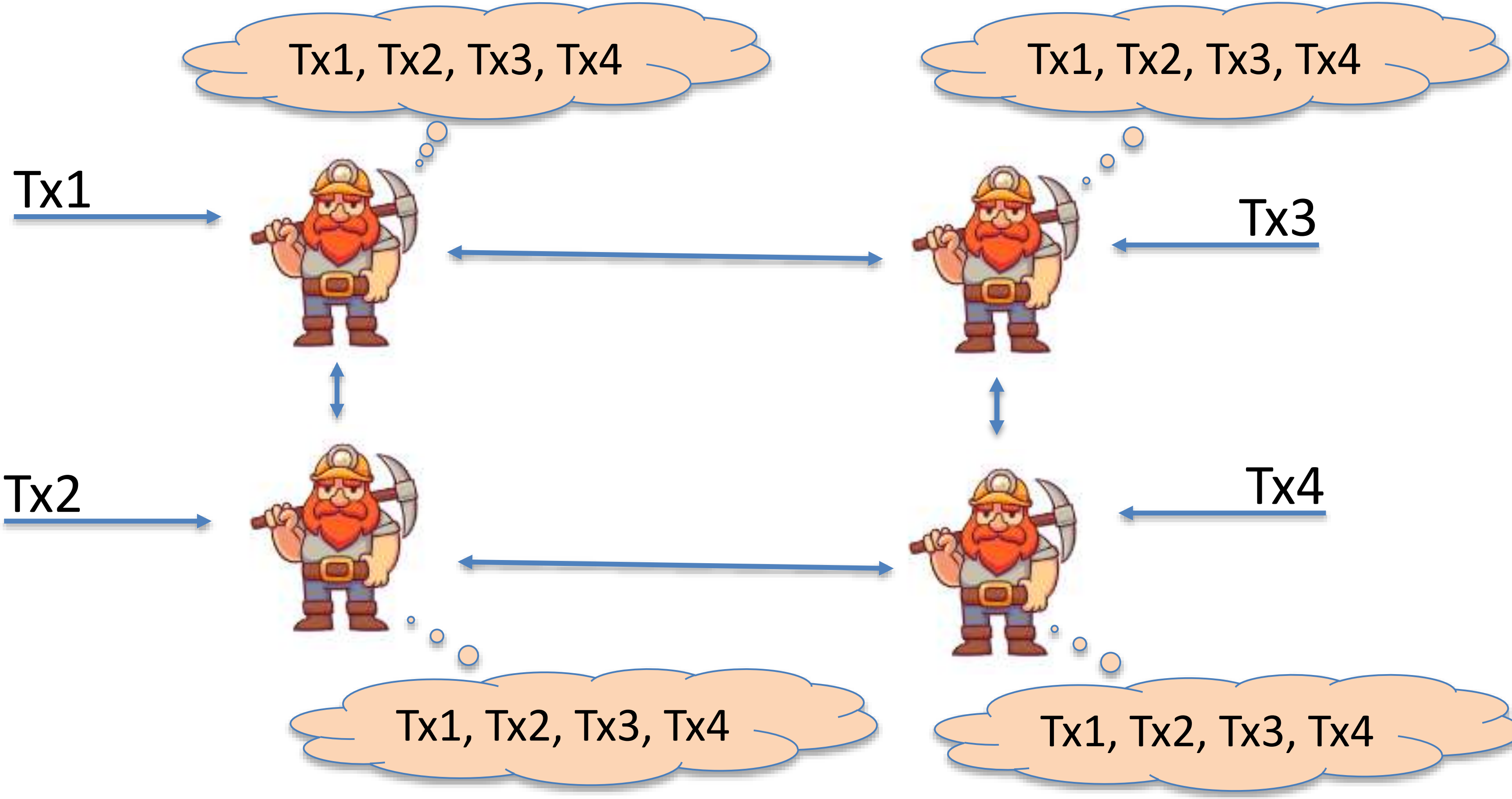
How are blocks added to chain?



How are blocks added to chain?



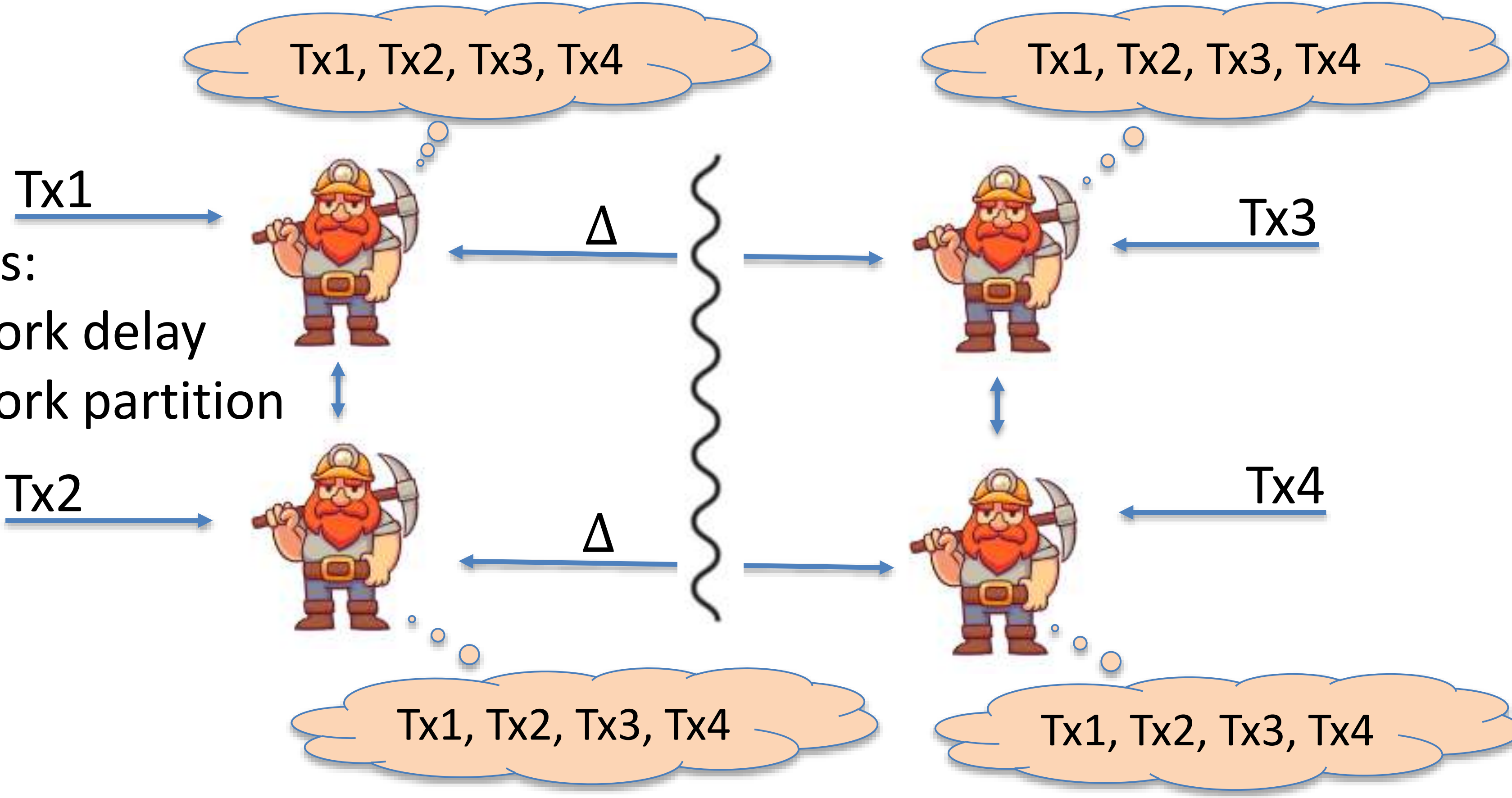
Why is consensus a hard problem?



Why is consensus a hard problem?

Problems:

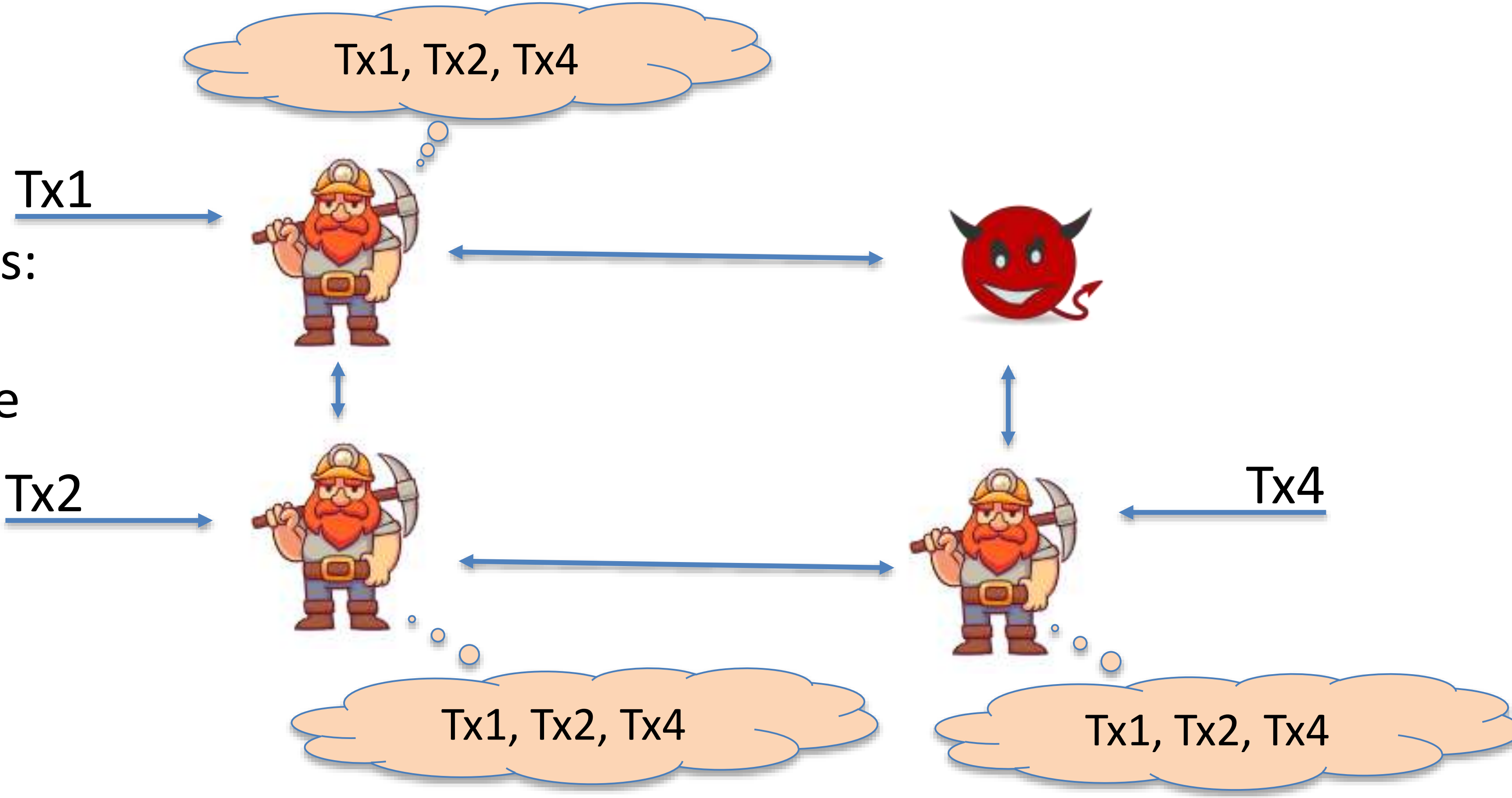
- Network delay
- Network partition



Why is consensus a hard problem?

Problems:

- crash
- malice



Layer 1.5: The blockchain computer

DAPP logic is encoded in a program that runs on blockchain

- Rules are enforced by a public program (public source code)

  ⇒ **transparency**: no single trusted 3rd party

- The DAPP program is executed by parties who create new blocks

  ⇒ **public verifiability**: everyone can verify state transitions

Layer 1.5:	compute layer
Layer 1:	consensus layer

Layer 2: Decentralized applications (DAPPS)

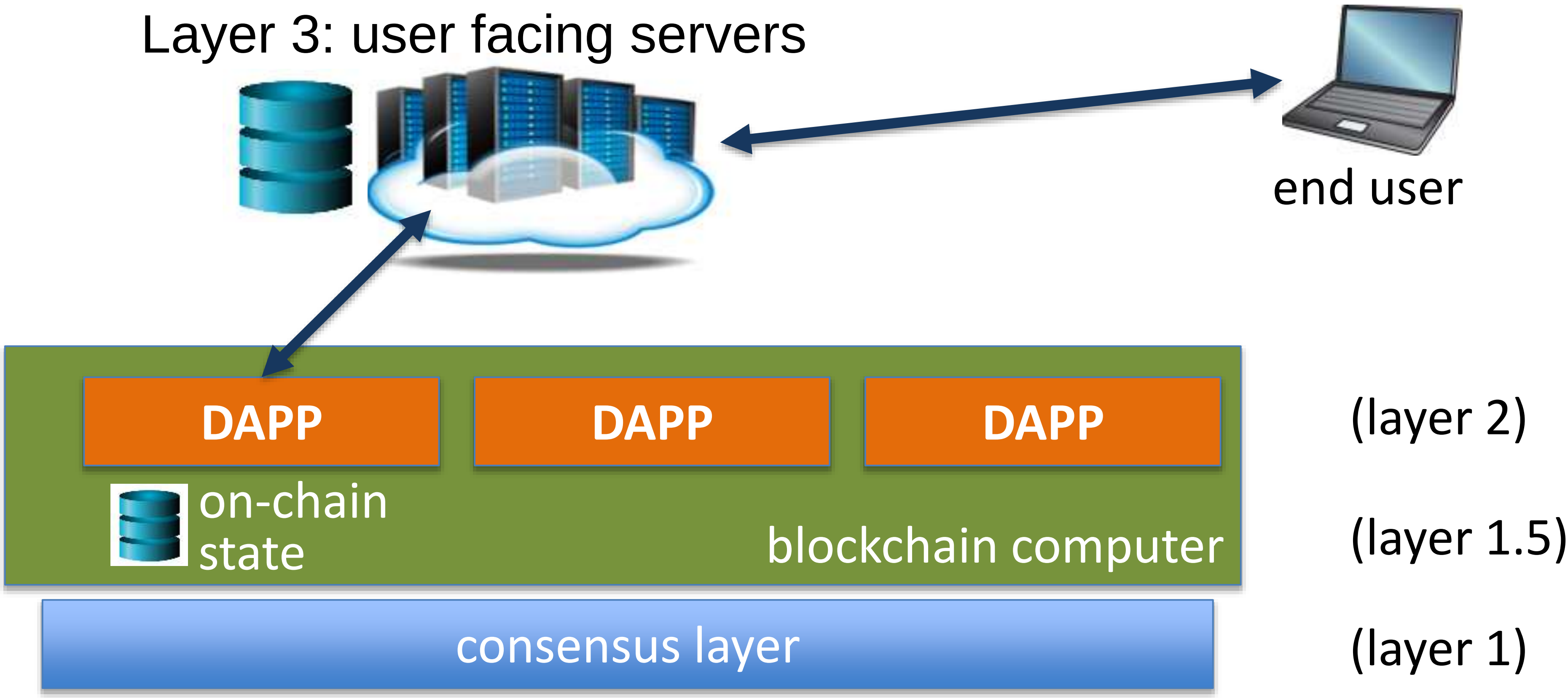
Run on blockchain computer

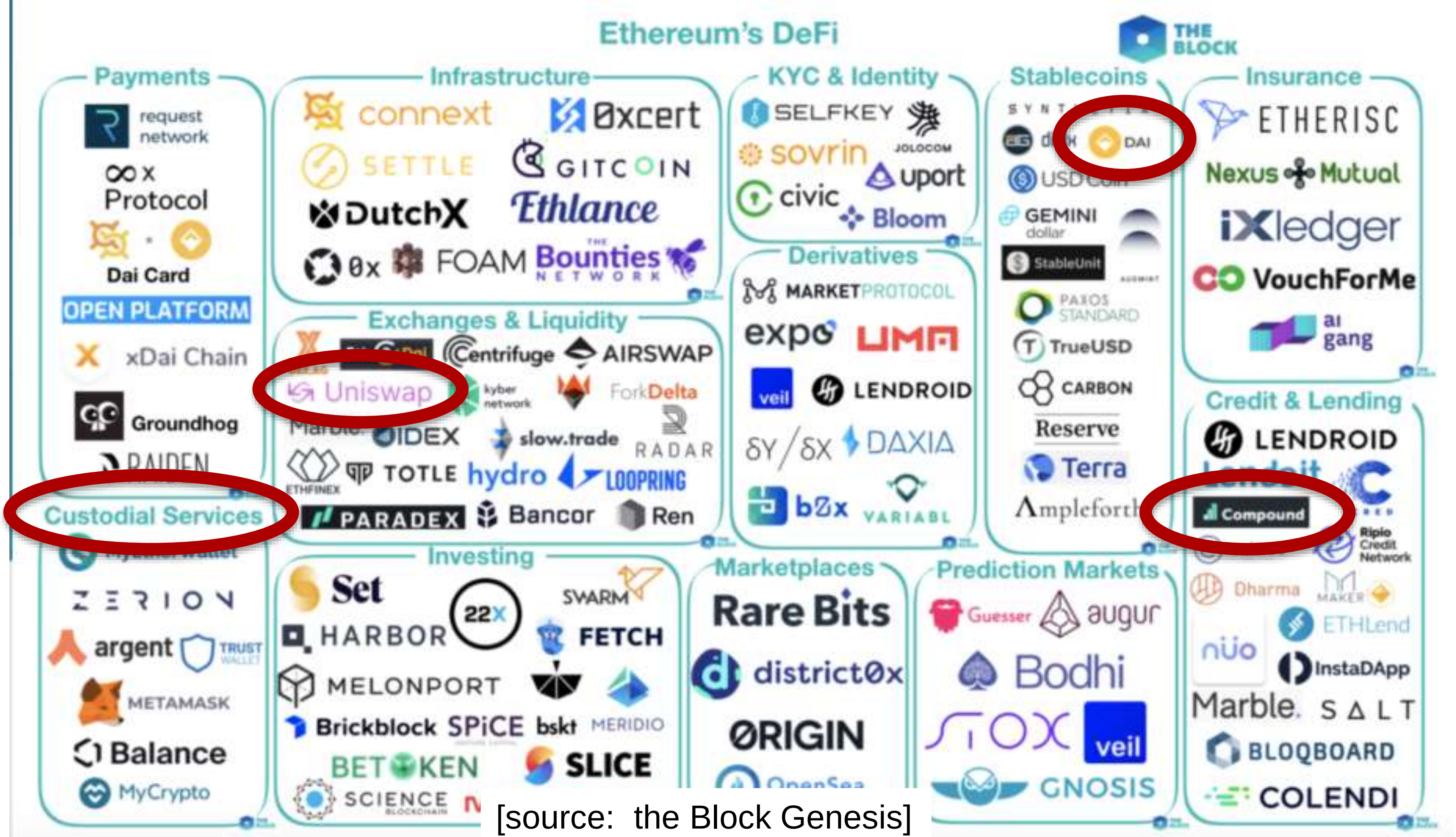
Layer 2: **applications** (DAPPs, smart contracts)

Layer 1.5: blockchain computer

Layer 1: consensus layer

Layer 3: Common DAPP architecture



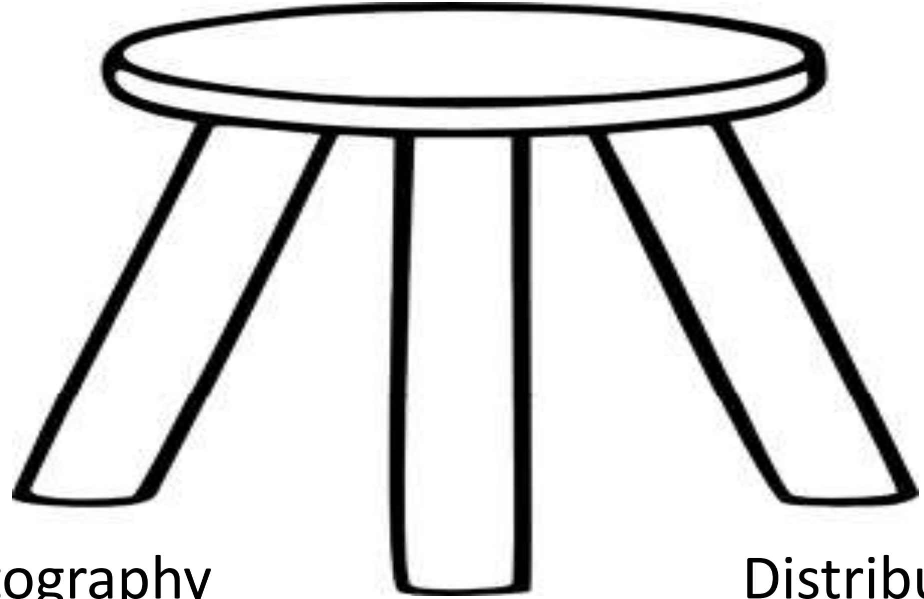


[source: the Block Genesis]

lots of experiments …

DEFI PULSE	Name	Chain	Category	Locked (USD) ▼
🏆 1.	Aave	Ethereum	Lending	$1.49B
🥈 2.	Maker	Ethereum	Lending	$1.26B
🥉 3.	Curve Finance	Ethereum	DEXes	$1.00B
4.	yearn.finance	Ethereum	Assets	$785.8M
5.	Synthetix	Ethereum	Derivatives	$769.4M
6.	Compound	Ethereum	Lending	$626.5M
7.	WBTC	Ethereum	Assets	$570.7M
8.	Uniswap	Ethereum	DEXes	$564.5M

This course



Cryptography

Distributed systems

Economics

Course organization

1. The starting point: Bitcoin mechanics
2. Consensus protocols
3. Ethereum and decentralized applications
4. Economics of decentralized applications
5. Scaling the blockchain: 10K Tx/sec and more
6. Private transactions on a public blockchain
   (SNARKs and zero knowledge proofs)
7. 跨链互操作性：bridges and wrapped coins

Let’s get started …

请随时提出问题，不要等到期末！

Cryptography Background

(1) cryptographic hash functions

An efficiently computable function $H: M \rightarrow T$

where $|M| \gg |T|$

$T = \{0,1\}256$

Collision resistance(抗碰撞)

<u>Def</u>: a **<u>collision</u>** for $H: M \to T$ is pair $x \neq y \in M$ s.t. $H(x) = H(y)$

$|M| \gg |T|$ implies that <u>many</u> collisions exist

<u>Def:</u> a function $H: M \to T$ is **<u>collision resistant</u>** if it is "hard" to find even a single collision for H (we say H is a CRHF)

Example: **SHA256**: $\{x : \text{len}(x) < 2^{64} \text{ bytes}\} \to \{0,1\}^{256}$

An application: committing to data(承诺)

Alice has a large file m. She publishes $h = H(m)$ (32 bytes)

Bob has h. Later he learns m' s.t. $H(m') = h$

H is a CRHF $\Rightarrow$ Bob is convinced that $m' = m$

(otherwise, m and m' are a collision for H)

We say that $h = H(m)$ is a **binding commitment (绑定性)**to m

(note: not hiding, h may leak information about m)

（隐匿性有限，不具备随机性，对同一个敏感数据，H(v)值总是固定的）

Committing to a list (of transactions)

Alice has $S = (m_1, m_2, \ldots, m_n)$

32 bytes

Goal:

- Alice publishes a <u>short</u> binding commitment to S, $h = \text{commit}(S)$
- Bob has h. Given $(m_i,\ \text{proof}\ \pi_i)$ can check that $S[i] = mi$

  Bob runs $\text{verify}(h, i, m_i, \pi_i) \rightarrow$ accept/reject

security: adv. cannot find (S, i, m, π) s.t. $m \neq S[i]$ and

$\text{verify}(h, i, m, \pi) = \text{accept}$ where $h = \text{commit}(S)$

Committing to a list

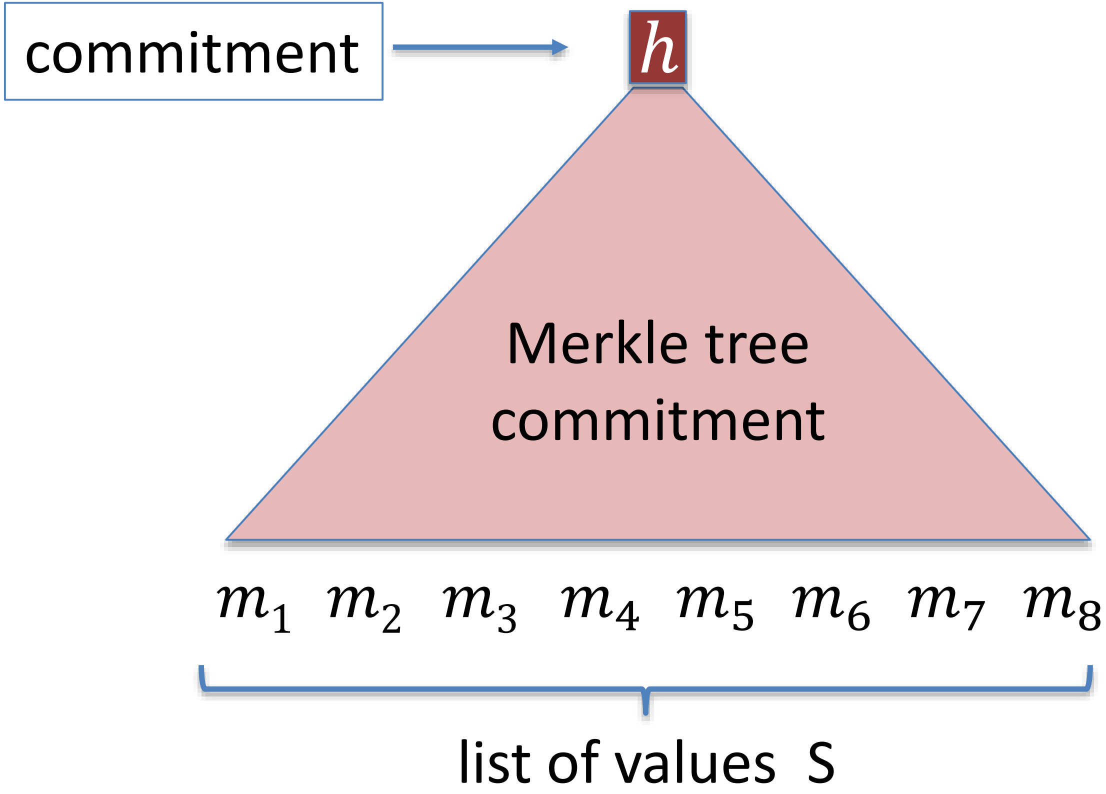
method 1: commit(S) $= h = H(H(m_1), \ldots, H(m_n))$

Later: given h, m_1 and $\underbrace{H(m_2), \ldots, H(m_n)}_{\text{proof } \pi_1}$ Bob can check $S[1] = m_1$

Problem: long proof! $(n-1)$ hash values

Better method: **Merkle tree.** Proof length = $\underline{\log_2 n}$ hash values

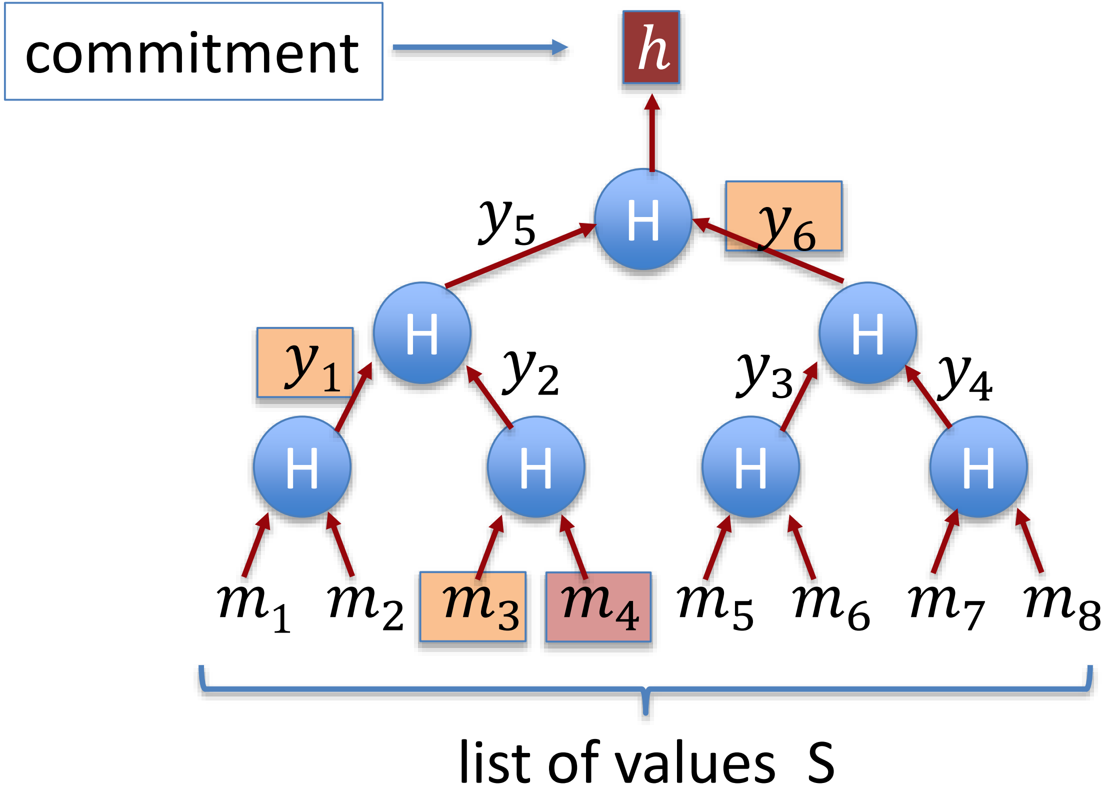
Merkle tree (Merkle 1989)



Goal:

- commit to list S
- Later prove $S[i] = mi$

Merkle tree (Merkle 1989)



Goal:

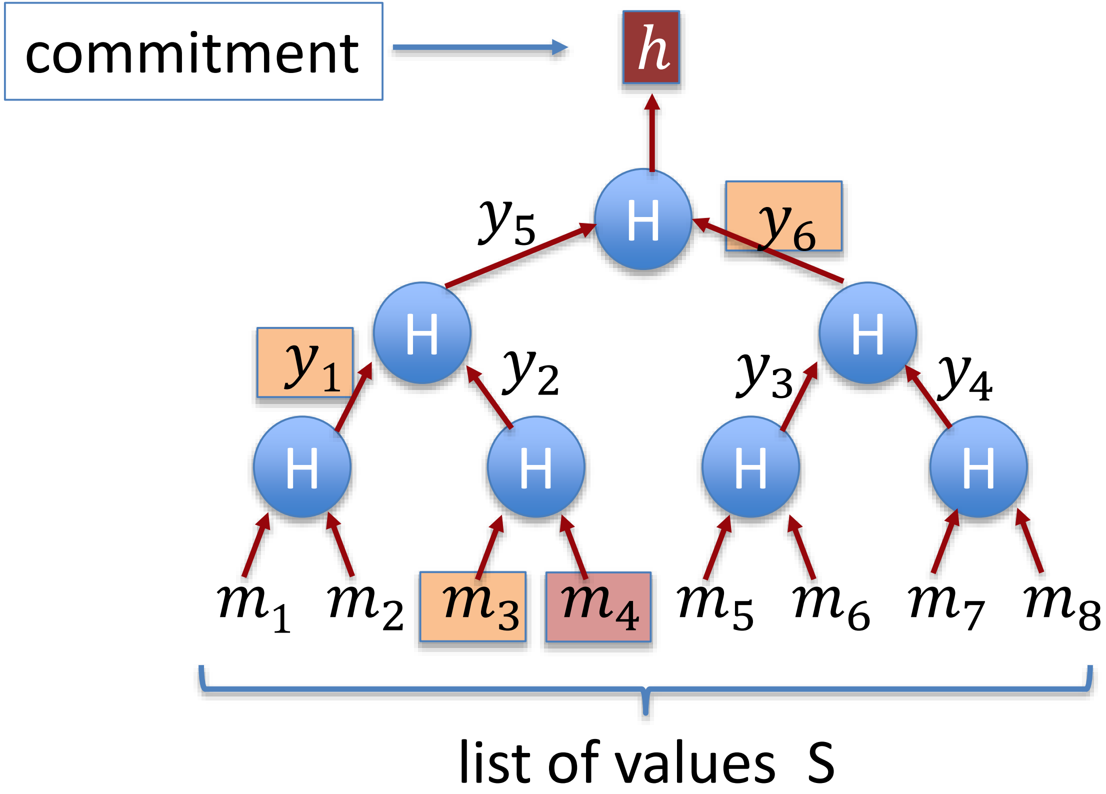
- commit to list S
- Later prove $S[i] = mi$

To prove $S[4] = m_4$,

proof $\pi = (m_3, y_1, y_6)$

length of π: $\log_2 |S|$

Merkle tree (Merkle 1989)



To prove $S[4] = m_4$,

proof $\pi = (m_3, y_1, y_6)$

Bob does:

$y_2 \leftarrow H(m_3, m_4)$

$y_5 \leftarrow H(y_1, y_2)$

$h' \leftarrow H(y_5, y_6)$

accept if $h = h'$

Merkle tree (Merkle 1989)

Thm: H CRHF $\Rightarrow$ adv. cannot find (S, i, m, π) s.t. $m \neq S[i]$ and

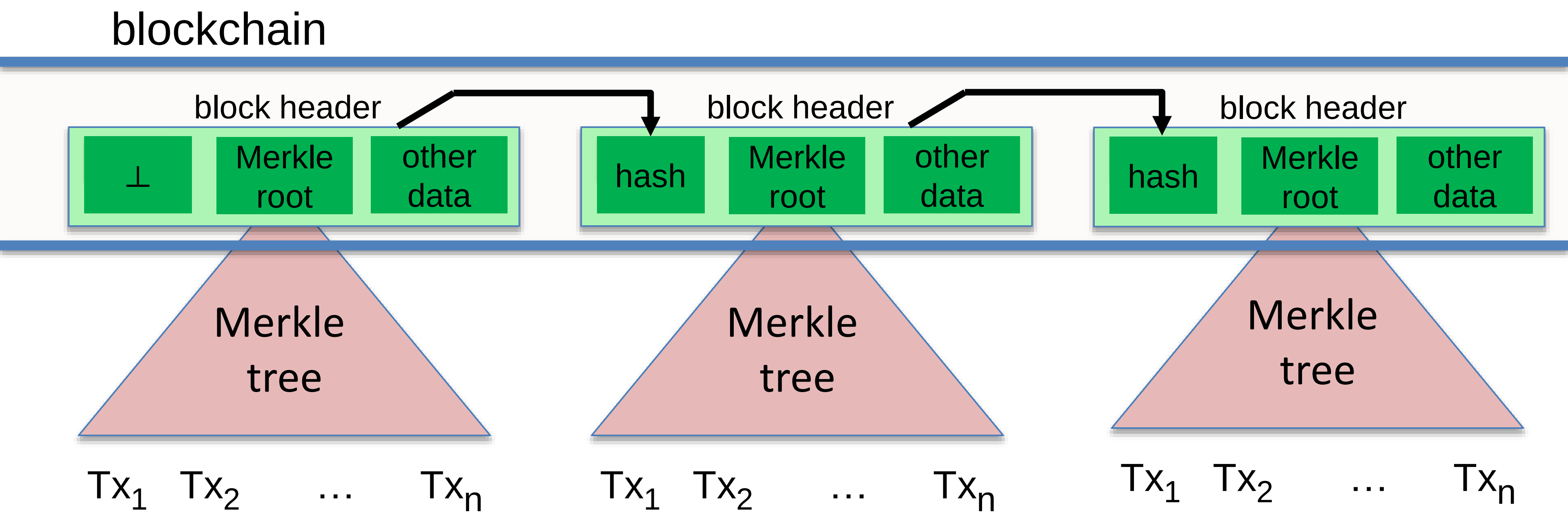
$$\text{verify}(h, i, m, \pi) = \text{accept} \quad \text{where} \quad h = \text{commit}(S)$$

(to prove, prove the contra-positive)

How is this useful? Super useful. Example

- When writing a block of transactions S to the blockchain, suffices to write commit(S) to chain. Keep chain small.
- Later, can prove contents of every Tx.

Abstract block chain



Merkle proofs are used to prove that a Tx is "on the block chain"

Another application: proof of work

<u>**Goal**</u>: computational problem that
- takes time $\Omega(D)$ to solve, but
- solution takes time O(1) to verify

(D is called the **difficulty**)

How? $H: X \times Y \to \{0,1,2,\ldots,2^n - 1\}$ e.g. $n = 256$

- puzzle: input $x \in X$, output $y \in Y$ s.t. $H(x,y) < 2^n/D$
- verify(x, y): accept if $H(x,y) < 2^n/D$

Another application: proof of work

Thm: if H is a “random function” then the best algorithm requires D evaluations of H in expectation.

Note: this is a parallel algorithm

⇒ the more machines I have, the faster I solve the puzzle.

Bitcoin uses $H(x) = \text{SHA256}(\text{SHA256}(x))$

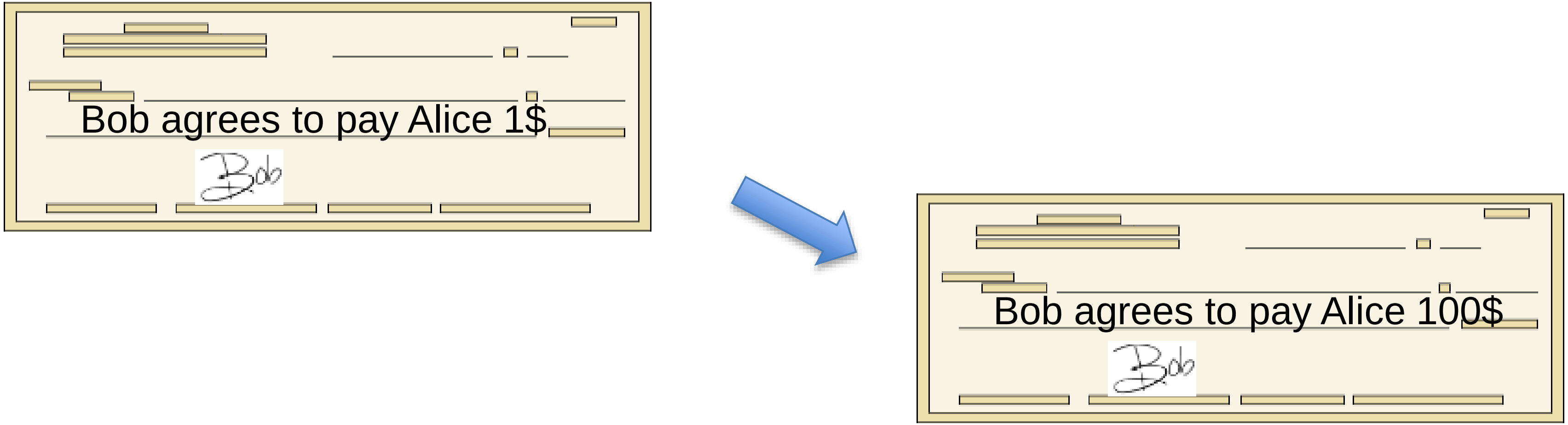
Cryptography background: Digital Signatures

数字签名

如何验证交易

Signatures

Physical signatures: bind transaction to author

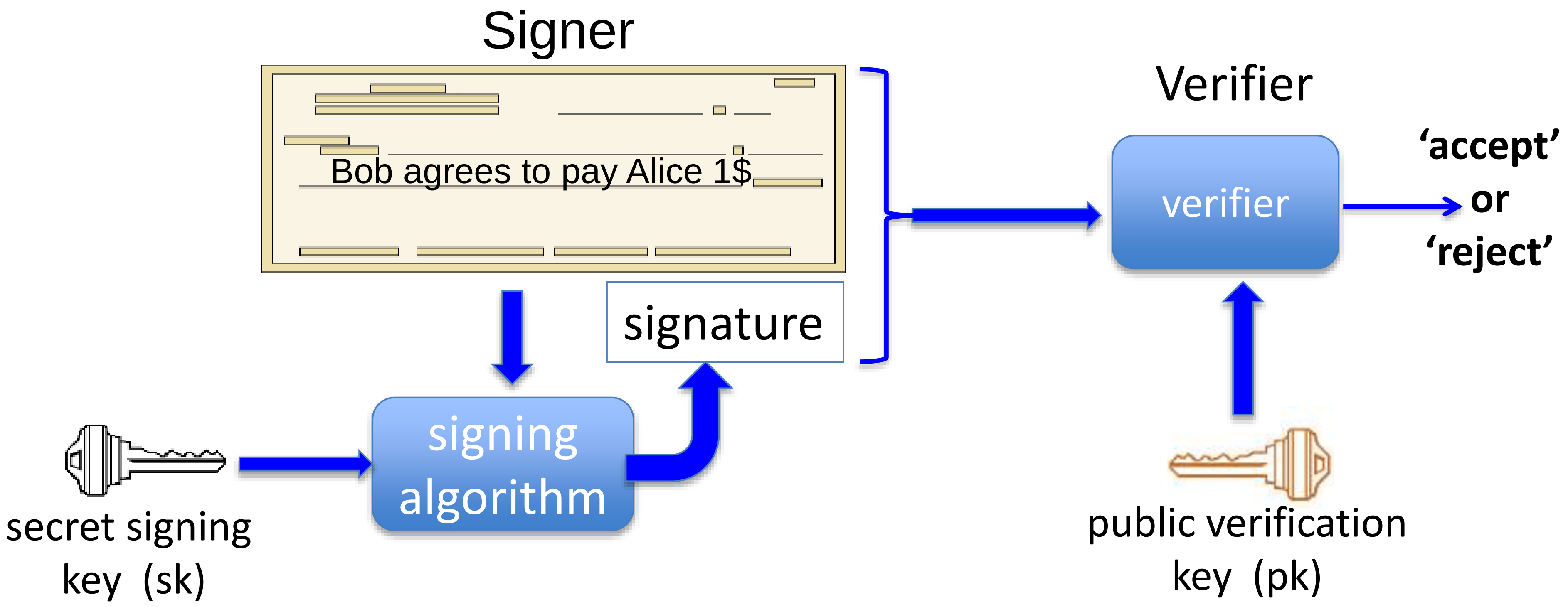


Problem in the digital world:

anyone can copy Bob’s signature from one doc to another

Digital signatures

Solution: make signature depend on document



Digital signatures: syntax

Def: a signature scheme is a triple of algorithms:

- **Gen**(): outputs a key pair (pk, sk)
- **Sign**(sk, msg) outputs sig. σ
- **Verify**(pk, msg, σ) outputs 'accept' or 'reject'

Secure signatures: (informal)

Adversary who sees signatures on many messages of his choice, cannot forge a signature on a new message.

Families of signature schemes

1. RSA signatures (old … not used in blockchains):
   - long sigs and public keys (≥256 bytes), fast to verify

2. Discrete-log signatures: Schnorr and ECDSA (Bitcoin, Ethereum)
   - short sigs (48 or 64 bytes) and public key (32 bytes)

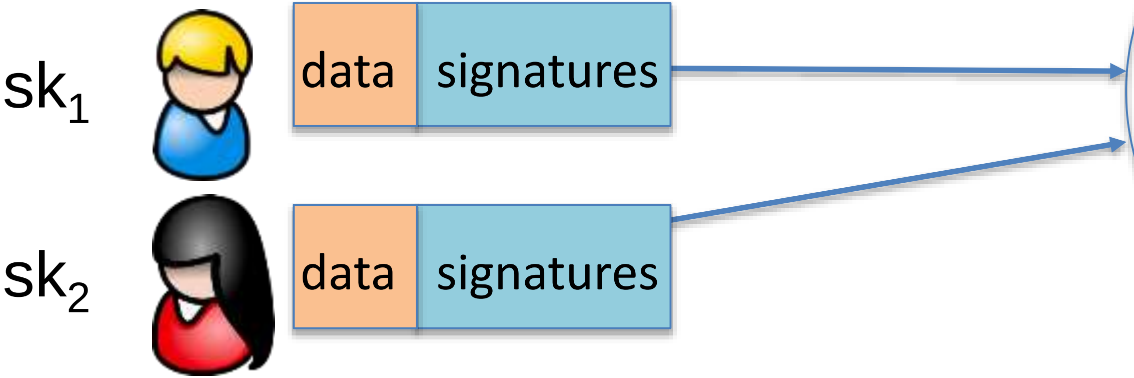
3. BLS signatures: 48 bytes, aggregatable, easy threshold
   (Ethereum 2.0, Chia, Dfinity)

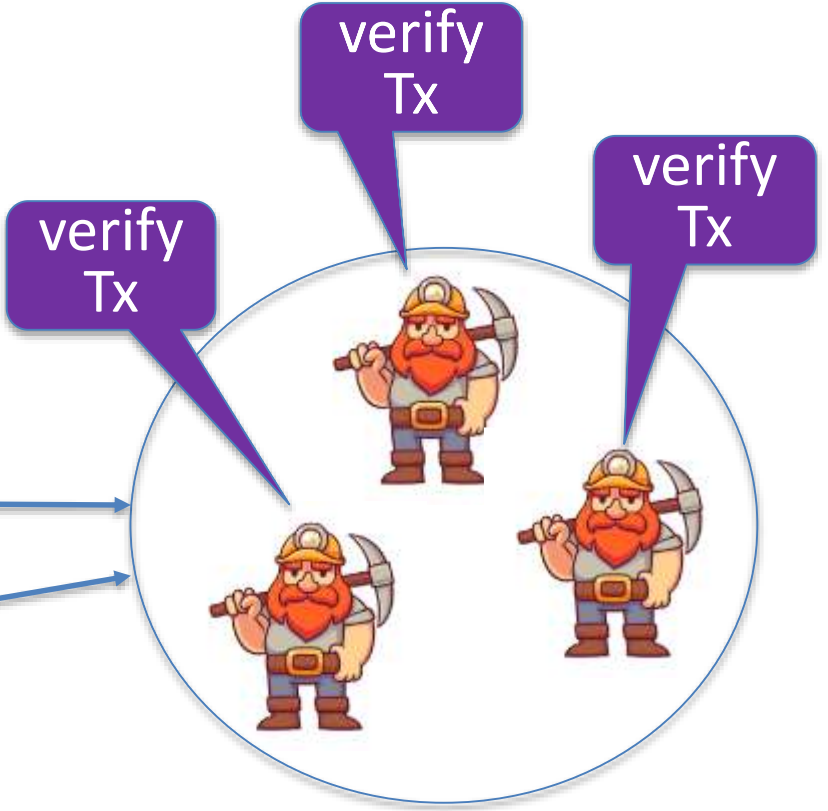
4. Post-quantum signatures: long (≥768 bytes)

Signatures on the blockchain

Signatures are used everywhere:

- ensure Tx authorization,
- governance votes,
- consensus protocol votes.





END OF LECTURE

Next lecture: the Bitcoin blockchain