CS251 Fall 2022

https://cs251.stanford.edu

Cryptocurrencies and Blockchain Technologies

Dan Boneh

Stanford University

[course videos on canvas, discussions on edstem, homework on gradescope]

[first project – Merkle trees – is out on the course web site]

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]

Blockchains: what is the new idea?

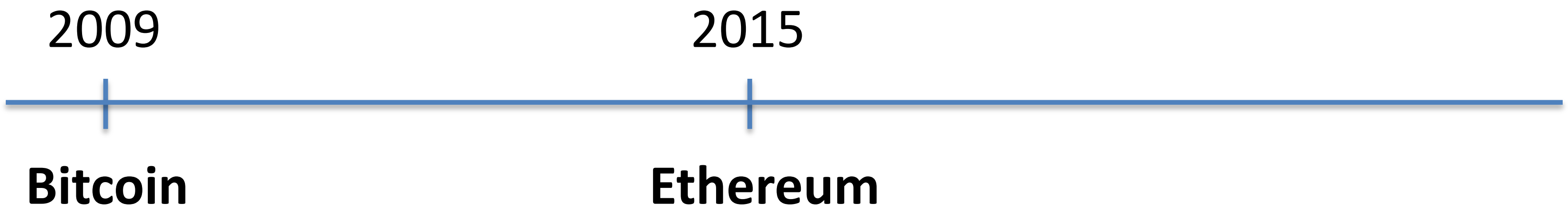
2009

Bitcoin

Several innovations:

- A practical **public append-only data structure**, secured by replication and incentives
- A fixed supply asset (BTC). Digital payments, and more.

Blockchains: what is the new idea?



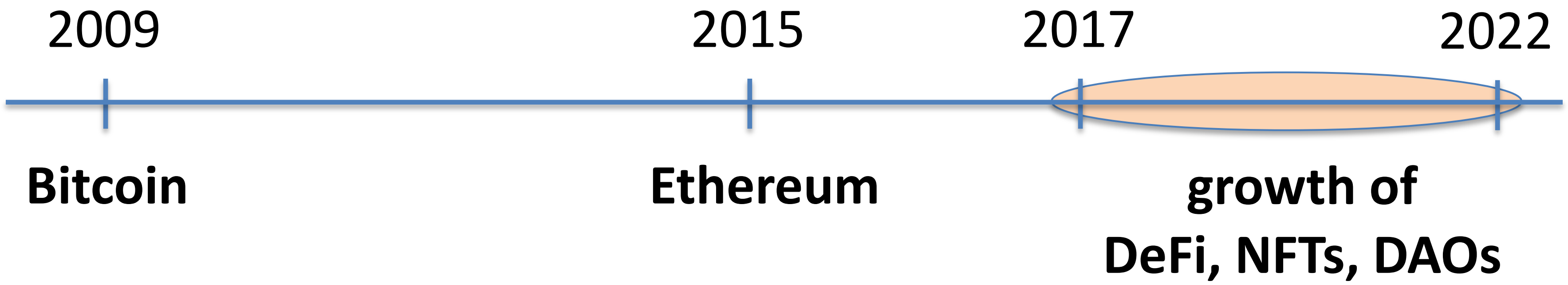
Several innovations:

- **Blockchain computer**: a fully programmable environment

  $\Longrightarrow$ public programs that manage digital and financial assets

- **Composability**: applications running on chain can call each other

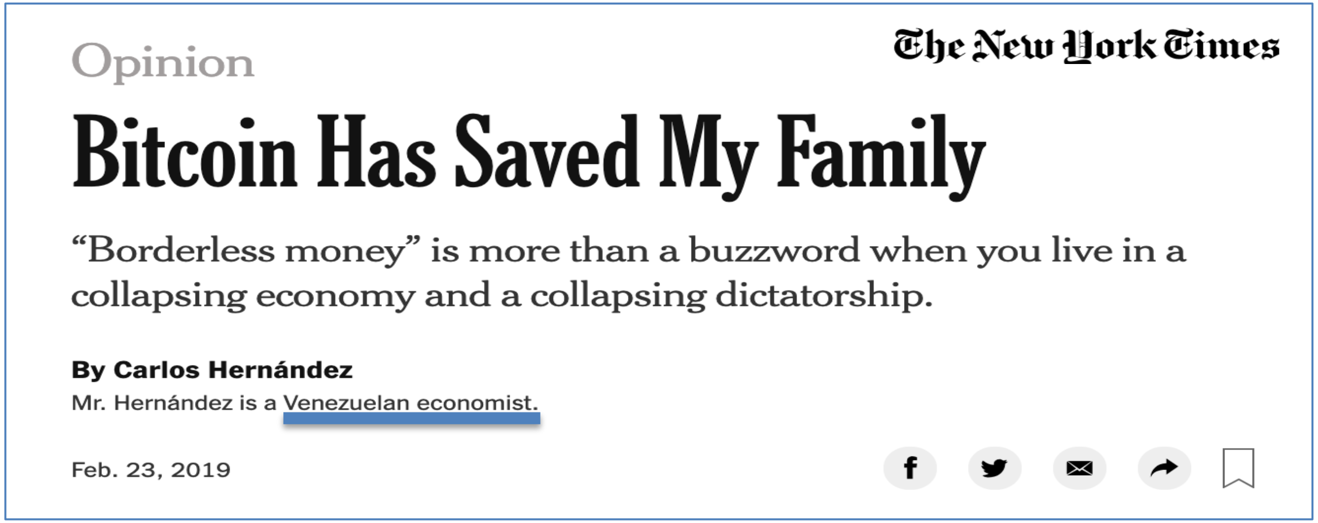
Blockchains: what is the new idea?



So what is this good for?

(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 else is it good for?

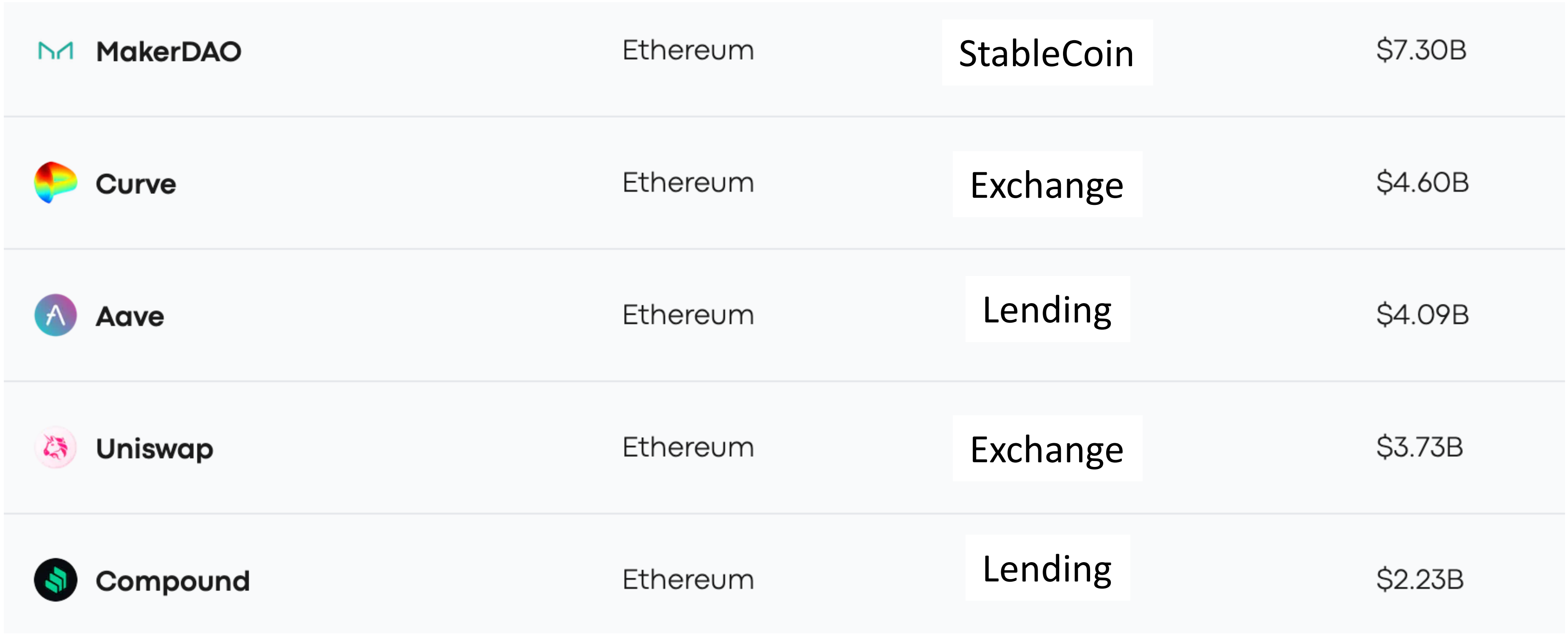
(2) Decentralized applications (DAPPs)

- **DeFi**: financial instruments managed by <u>public</u> programs
  - examples: stablecoins, lending, exchanges, ….
- **Asset management** (NFTs)**:** art, game assets, domain names.
- **Decentralized organizations** (DAOs): (decentralized governance)
  - DAOs for investment, for donations, for collecting art, etc.

---

(3) New programming model: writing decentralized programs

Assets managed by DAPPs

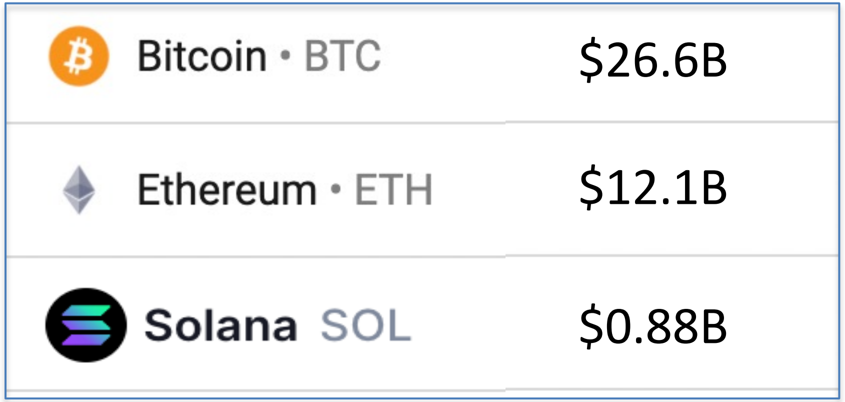


MakerDAO	Ethereum	StableCoin	$7.30B
Curve	Ethereum	Exchange	$4.60B
Aave	Ethereum	Lending	$4.09B
Uniswap	Ethereum	Exchange	$3.73B
Compound	Ethereum	Lending	$2.23B

Sep. 2022

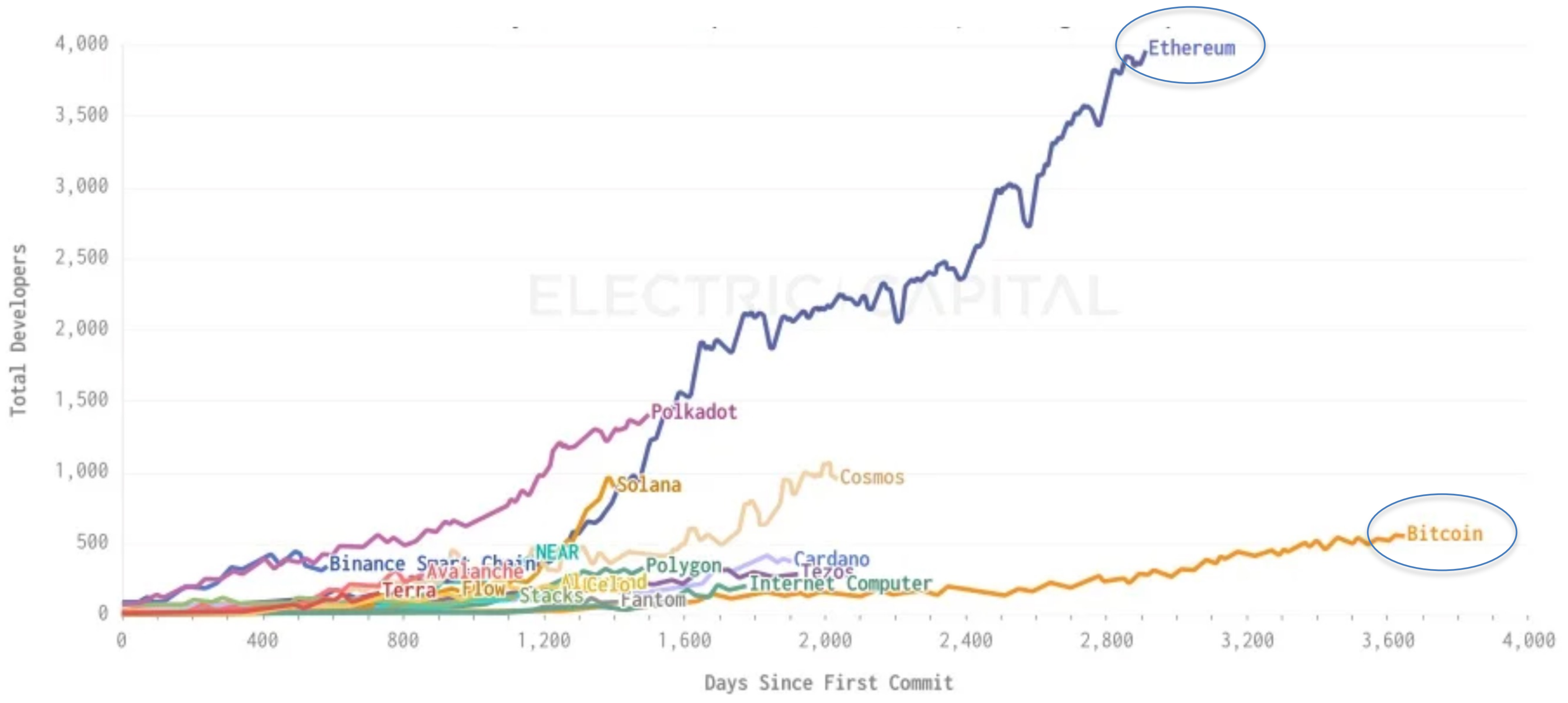
Transaction volume

Sep. 2022



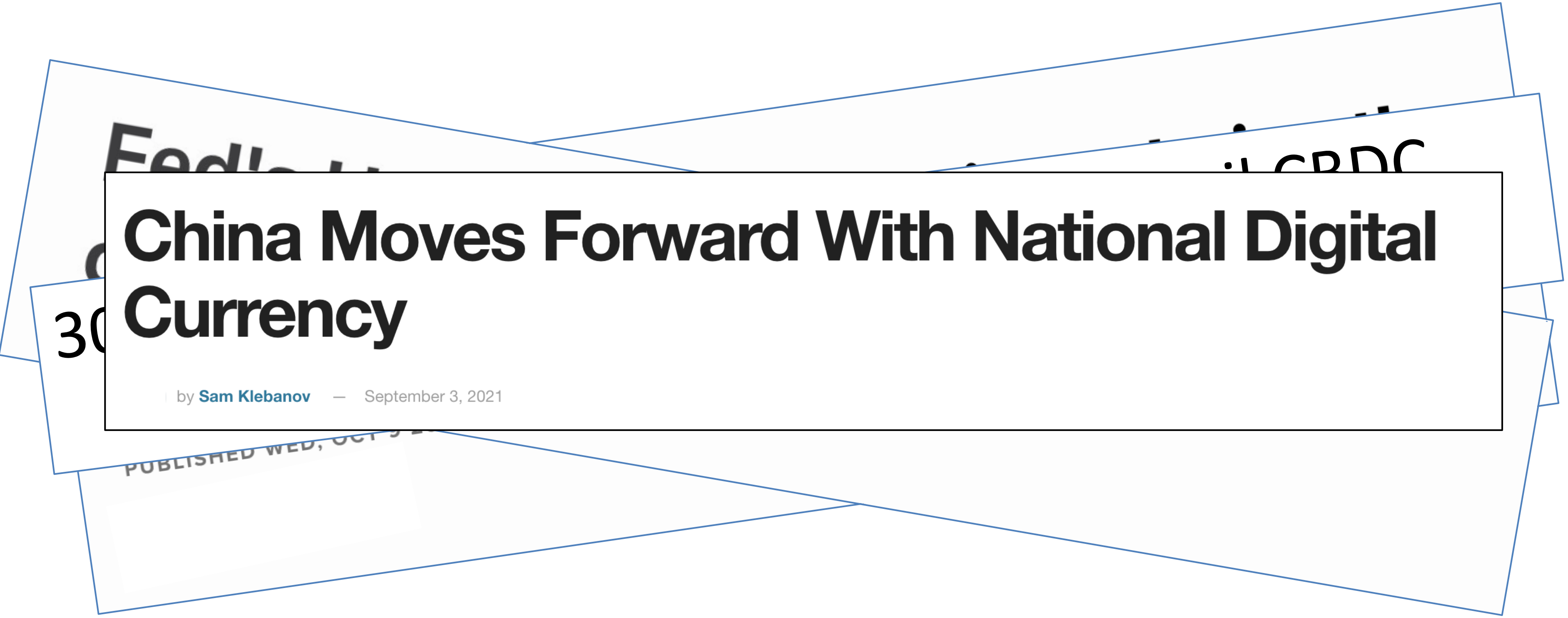
	24h volume
Bitcoin • BTC	$26.6B
Ethereum • ETH	$12.1B
Solana SOL	$0.88B

# Active developers since launch (as of 12/31/2021)



source: electric capital

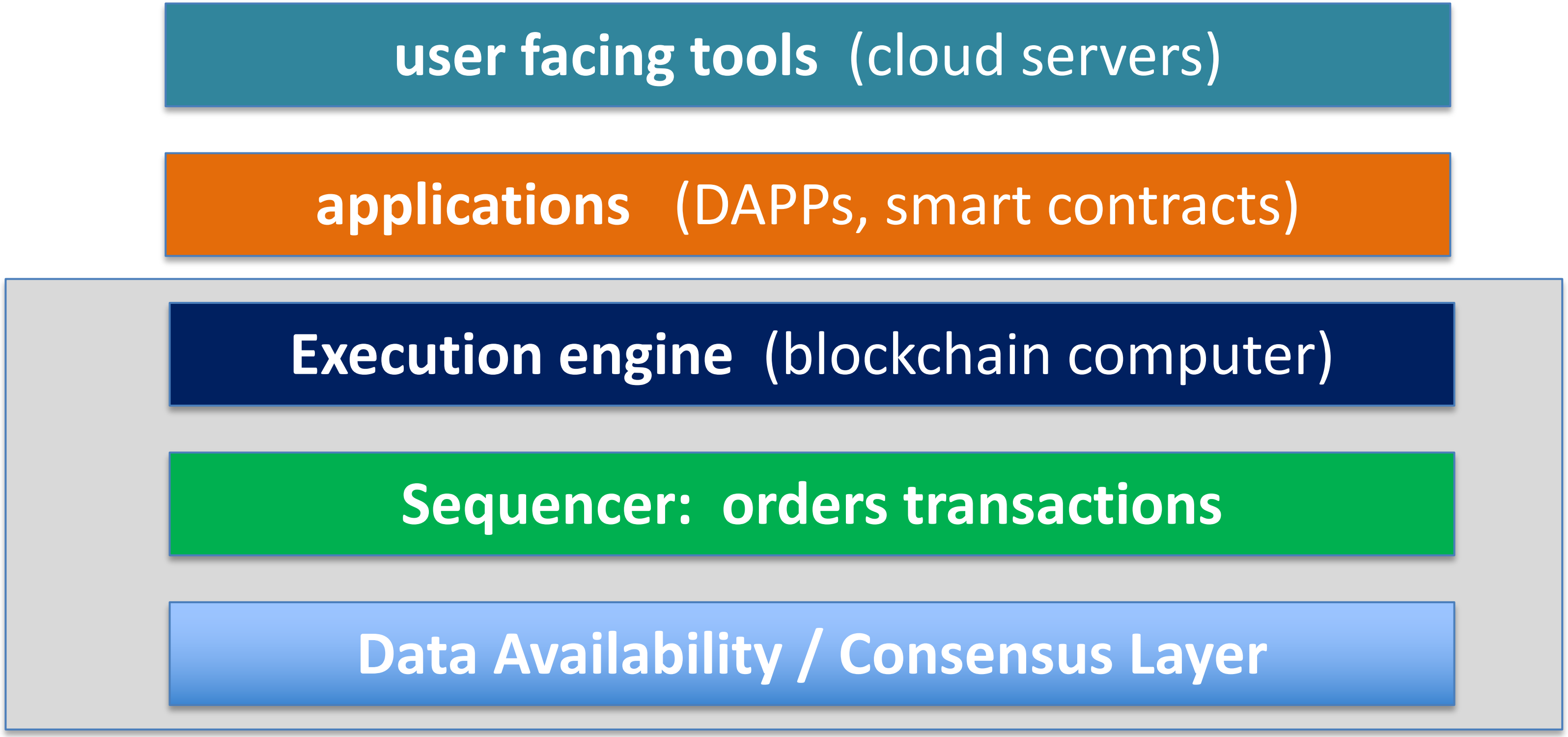
Central Bank Digital Currency (CBDC)



China Moves Forward With National Digital Currency

by Sam Klebanov — September 3, 2021

What is a blockchain?


user facing tools (cloud servers)
applications (DAPPs, smart contracts)
Execution engine (blockchain computer)
Sequencer: orders transactions
Data Availability / Consensus Layer

Consensus layer (informal)

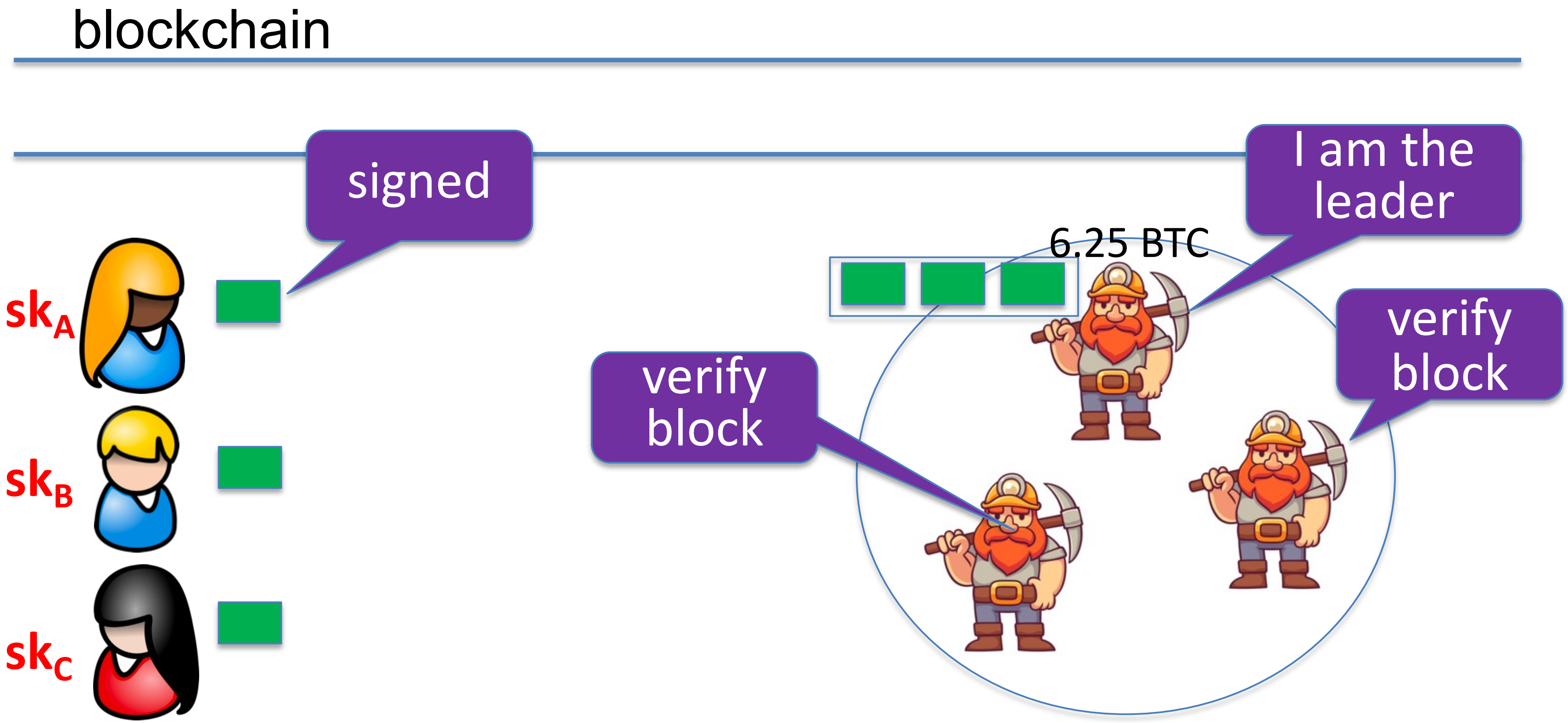
achieved by replication

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

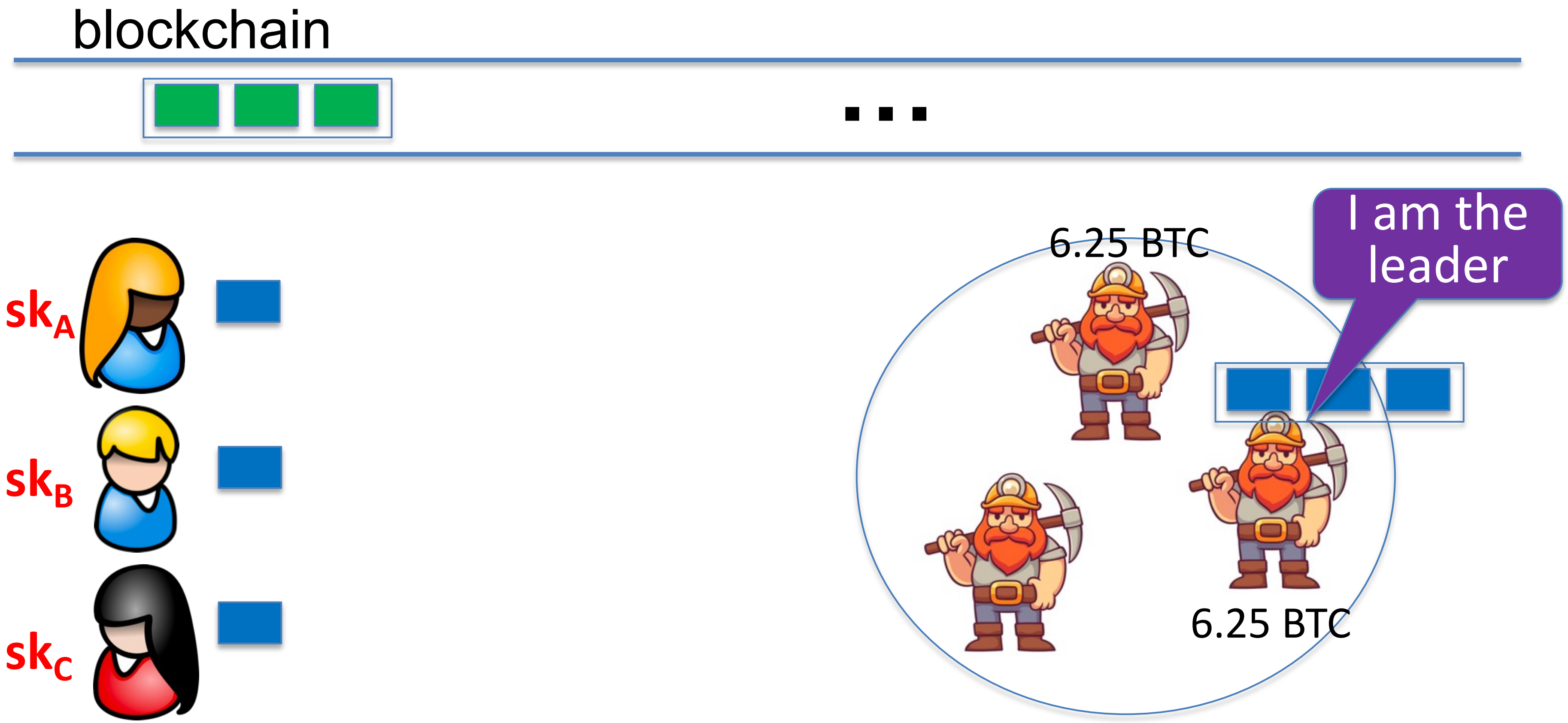
- **Persistence**: once added, data can never be removed*
- **Safety**: all honest participants have the same data**
- **Liveness:** honest participants can add new transactions
- **Open(?)**: anyone can add data (no authentication)

Data Availability / Consensus layer

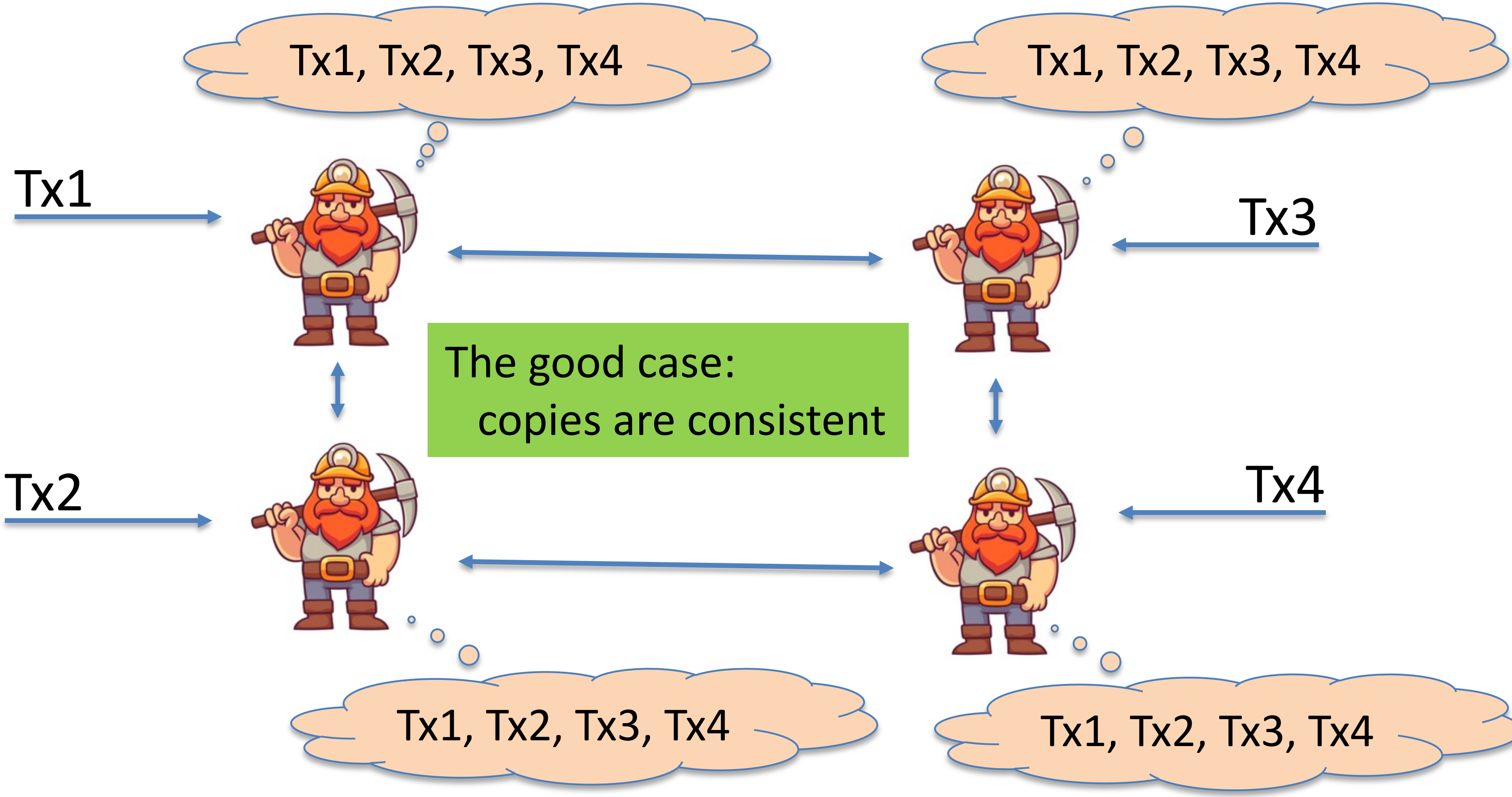
How are blocks added to chain?



How are blocks added to chain?



Why is consensus a hard problem?

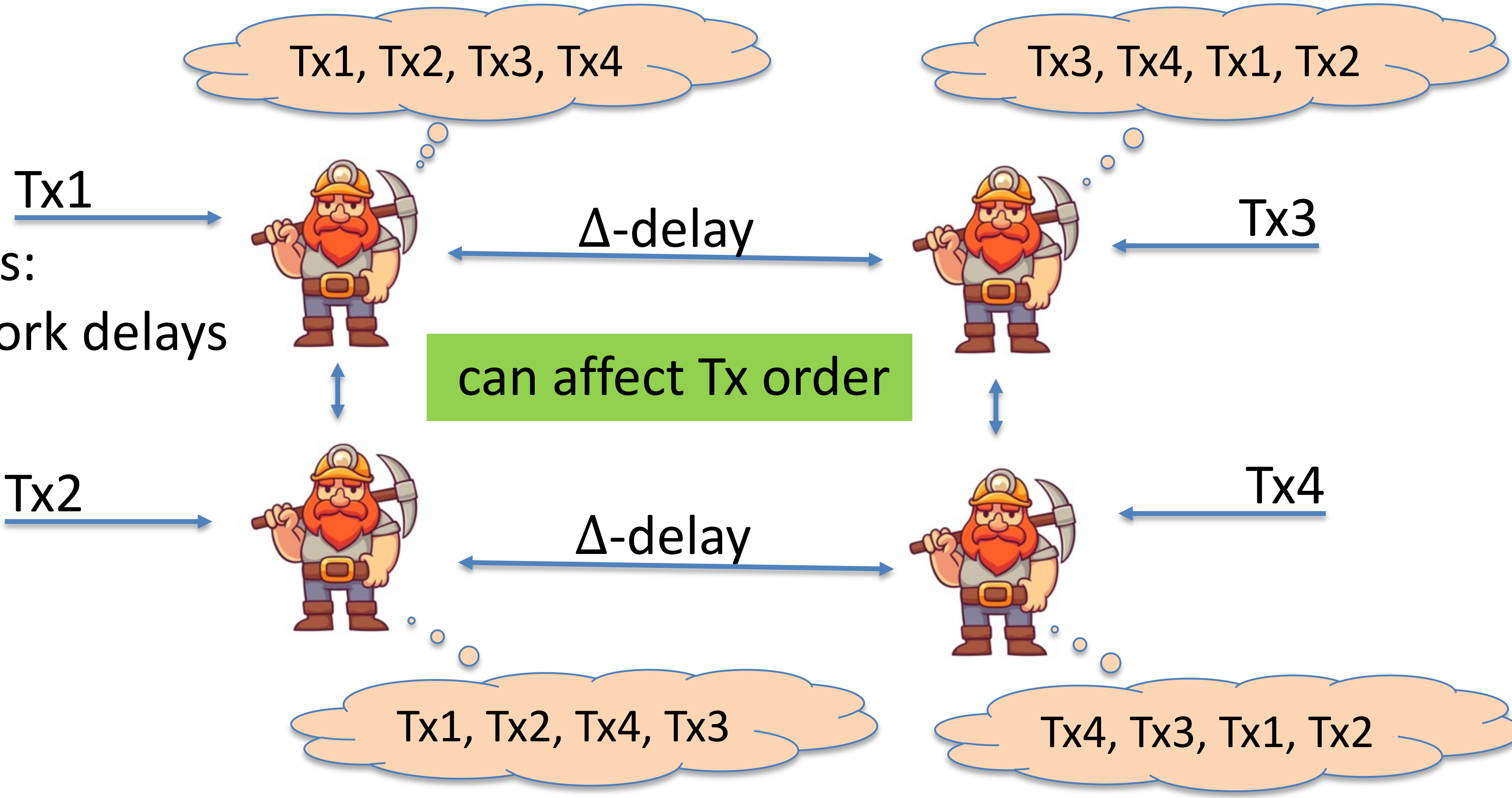


The good case:
copies are consistent

Why is consensus a hard problem?

Problems:

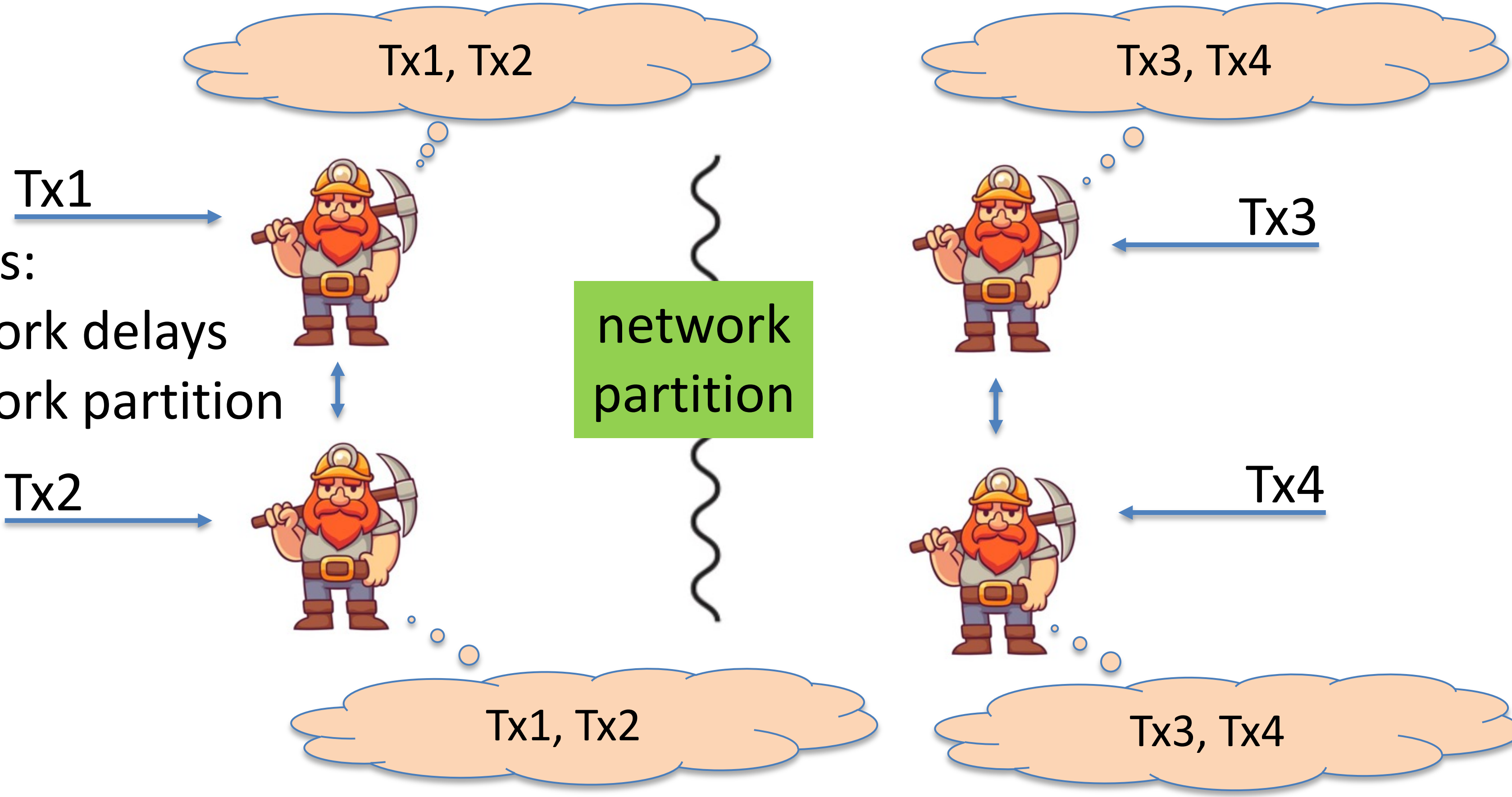
- Network delays



Why is consensus a hard problem?

Problems:

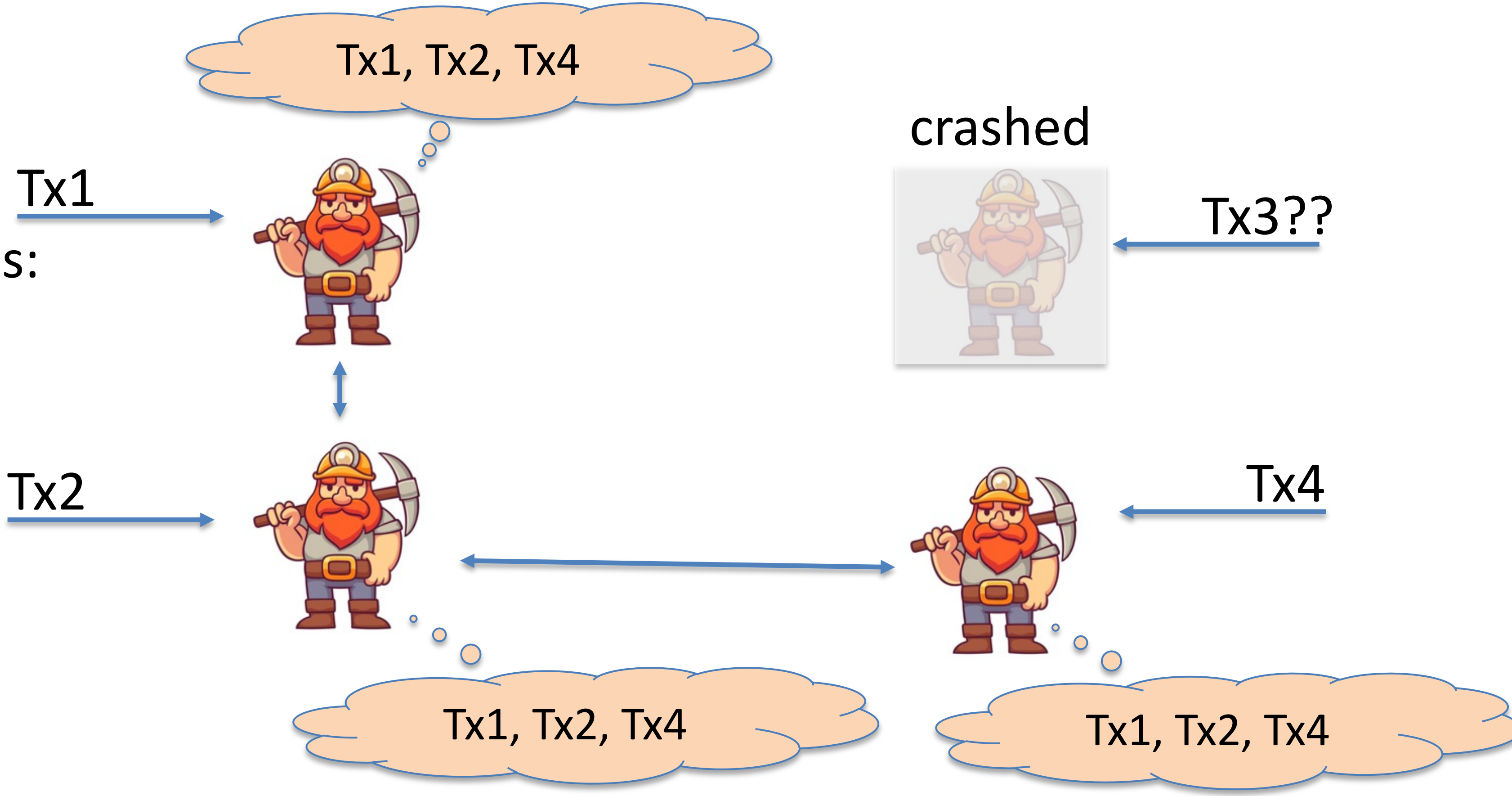
- Network delays
- Network partition



Why is consensus a hard problem?

Problems:

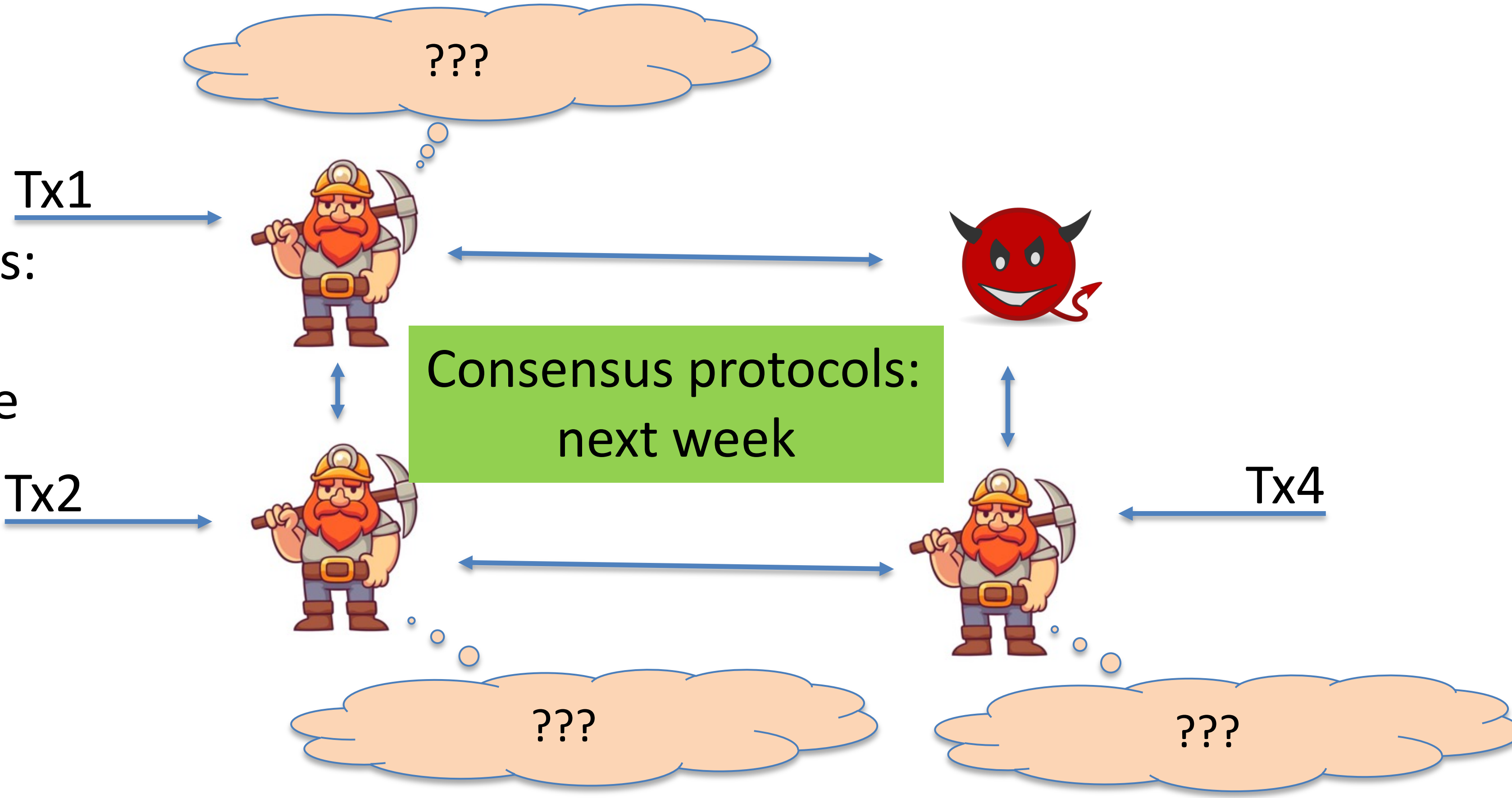
- crash



Why is consensus a hard problem?

Problems:

- crash
- malice

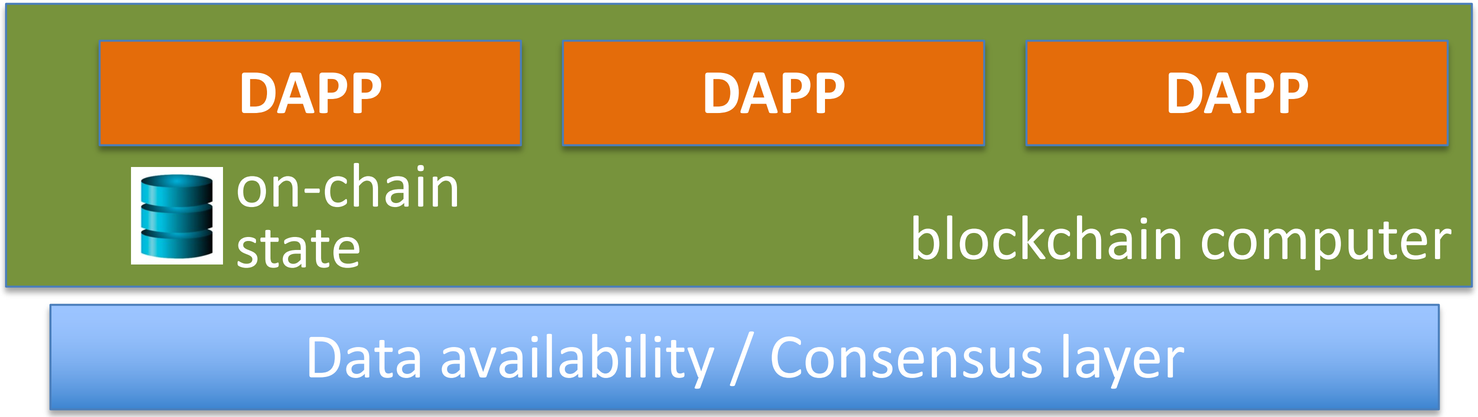


Consensus protocols: next week

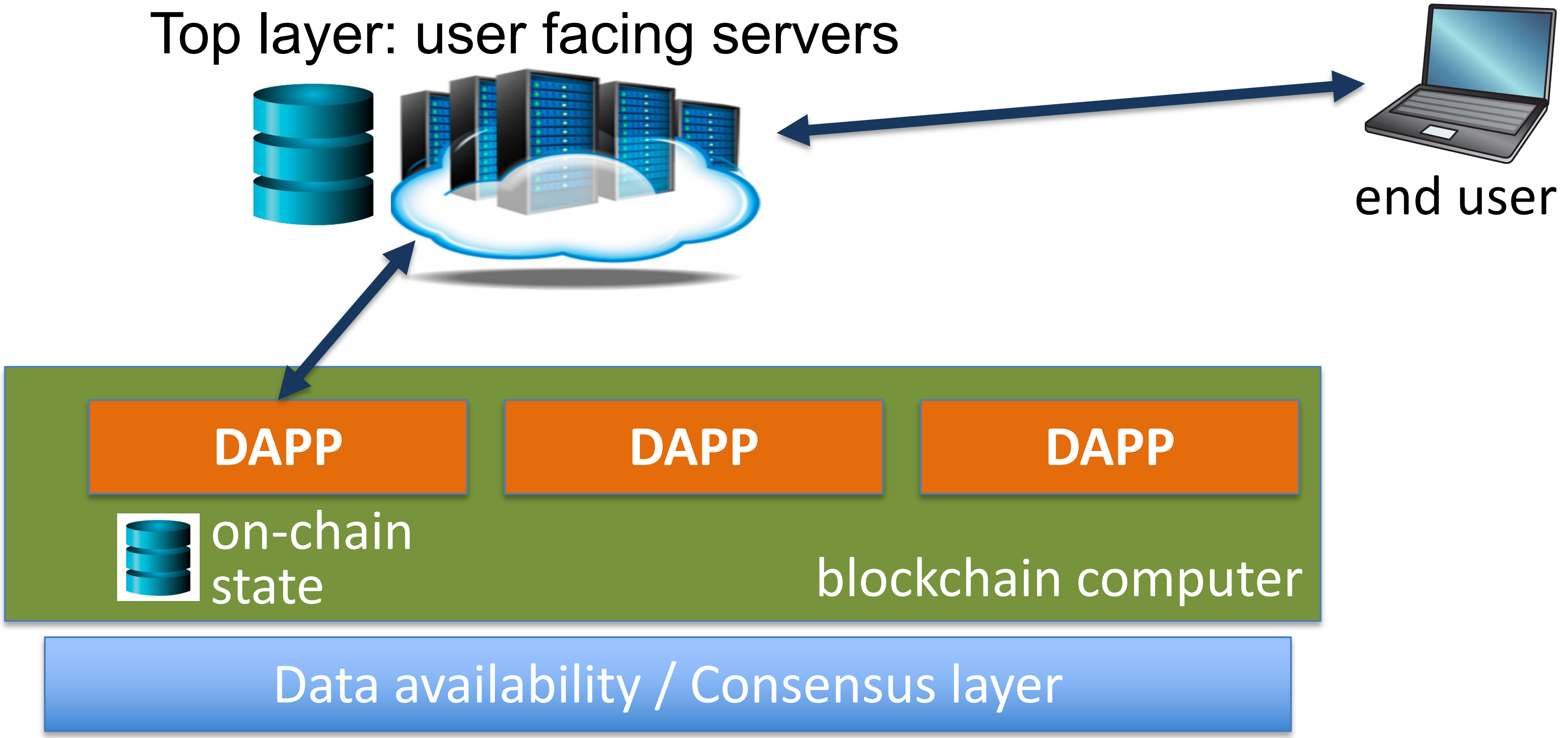
Next layer: the blockchain computer

Decentralized applications (DAPPs):

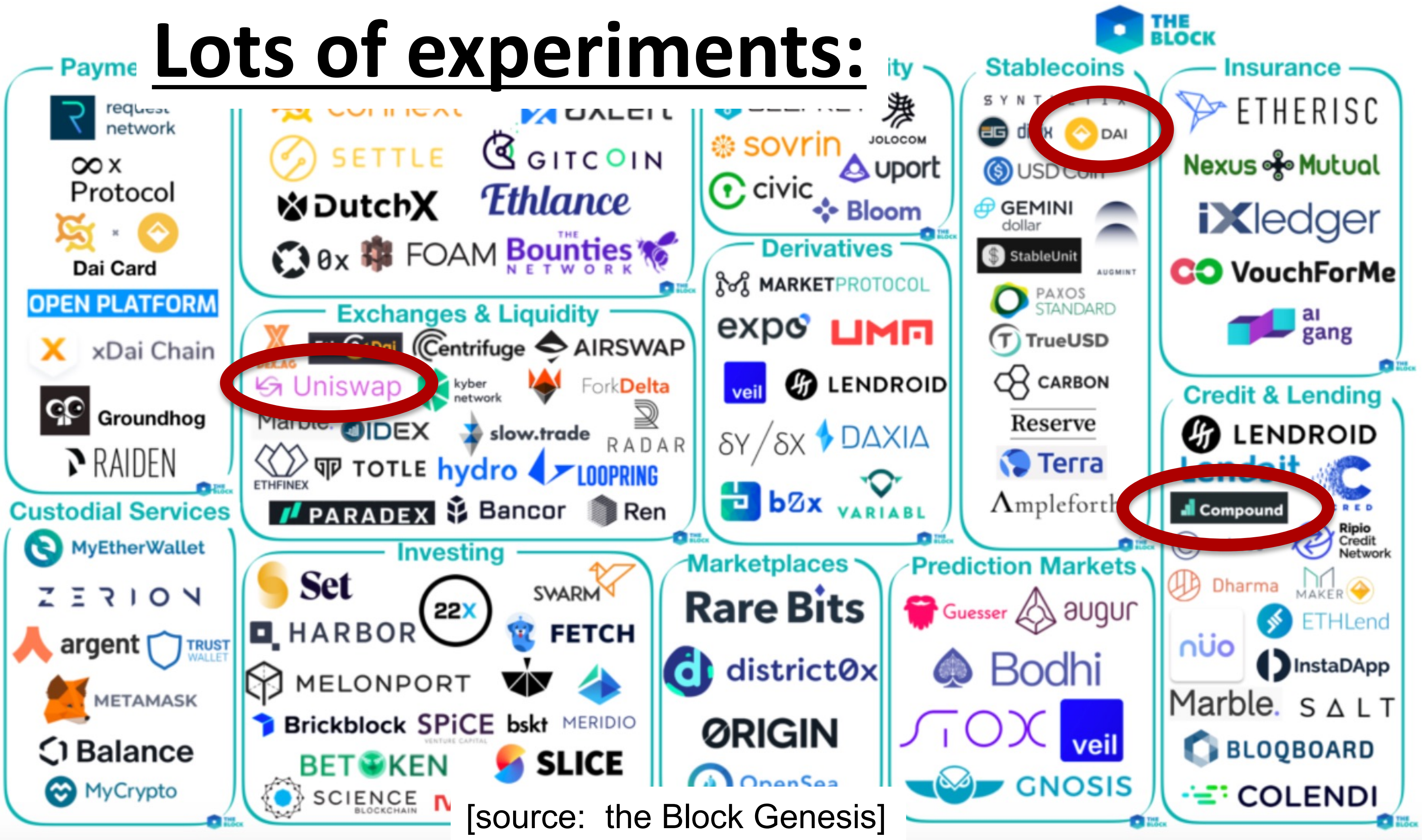
- Run on blockchain: code and state are written on chain
- Accept Tx from users $\Rightarrow$ state transitions are recorded on chain



Next layer: the blockchain computer

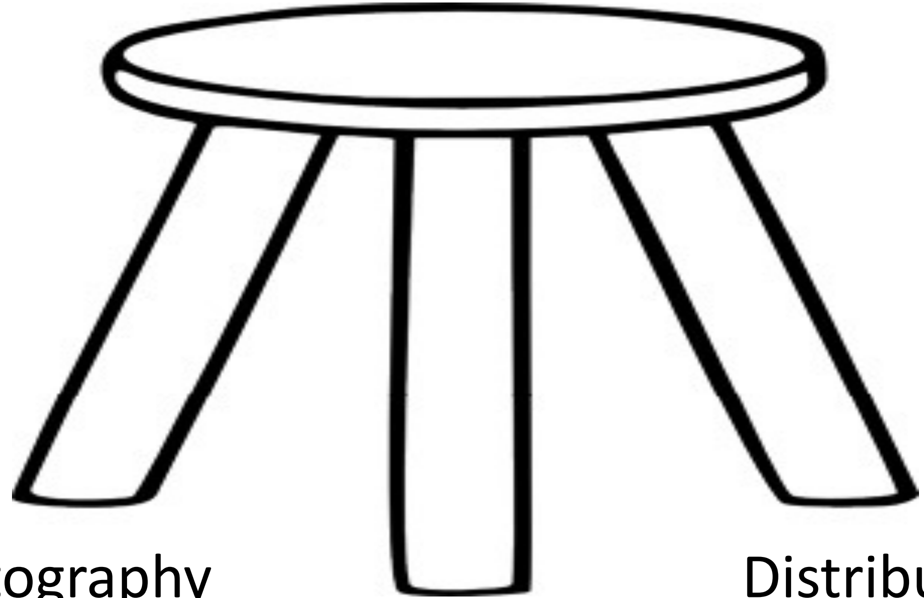

Top layer: user facing servers
end user
DAPP
DAPP
DAPP
on-chain state
blockchain computer
Data availability / Consensus layer

Lots of experiments:



[source: the Block Genesis]

This course



Cryptography

Distributed systems

Economics

Course organization

1. The starting point: Bitcoin mechanics
2. Consensus protocols
3. Ethereum and decentralized applications
4. DeFi: decentralized applications in finance
5. Private transactions on a public blockchain
   (SNARKs and zero knowledge proofs)
6. Scaling the blockchain: getting to 10K Tx/sec
7. Interoperability among chains: bridges and wrapped coins

Course organization

cs251.stanford.edu

- Homework problems, projects, final exam
- Optional weekly sections on Friday

Please tell us how we can improve …
Don’t wait until the end of the quarter

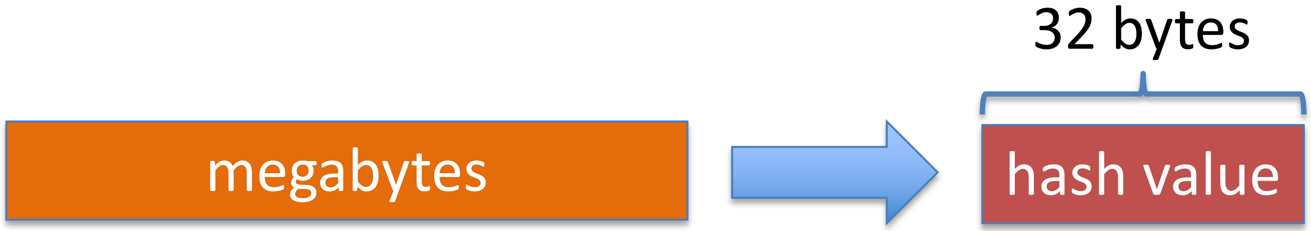
Let’s get started …

Cryptography Background

(1) cryptographic hash functions

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

where $|M| \gg |T|$



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

Collision resistance

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

$|M| \gg |T|$ 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 CRF)

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

(output is 32 bytes)

details in CS255

Application: committing to data on a blockchain

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

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

H is a CRF $\Rightarrow$ 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** to m

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

Committing to a list (of transactions)

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

32 bytes

Goal:

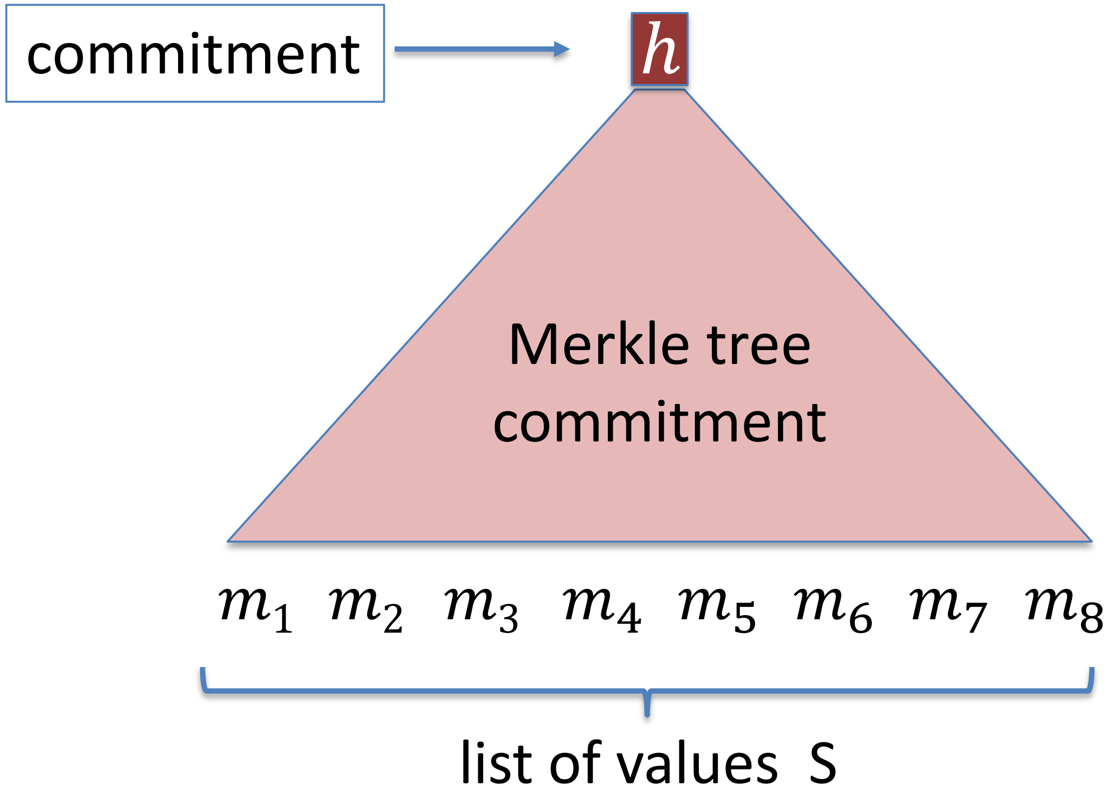
- Alice posts a <u>short</u> binding commitment to S, $h = \text{commit}(S)$
- Bob reads h. Given $(m_i,\ \text{proof}\ \pi_i)$ can check that $S[i] = m_i$

  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

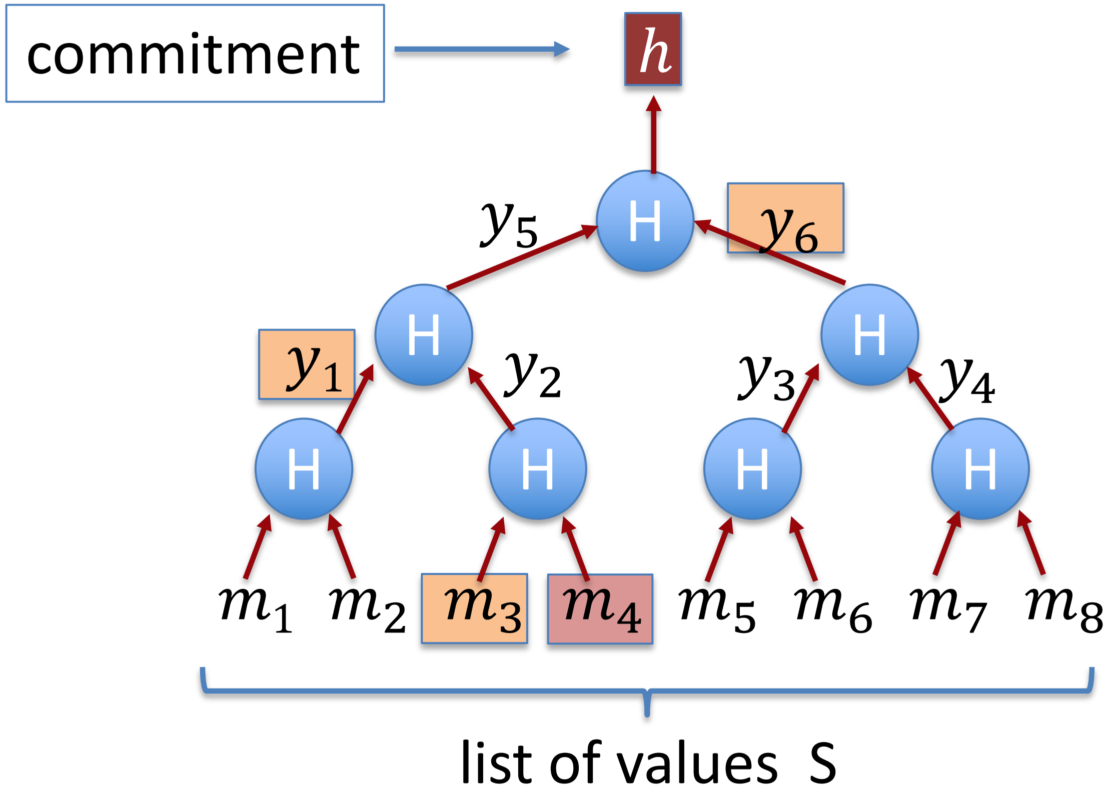
$\text{verify}(h, i, m, \pi) = \text{accept}$ where $h = \text{commit}(S)$

Merkle tree (Merkle 1989)



Goal:
- commit to list S of size n
- Later prove $S[i] = m_i$

Merkle tree (Merkle 1989) [simplified]



Goal:

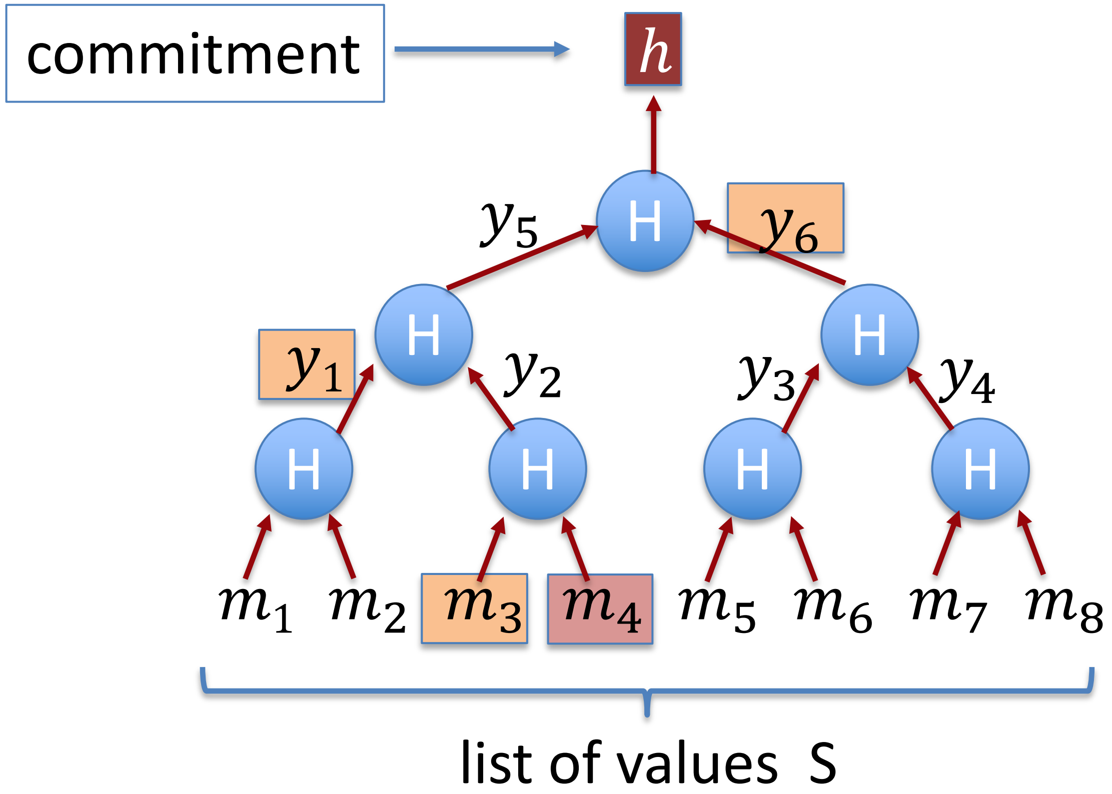
- commit to list S of size n
- Later prove $S[i] = m_i$

To prove $S[4] = m_4$,

proof π = (m_3, y_1, y_6)

length of proof: $\log_2 n$

Merkle tree (Merkle 1989) [simplified]



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: For a given n: if H is a CRF then

adv. cannot find (S, i, m, π) s.t. $|\mathrm{S}| = \mathrm{n}$, $m \neq S[i]$,

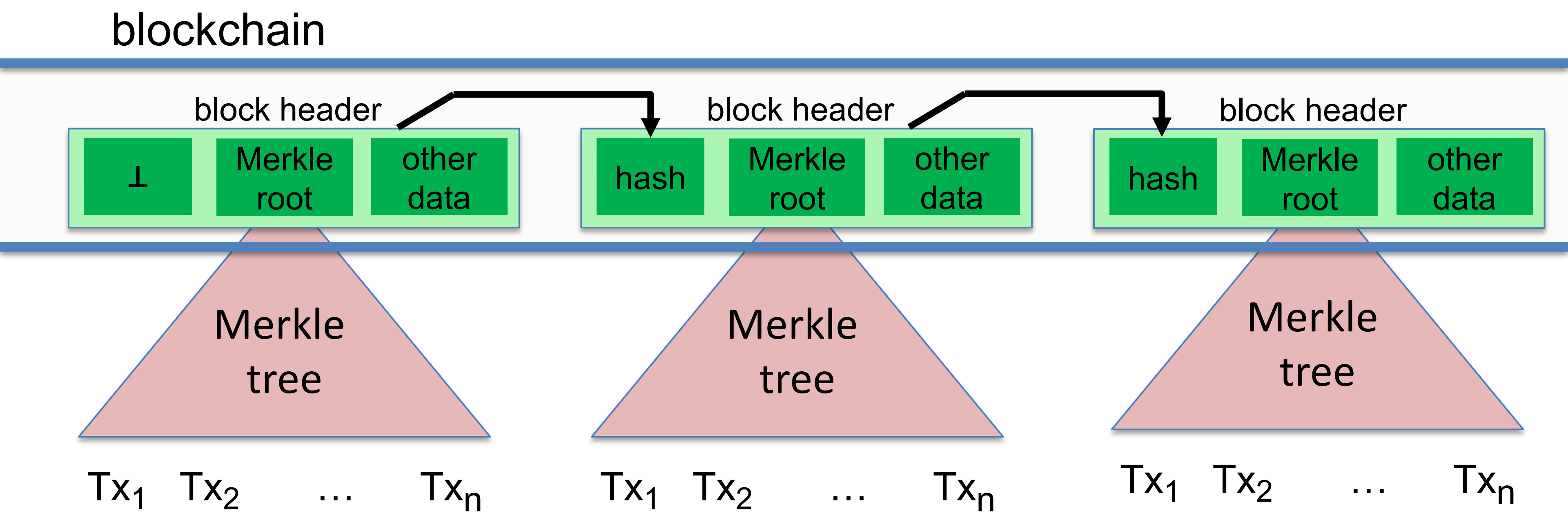
$h = \mathrm{commit}(S)$, and $\mathrm{verify}(h, i, m, \pi) = \mathrm{accept}$

(to prove, prove the contra-positive)

How is this useful? To post a block of transactions S on chain suffices to only write commit(S) to chain. Keeps chain small.

$\Rightarrow$ 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 $\boxed{H(x,y) < 2^n/D}$

Another application: proof of work

<u>Thm</u>: if H is a "random function" then the best algorithm requires D evaluations of H in expectation.

Note: this is a parallel algorithm

$\Rightarrow$ the more machines I have, the faster I solve the puzzle.

Proof of work is used in some consensus protocols (e.g., Bitcoin)

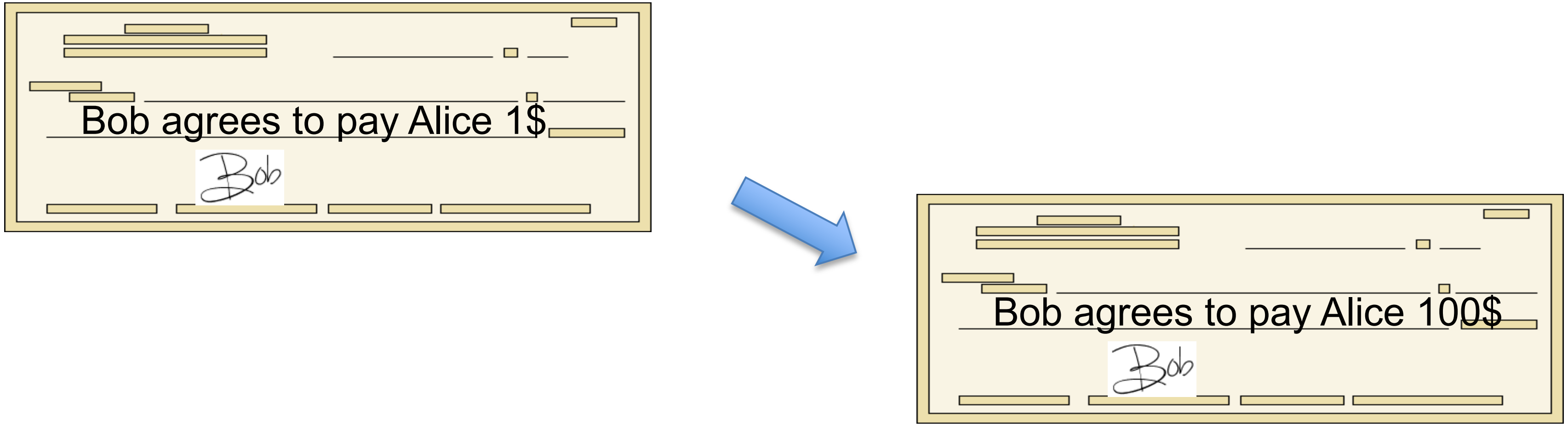
Bitcoin uses $H(x, y) = \mathrm{SHA256}(\mathrm{SHA256}(x.\,y))$

Cryptography background: Digital Signatures

How to authorize a transaction

Signatures

Physical signatures: bind transaction to author

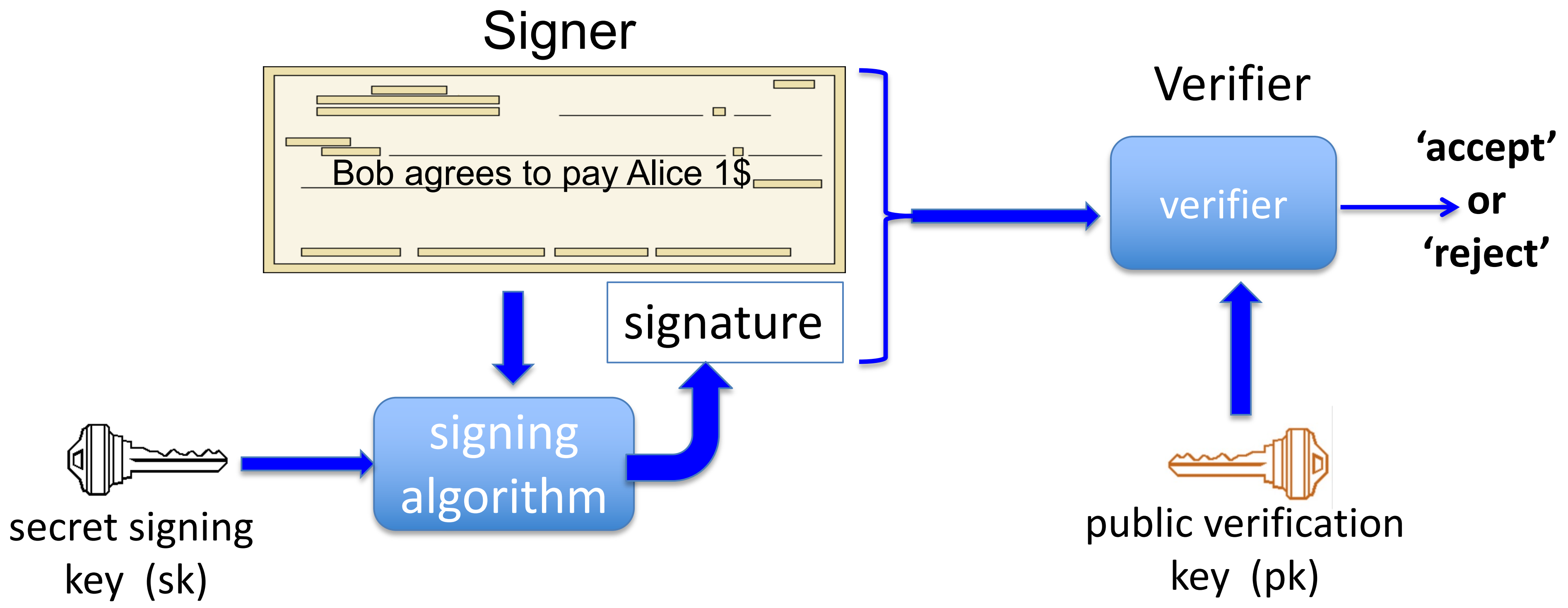


Problem in the digital world:

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

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

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

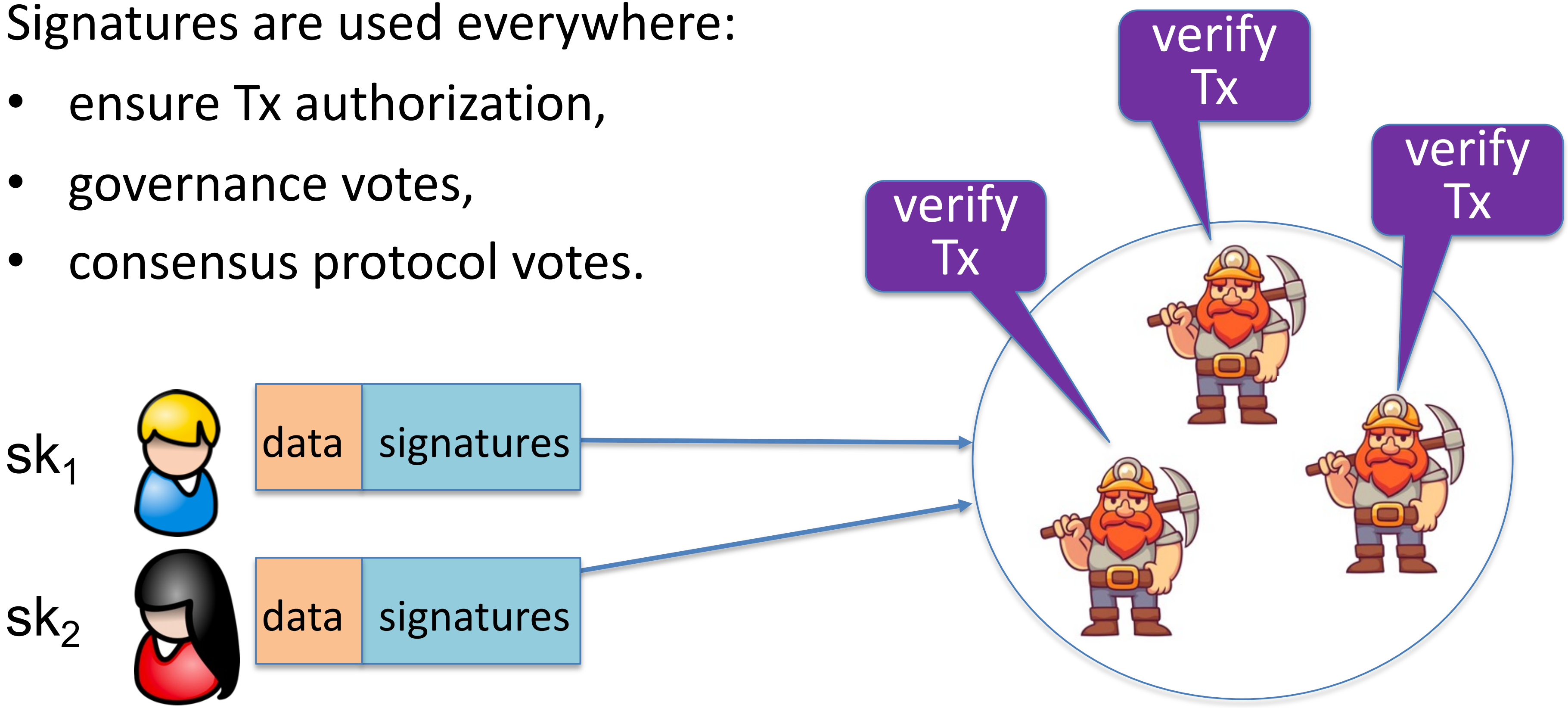
4. <u>Post-quantum</u> signatures: long (≥600 bytes)

details in CS255

Signatures on the blockchain

Signatures are used everywhere:

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



END OF LECTURE

Next lecture: the Bitcoin blockchain