Decentralized Finance

Introduction to Blockchain technology

Instructors: **Dan Boneh**, Arthur Gervais, Andrew Miller, Christine Parlour, Dawn Song





Stanford University
Imperial College London
University of Illinois Urbana-Champaign
Berkeley University of California

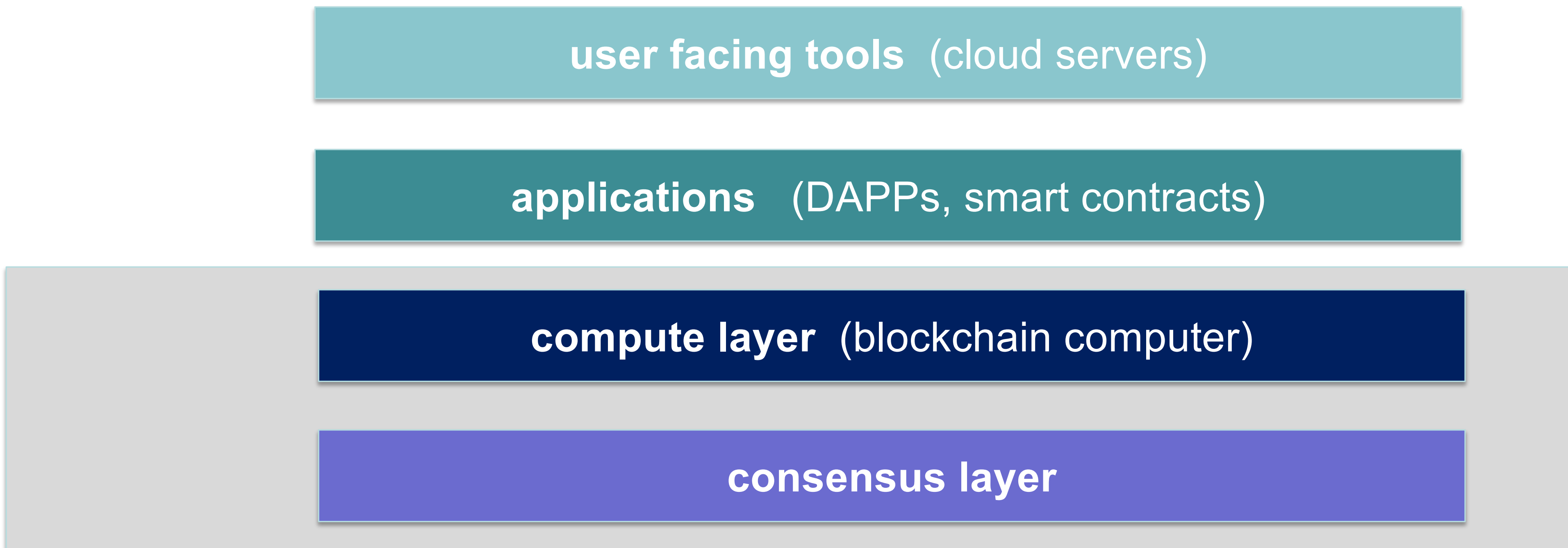
What is a blockchain?

Abstract answer: a blockchain provides
coordination between many parties,
when there is no single trusted party

if trusted party exists ⇒ no need for a blockchain

[financial systems: often no trusted party]

What is a blockchain?



Consensus layer (informal – not the topic of this course)

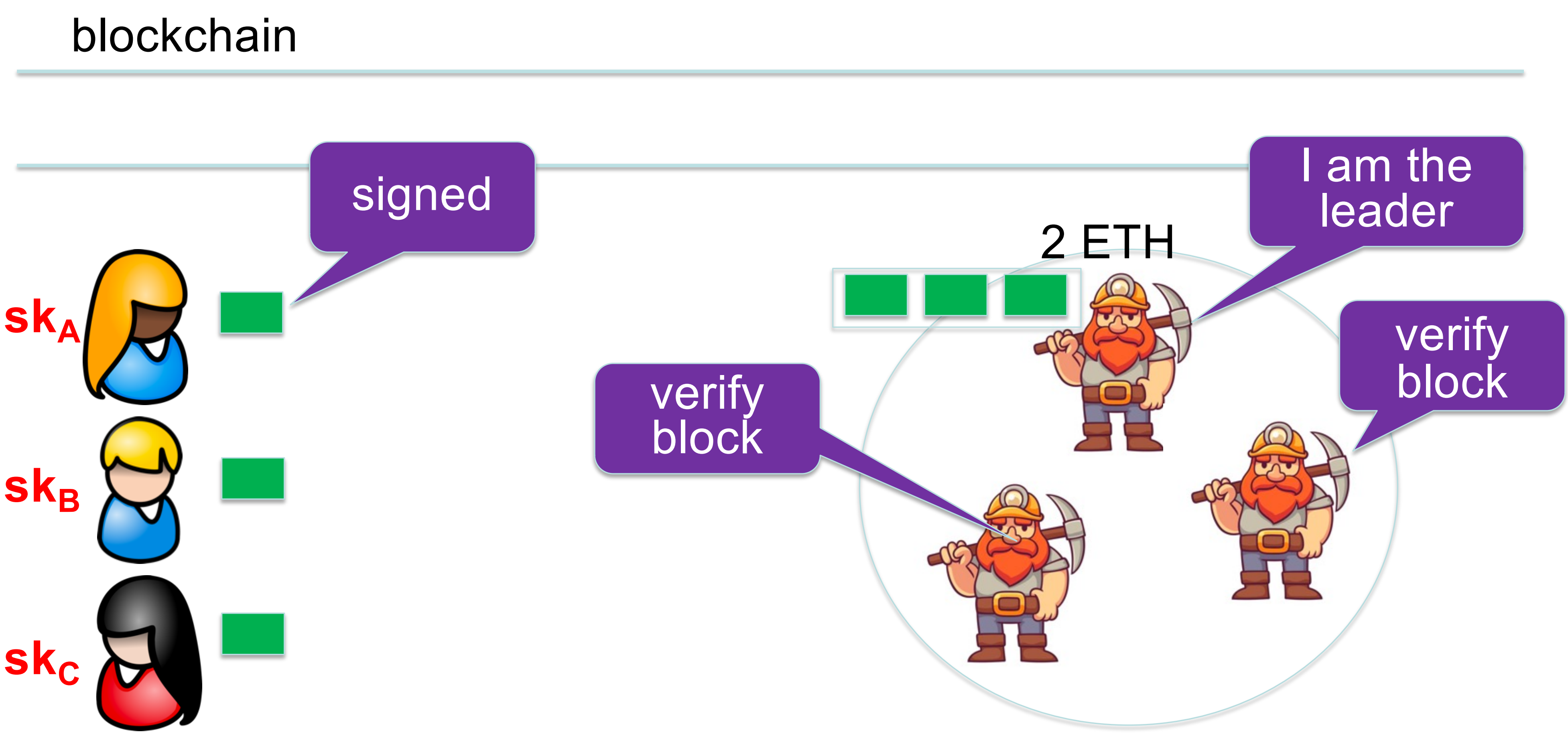
achieved by replication

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

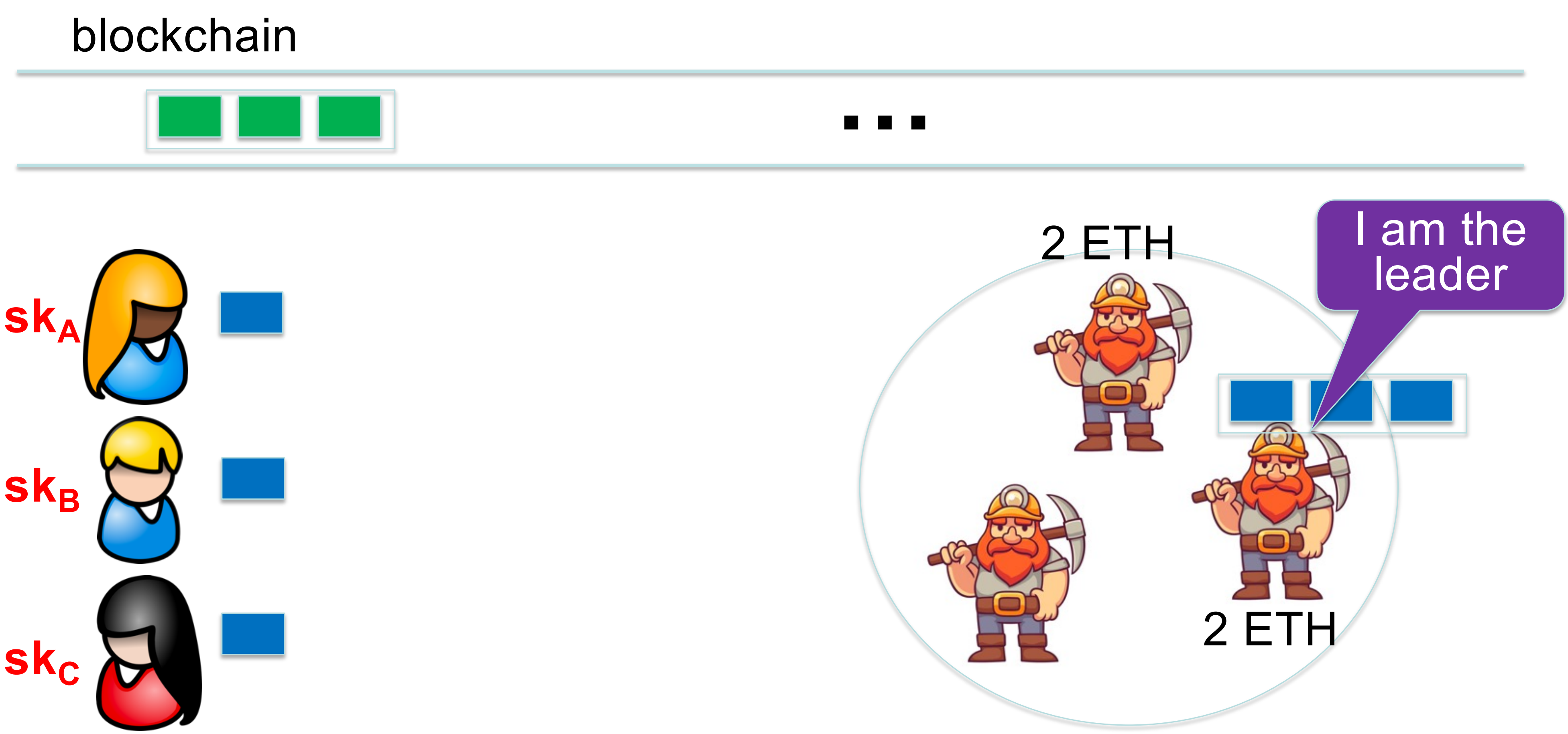
- **Persistence**: once added, data can never be removed*
- **Consensus**: all honest participants have the same data**
- **Liveness:** honest participants can add new transactions
- **Open(?)**: anyone can add data

Layer 1: consensus layer

How are blocks added to chain?



How are blocks added to chain?



Compute layer: The blockchain computer

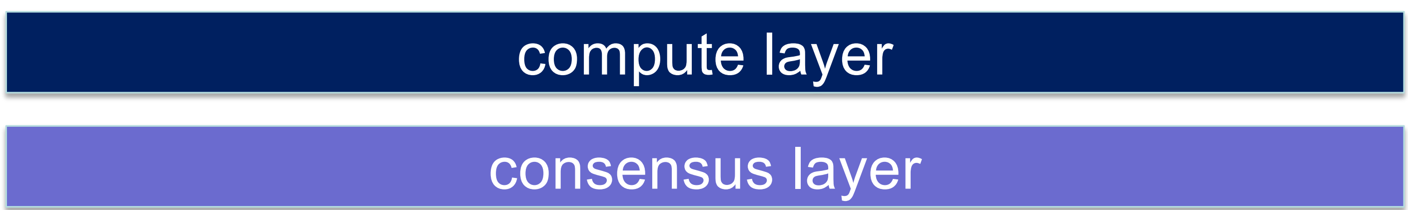
DAPP logic is encoded in a program that runs on blockchain

- Rules are enforced by a <u>public</u> program (public source code)

  ⇒ **<u>transparency</u>**: no single trusted 3rd party

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

  ⇒ **<u>public verifiability</u>**: everyone can verify state transitions



Apps layer: Decentralized applications (DAPPS)

Run on blockchain computer

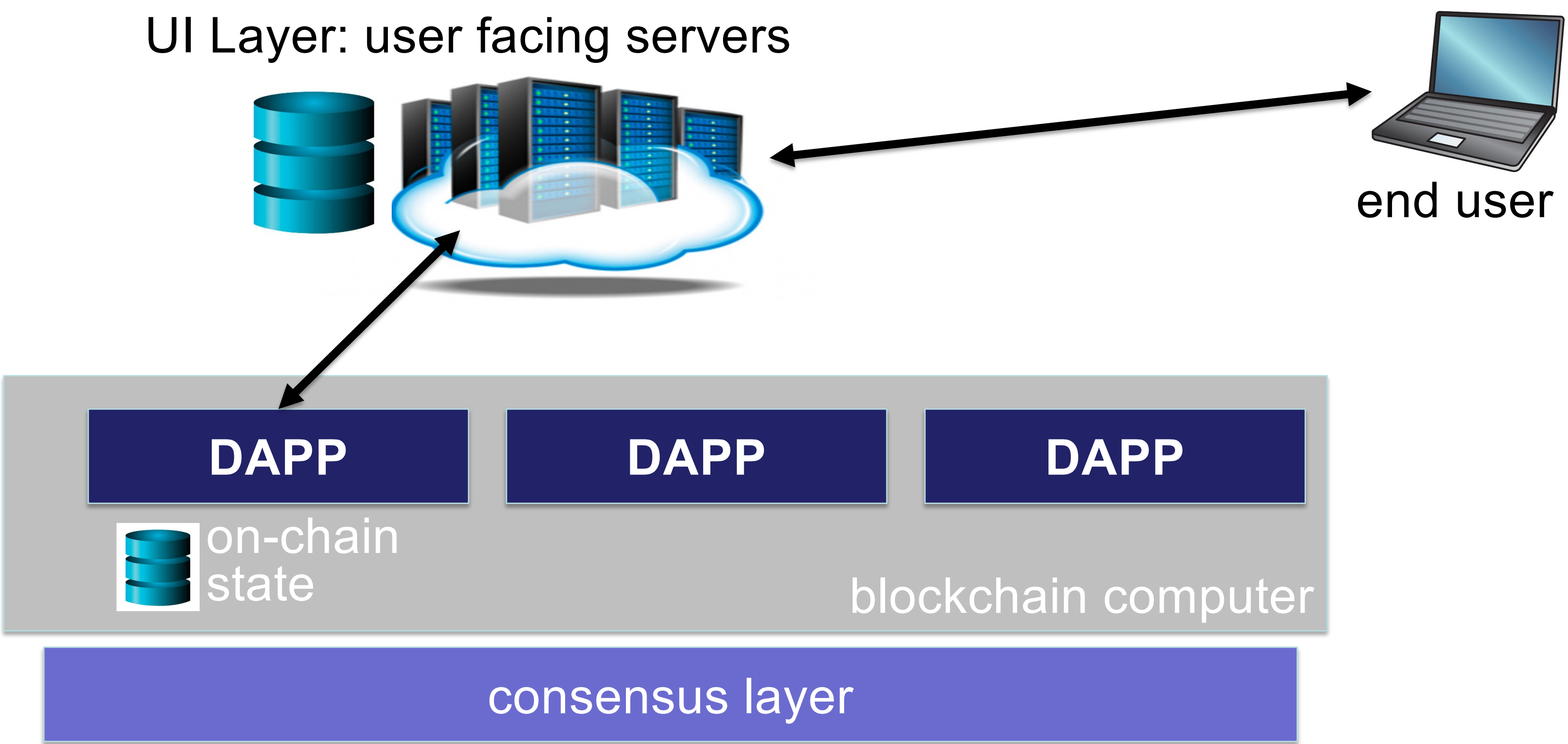
δY/δX

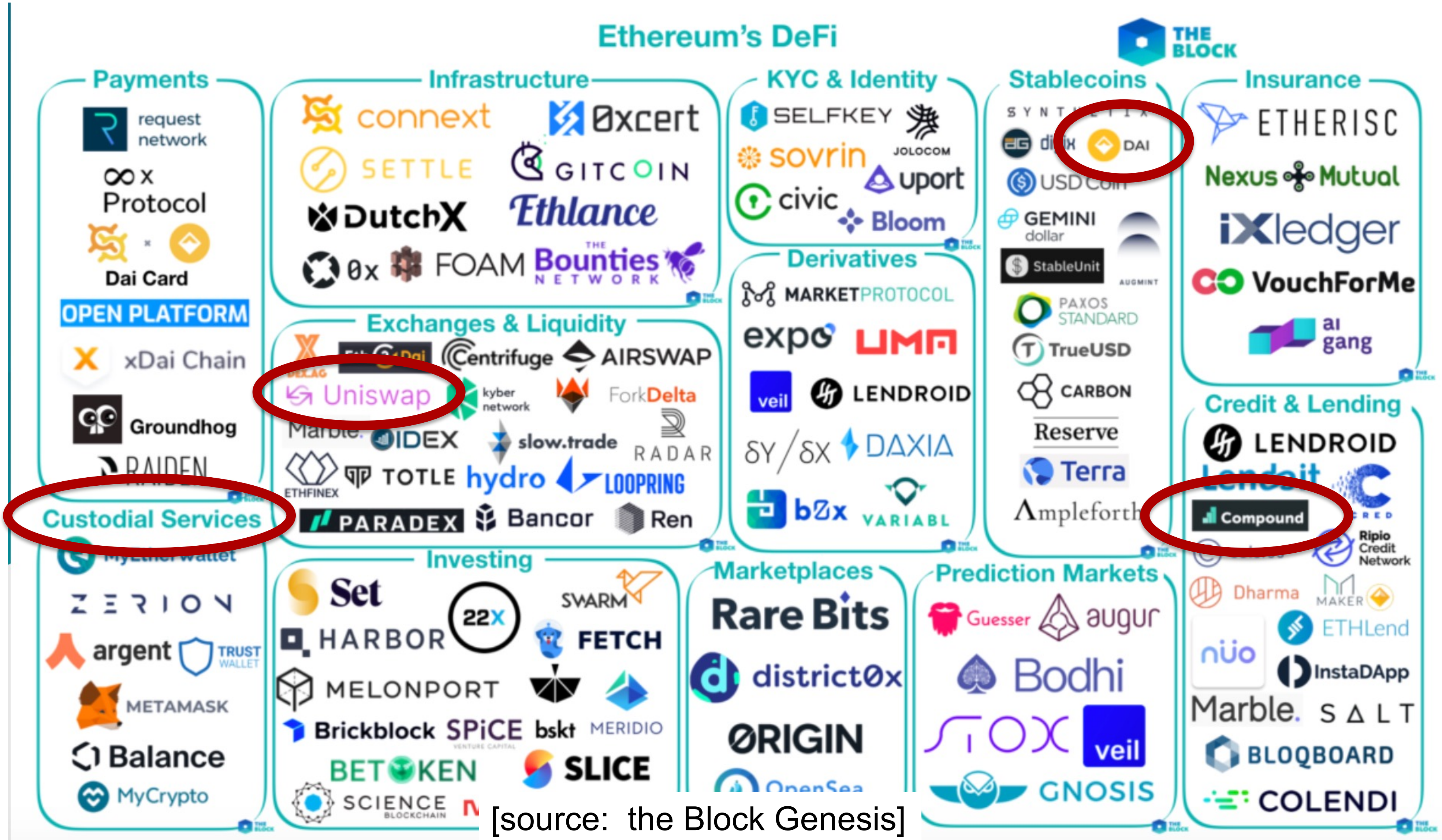
applications (DAPPs, smart contracts)

blockchain computer

consensus layer

UI Layer: Common DAPP architecture





[source: the Block Genesis]

lots of experiments …

DEFI PULSE	Name	Chain	Category	Locked (USD) ▼
🏆 1.	Aave	Multichain	Lending	$15.63B
🥈 2.	Curve Finance	Multichain	DEXes	$10.71B
🥉 3.	InstaDApp	Ethereum	Lending	$10.66B
4.	Compound	Ethereum	Lending	$10.43B
5.	Maker	Ethereum	Lending	$9.11B
6.	Uniswap	Ethereum	DEXes	$7.29B
7.	Convex Finance	Ethereum	Assets	$5.59B

Let's get started …

Next segment: cryptographic background

See you there

Cryptographic Background:

hash functions

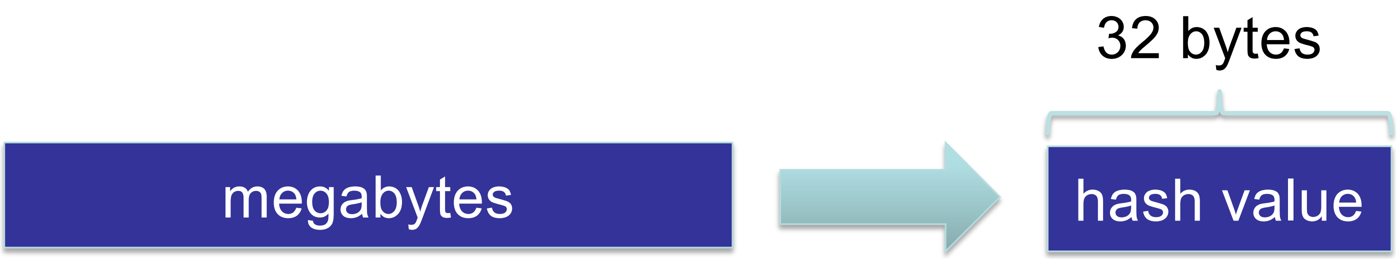
https://defi-learning.org/

Cryptography Background

(1) cryptographic hash functions

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

where $|M| \gg |T|$



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

Collision resistance

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

$|M| \gg |T|$ implies that many collisions exist

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

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

details in crypto MOOC

An application: committing to data

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

Bob has h. Later Alice sends m' s.t. $H(m') = h$

H is a CRH $\Rightarrow$ 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 $S = (m_1, m_2, \ldots, m_n)$

32 bytes

<u>Goal</u>:

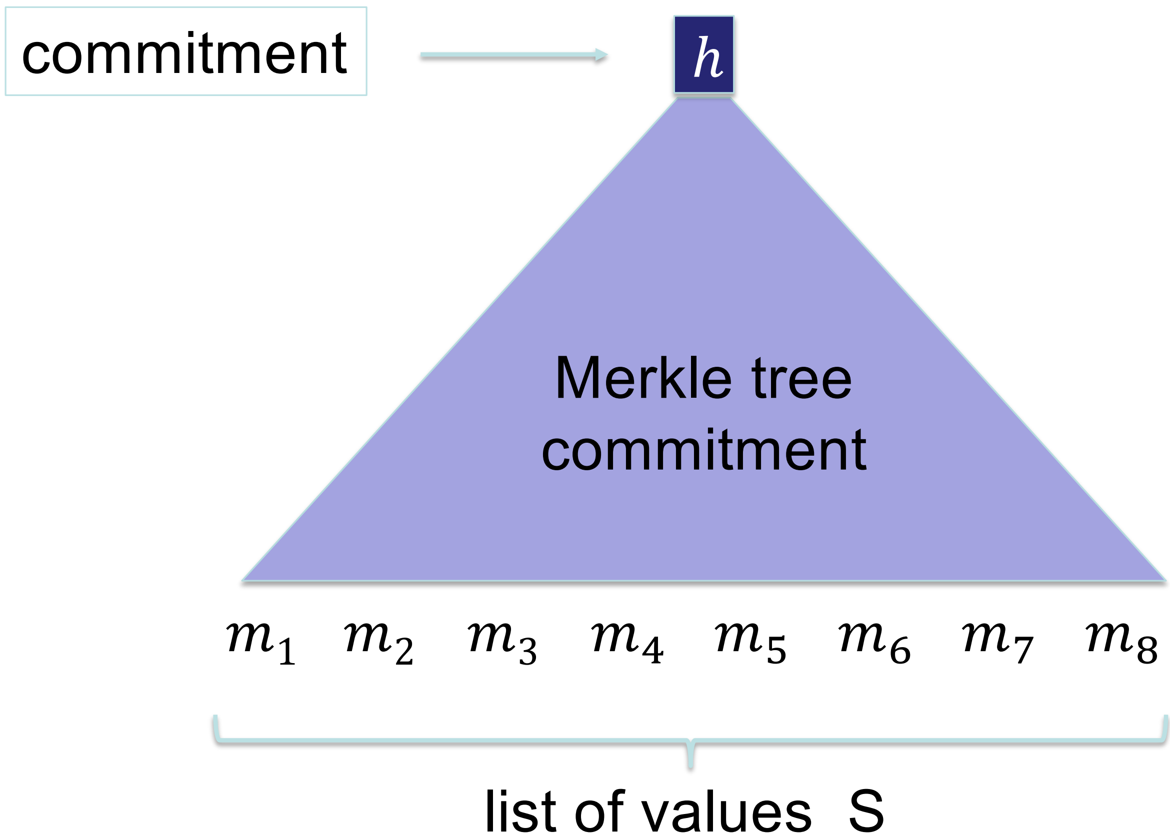
- Alice publishes a <u>short</u> binding commitment to S, $h = \text{commit}(S)$
- Bob has h. Given $(m_i,\ \text{proof}\ \pi_i)$ can check that $S[i] = m_i$

  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 $h = \text{commit}(S)$

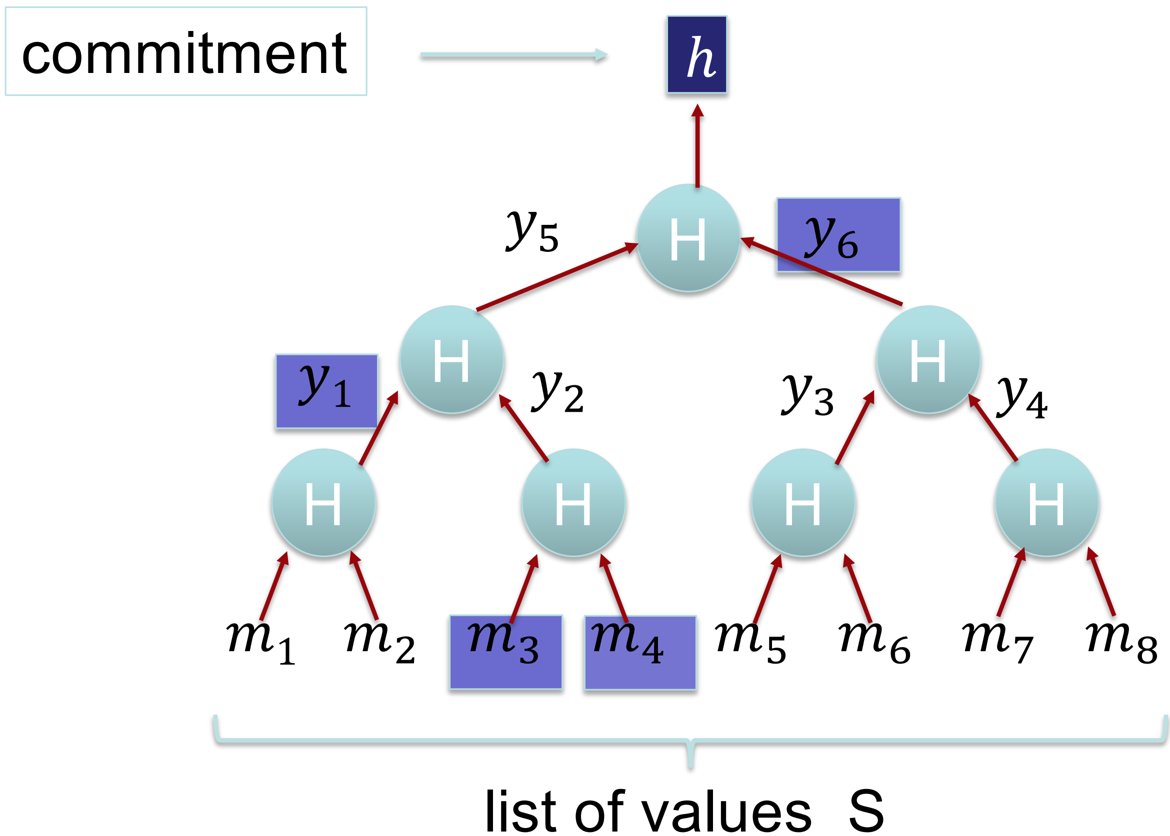
Merkle tree (Merkle 1989)



Goal:

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

Merkle tree (Merkle 1989)



Goal:

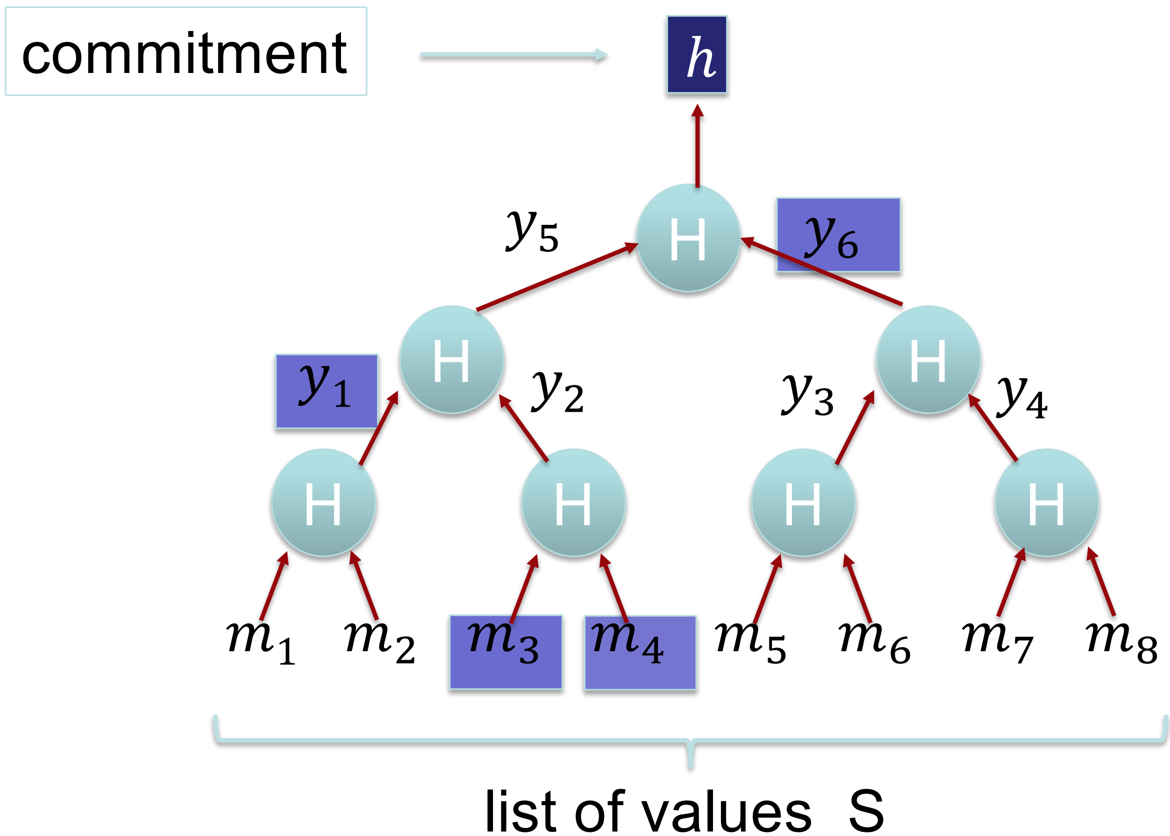
- commit to list S of size n
- Later prove $S[i] = m_i$

To prove $S[4] = m_4$

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

length of π: $\log_2 n$

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

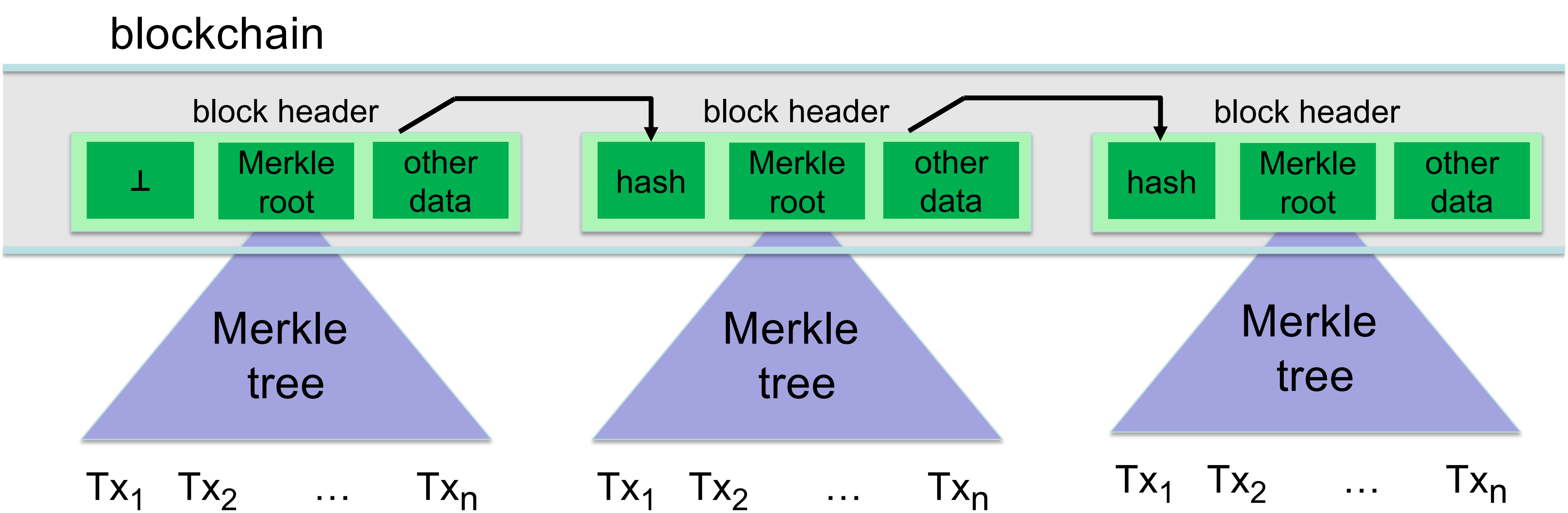
$$\text{verify}(h, i, m, \pi) = \text{accept} \quad \text{where} \quad h = \text{commit}(S)$$

(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”

Next segment: digital signatures

How to authorize transactions??

Cryptographic Background:

Digital Signatures

https://defi-learning.org/

Digital Signatures

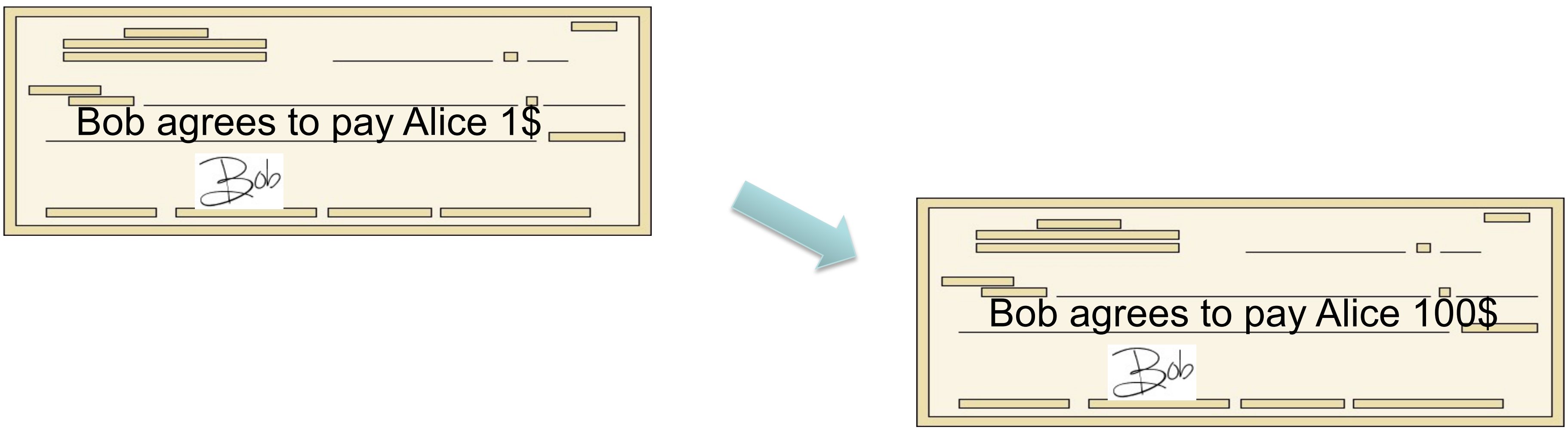
- In the last segment we looked at cryptographic hash functions.

- In this segment we will look at digital signatures:

how to approve a transaction?

Signatures

Physical signatures: bind transaction to author

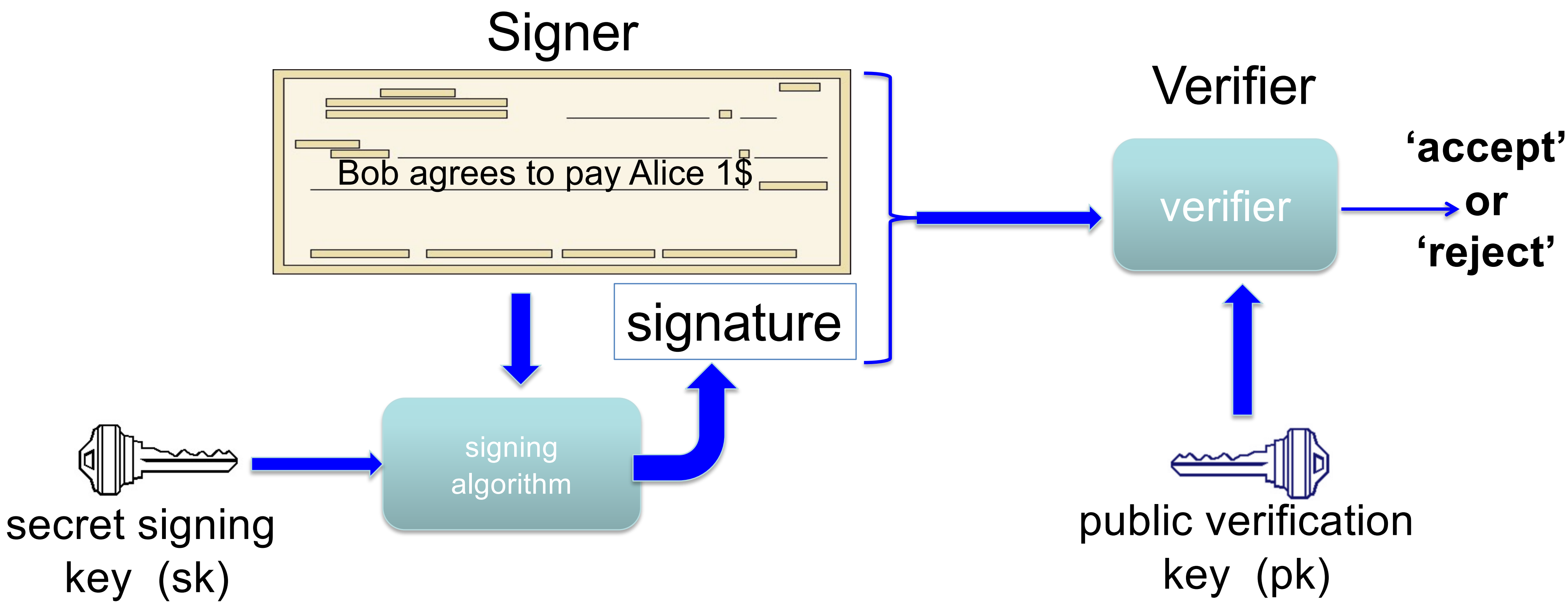


Problem in the digital world:

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 pk and sigs on many messages of her choice, cannot forge a signature on a new message.

Families of signature schemes

1. RSA signatures (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 keys (32 bytes)

3. BLS signatures: 48 bytes, aggregatable, easy threshold

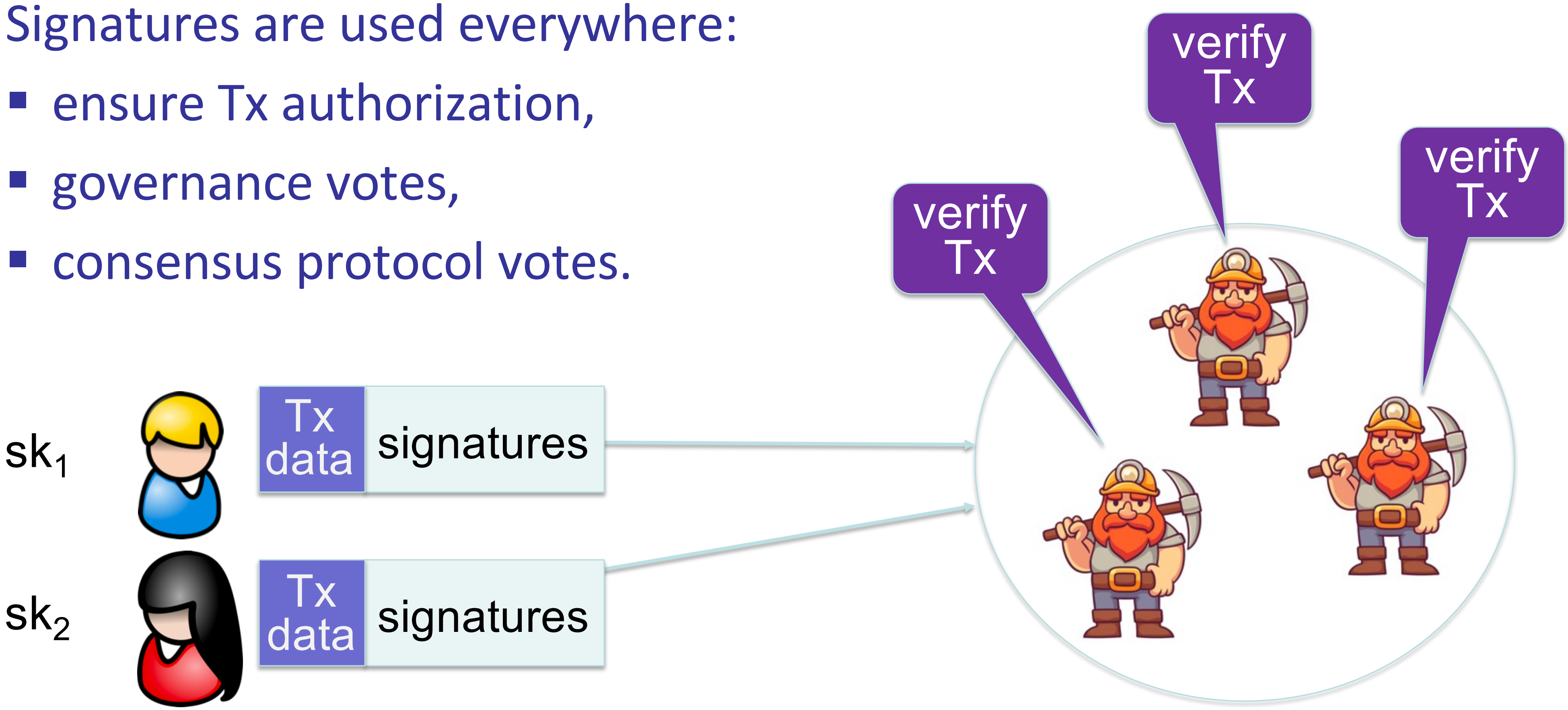
   (Ethereum 2.0, Chia, Dfinity)

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

Signatures on the blockchain

Signatures are used everywhere:

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



SNARK proofs

We covered two important cryptographic primitives:

1. Collision resistant hash functions and Merkle trees,
2. Digital signatures.

Another important cryptographic primitive is a **SNARK proof**:

- Used for scaling and privacy
- We will discuss SNARKs in detail in the lecture on privacy

Next segment: scaling the blockchains

Can we make it fast??

Scaling Blockchains

https://defi-learning.org/

Scaling

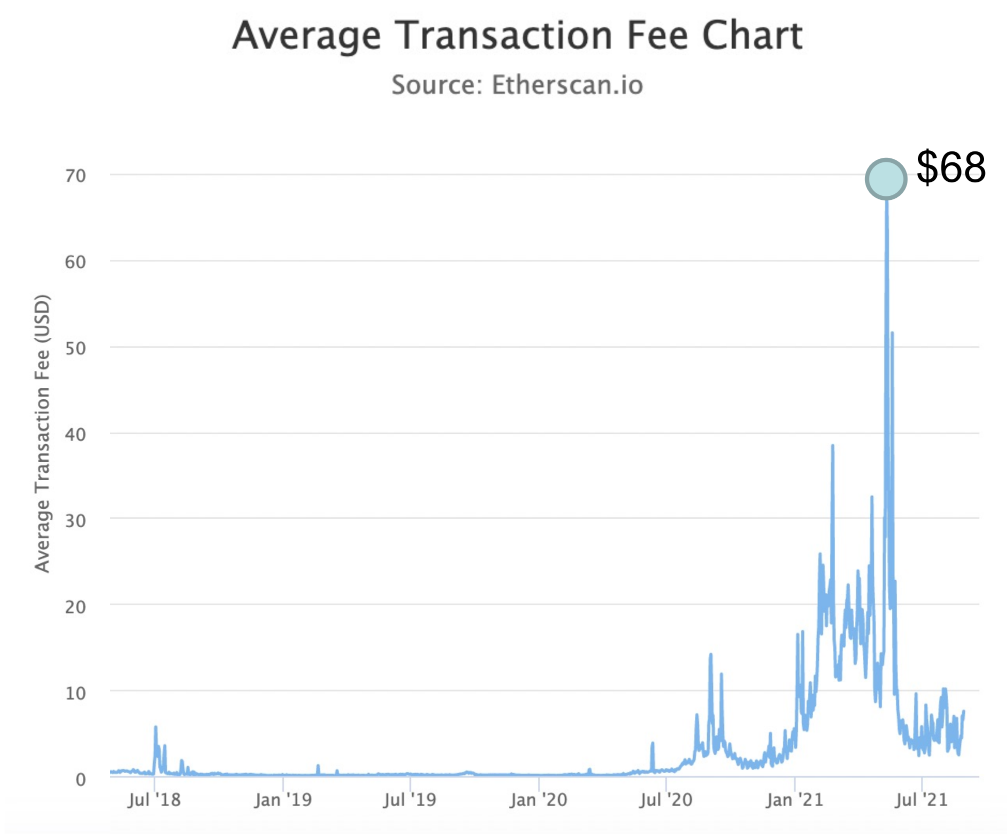
Transaction rates (Tx/sec):

- Bitcoin: can process about **5 (Tx/sec)**
- Ethereum: can process about **20 (Tx/sec)**

Tx Fees fluctuate:

2$ to 60$ for simple Tx

Ethereum Tx fees (gas prices)



Scaling

Transaction rates (Tx/sec):

- Bitcoin: can process about **5 (Tx/sec)**
- Ethereum: can process about **20 (Tx/sec)**

Tx Fees fluctuate:

2$ to 60$ for simple Tx

- The visa network: can process up to **24,000 (Tx/sec)**

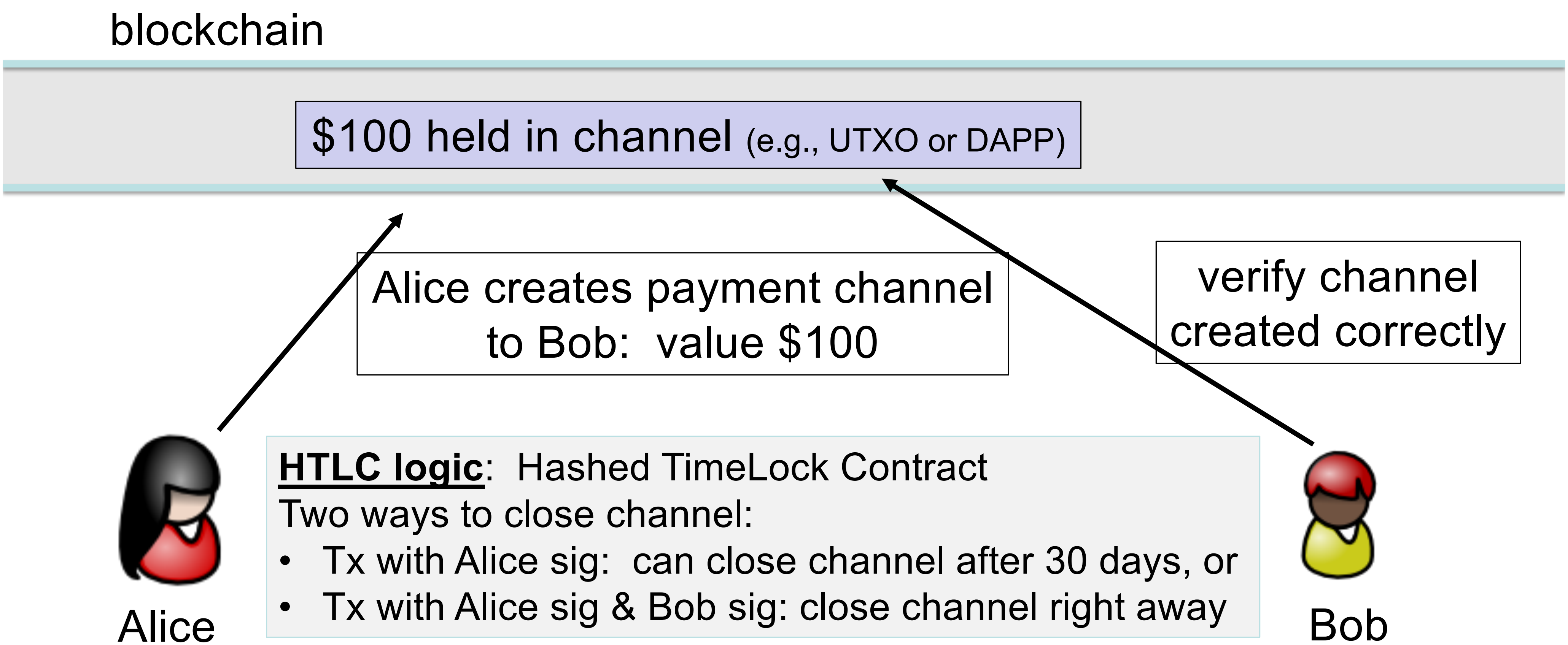
Can we scale blockchains to visa speeds? … with low Tx fees

Scaling approaches

Many approaches to scaling blockchains:

- **Faster consensus**: modern blockchains (e.g., Solana, Polkadot, Avalanche, …)
- **Payment channels**: most Tx are off chain Peer-to-Peer (e.g., Lightening)
- **Layer 2 approaches:**
  zkRollup, optimistic Rollup: batch many Tx into a single Tx
- Sidechains: Polygon and others
- many other ideas …

(1) Payment channels (high level idea)



(1) Payment channels (high level idea)

blockchain

$100 held in channel (e.g., UTXO)

Bob can sign Tx and close channel
… but he would rather wait (up to 30 days)

Alice

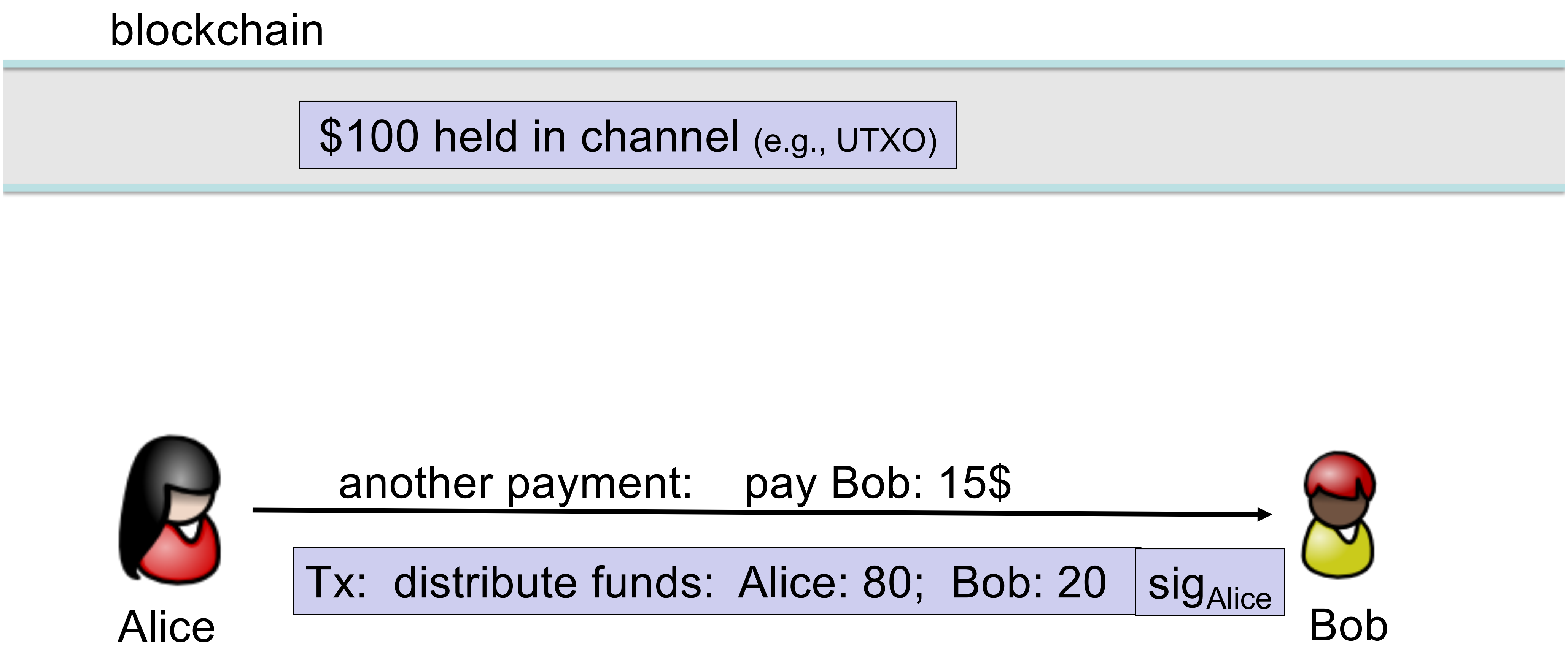
pay Bob: 5$

Tx: distribute funds: Alice: 95; Bob: 5 | $\text{sig}_{\text{Alice}}$

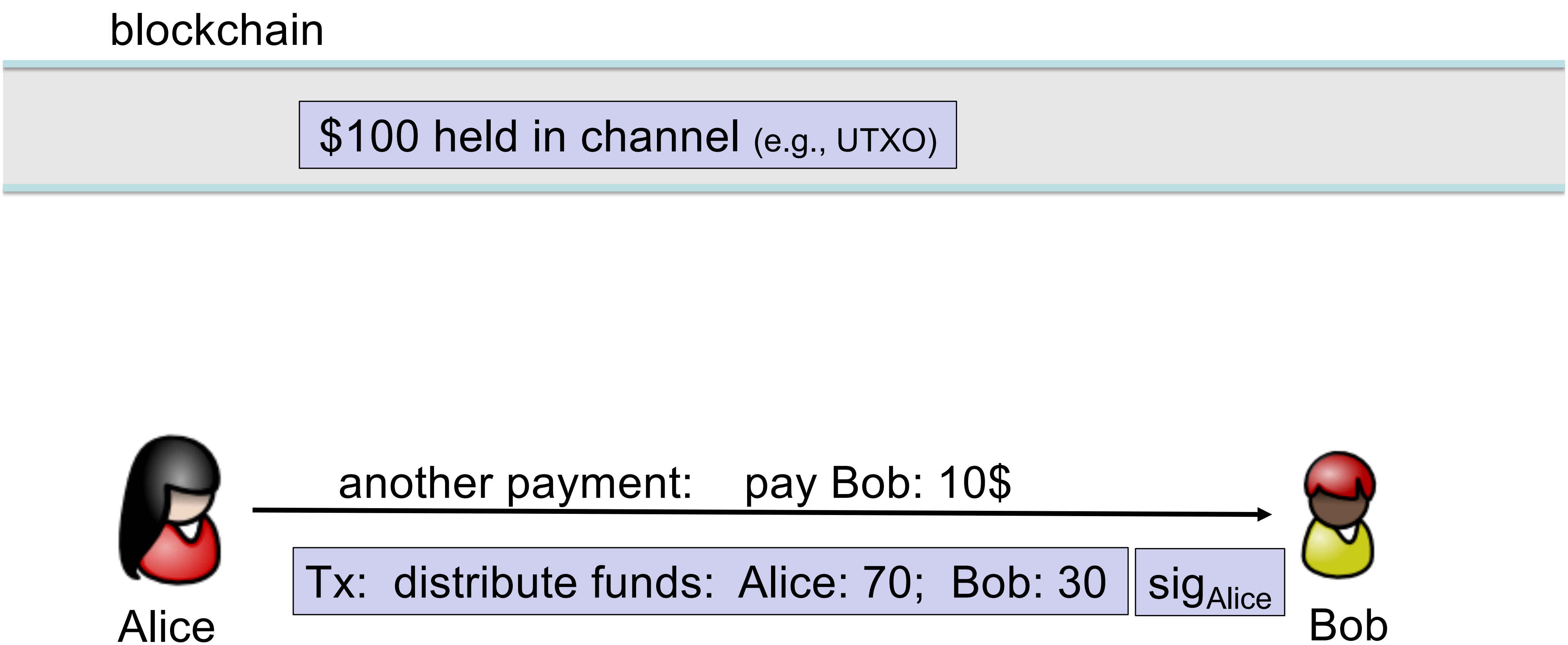
Bob

(off chain message!)

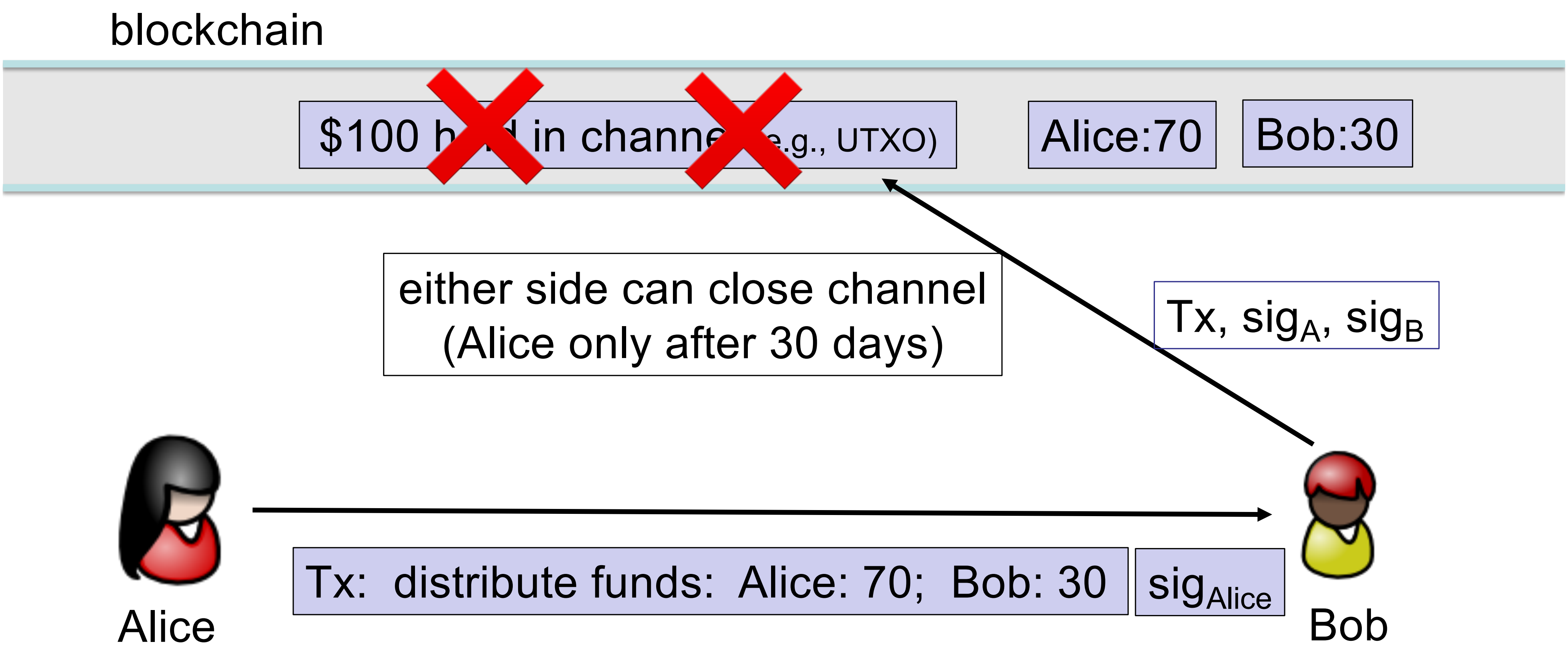
(1) Payment channels (high level idea)



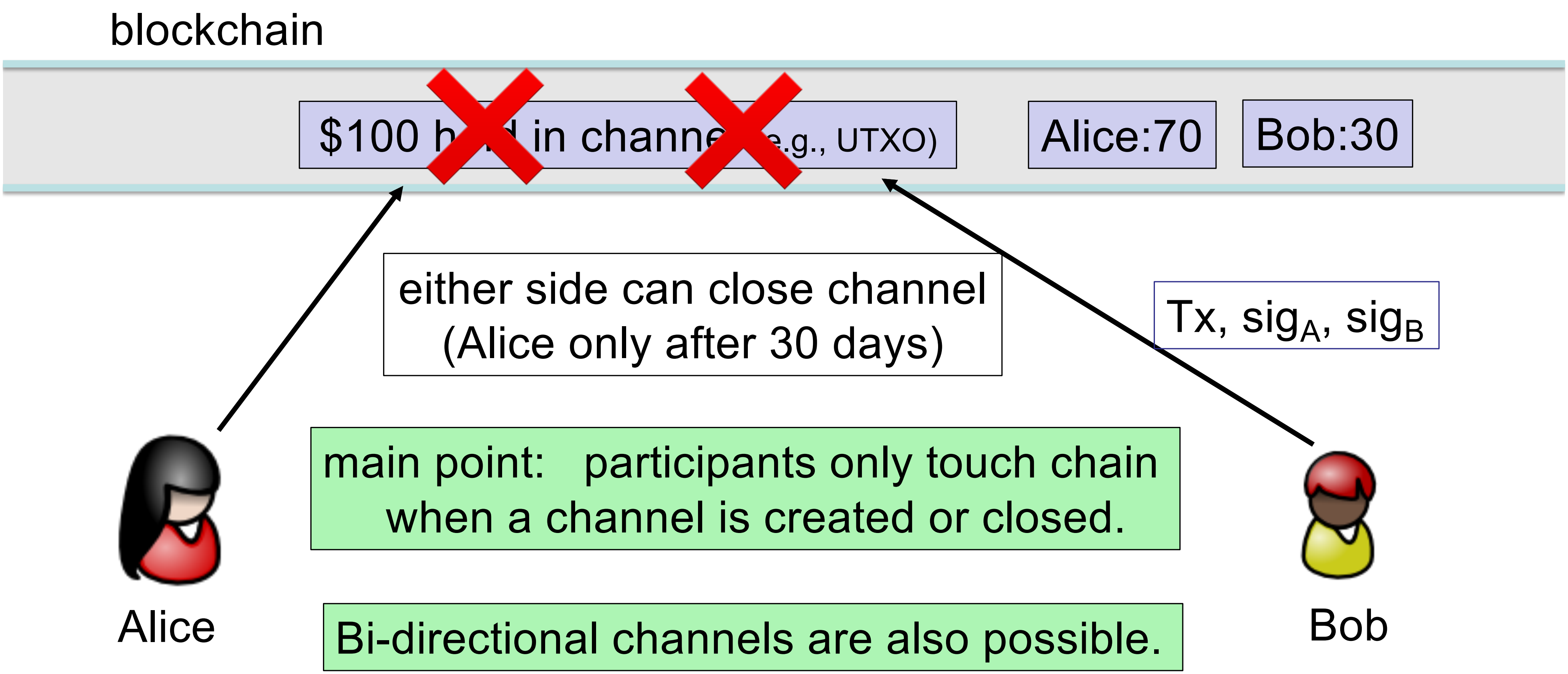
(1) Payment channels (high level idea)



(1) Payment channels (high level idea)

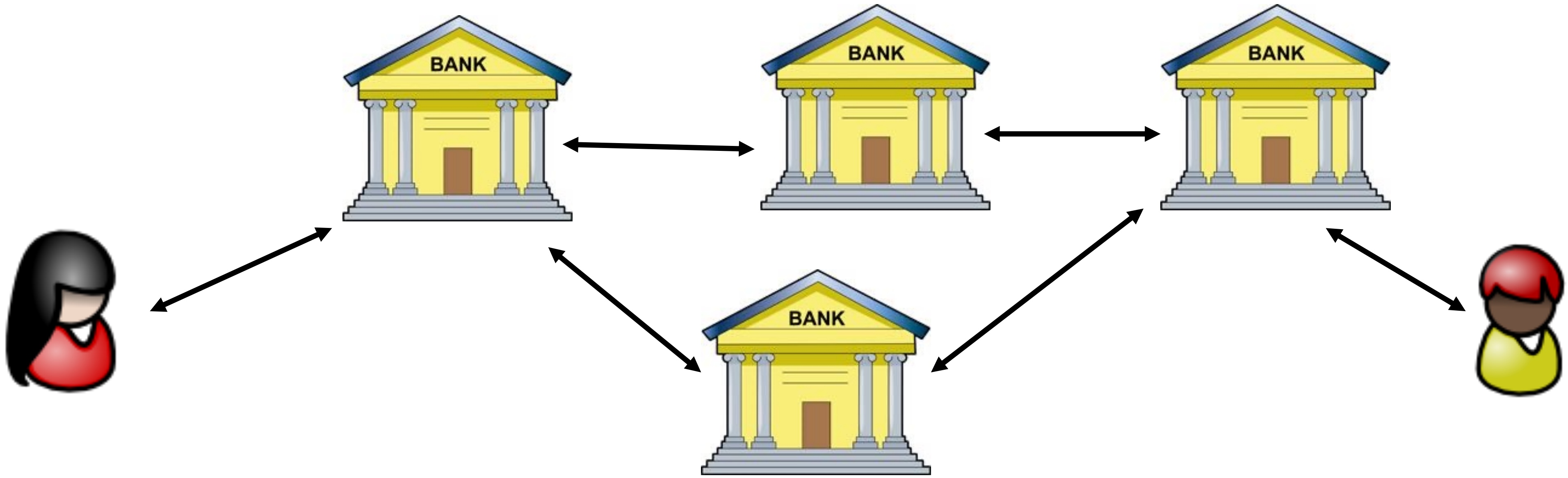


(1) Payment channels (high level idea)



Payment networks

Lots of bi-directional payment channels



Alice pays Bob by finding the cheapest route through the network

$\Longrightarrow$ while channels are open, nothing touches the blockchain

The case of El Salvador

June 24, 2021
8:18 PM PDT
Last Updated 2 months ago

Technology

Bitcoin to become legal tender in El Salvador on Sept 7, 2021

3 minute read

Reuters

Payment channels are necessary to enable state-wide adoption

- Strike wallet: connects to the Bitcoin Lightening network

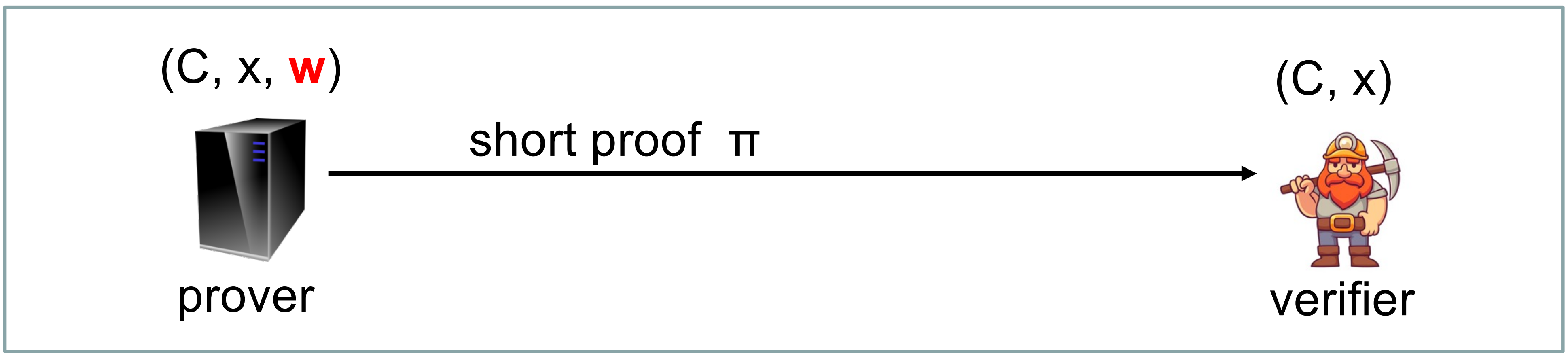
(2) Scaling Ethereum Using Rollup

(2) Scaling Ethereum Using Rollup

Main tool: SNARK (much more on SNARKs later)

C: a program that always terminates in ≤B steps

x: public input to C, **w**: private input to C



(2) Scaling Ethereum Using Rollup

Main point:
Verifier's run time is
much less than running C

more on SNARKs

I am convinced
prover knows w
s.t. C(x, w) = 1

inates in ≤B ste

x: public input to C, private input to C

(C, x, **w**)

short proof π

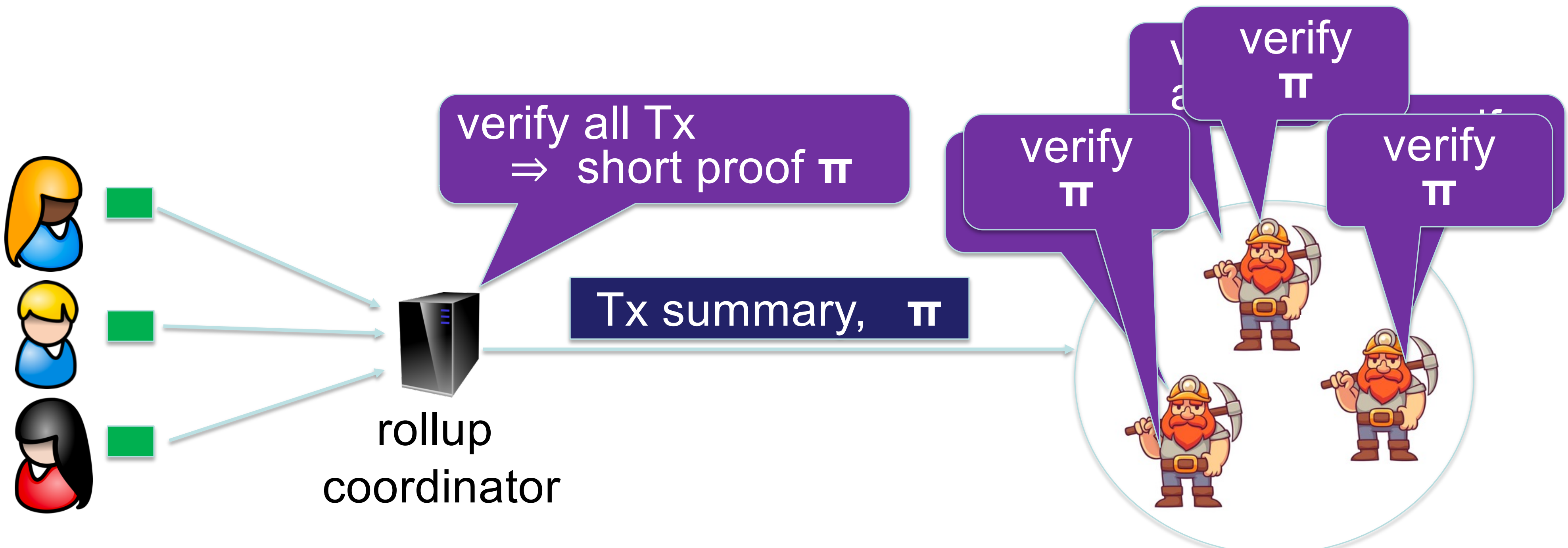
(C, x)

prover

verifier

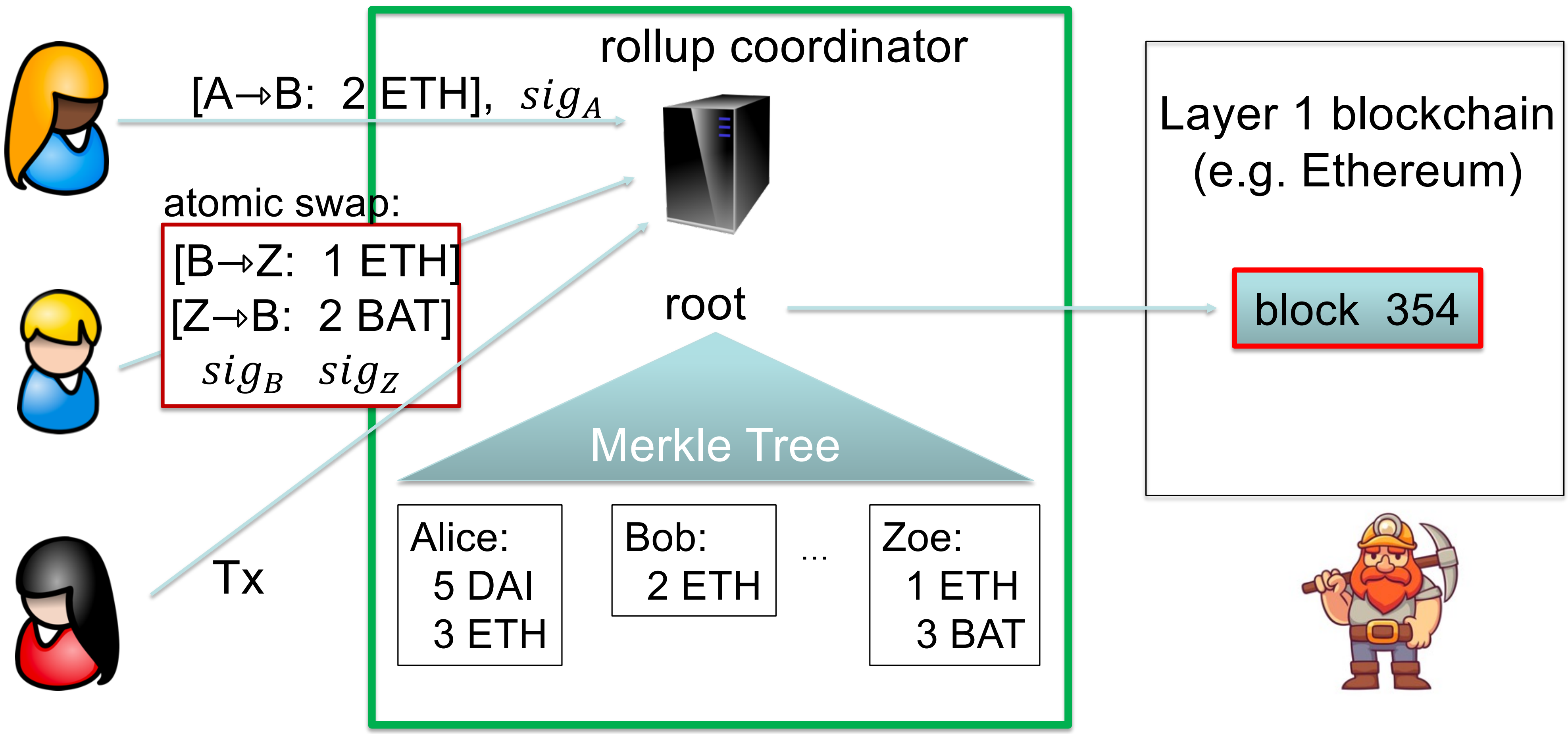
(2) Rollup: zk and optimistic

Standard L1 chains: every miner must verify every posted Tx

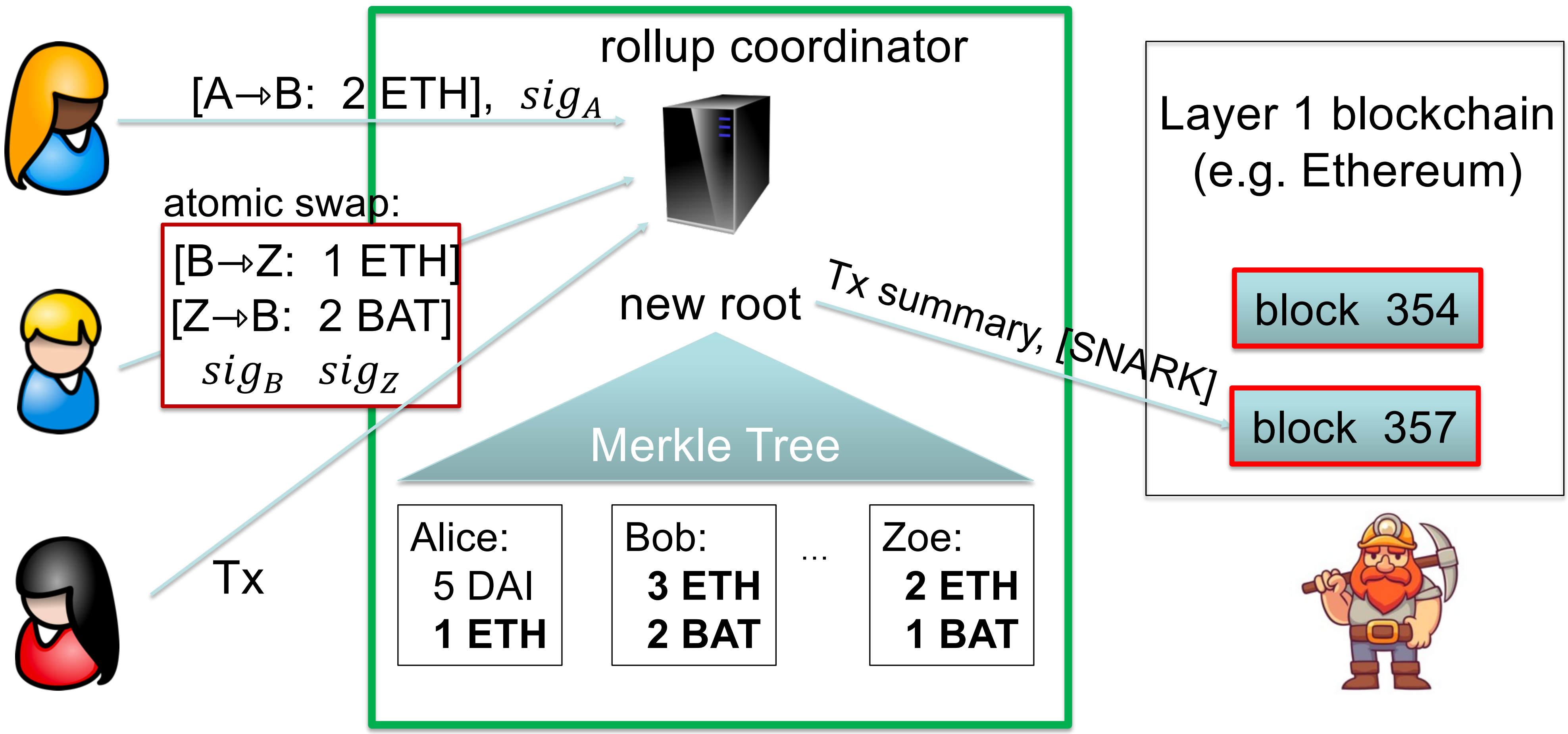


Rollup coordinator: compresses a thousand Tx into one on-chain proof (SNARK)

zkRollup (simplified)



zkRollup (simplified)



Transferring assets to and from L2

- Transactions within a Rollup system are easy:
  - Batch settlement on L1 network (e.g., Ethereum)

- Moving funds in to or out of Rollup system (L1 $\Leftrightarrow$ L2) is more expensive:
  - Requires posting more data on L1 network $\Longrightarrow$ higher Tx fees.

- Moving funds from one Rollup system to another (L2 $\Leftrightarrow$ L2)
  - Either via L1 network (expensive), or via a direct L2 $\Leftrightarrow$ L2 bridge (cheap)

Migrating a project from L1 Ethereum to L2 zkRollup

Upcoming development: **zkEVM** (e.g., MatterLabs and others).

Solidity compatibility:

- Coordinator can produce a SNARK proof for the execution of a short Solidity program:

$\Longrightarrow$ easy to migrate a DAPP from L1 Ethereum to L2 zkRollup.

$\Longrightarrow$ reduced Tx fees and increased Tx rate compared to L1

Optimistic Rollup (simplified) [e.g., Optimism, Arbitrum]

Same principle as zkRollup, but no SNARK proof

Instead: coordinator posts Tx data on chain without a proof

- Then give a few days for validators to complain:

  if a posted Tx is invalid $\Longrightarrow$

  anyone can submit a **fraud proof** and win a reward,

  Rollup server gets slashed.

Benefit: simple full EVM compatibility, less work for server.

Data availability: zkSync vs. zkPorter

Is the coordinator a central point of failure? (centralization fears??)

Answer: No!

coordinator fails $\Longrightarrow$ users find another coordinator to produce proofs

- Complication: new coordinator needs all current account information
  - How to get the data if the old coordinator is dead?

- Two solutions: zkSync and zkPorter. They work concurrently.

Data availability: zkSync vs. zkPorter

- **zkSync**: store all Tx summaries on the L1 blockchain (Ethereum)
  - L1 chain accepts Tx batch only if it includes summary of all Tx
  - Other coordinators can reconstruct L2 state from L1 blockchain
  - Downside: higher Ethereum Tx fees. Good for high value assets
- **zkPorter**: store Tx data on a new blockchain
  - maintained by a set of staked coordinators
  - Cheap off-chain storage, but lower guarantee than zkSync
- Customer can choose how coordinator will store its account.

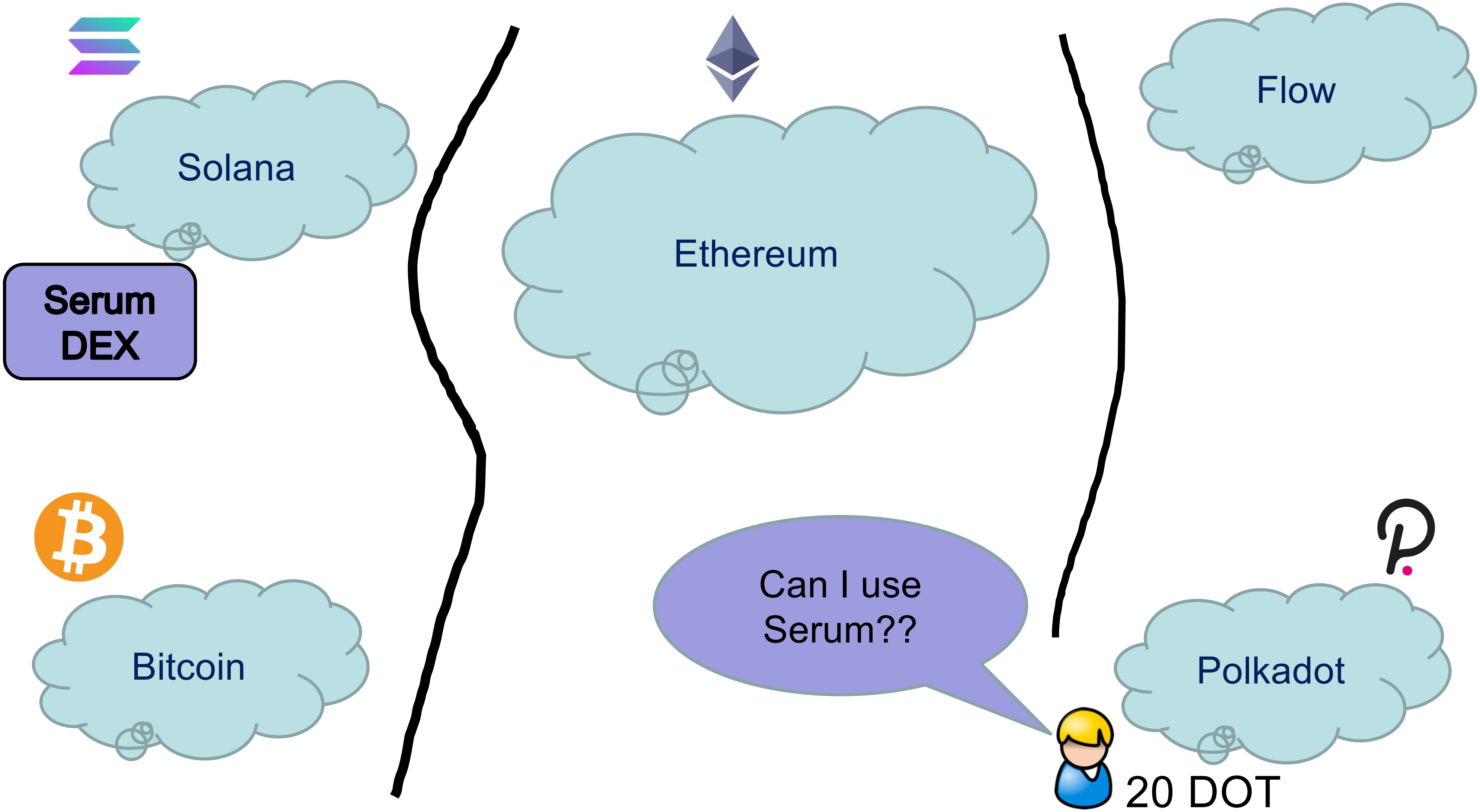
That’s it on this topic …

Next segment: interoperability

How to move assets from one chain to another

Interchain Interoperability

https://defi-learning.org/


Solana
Serum DEX
Bitcoin
Ethereum
Can I use Serum??
Flow
Polkadot
20 DOT

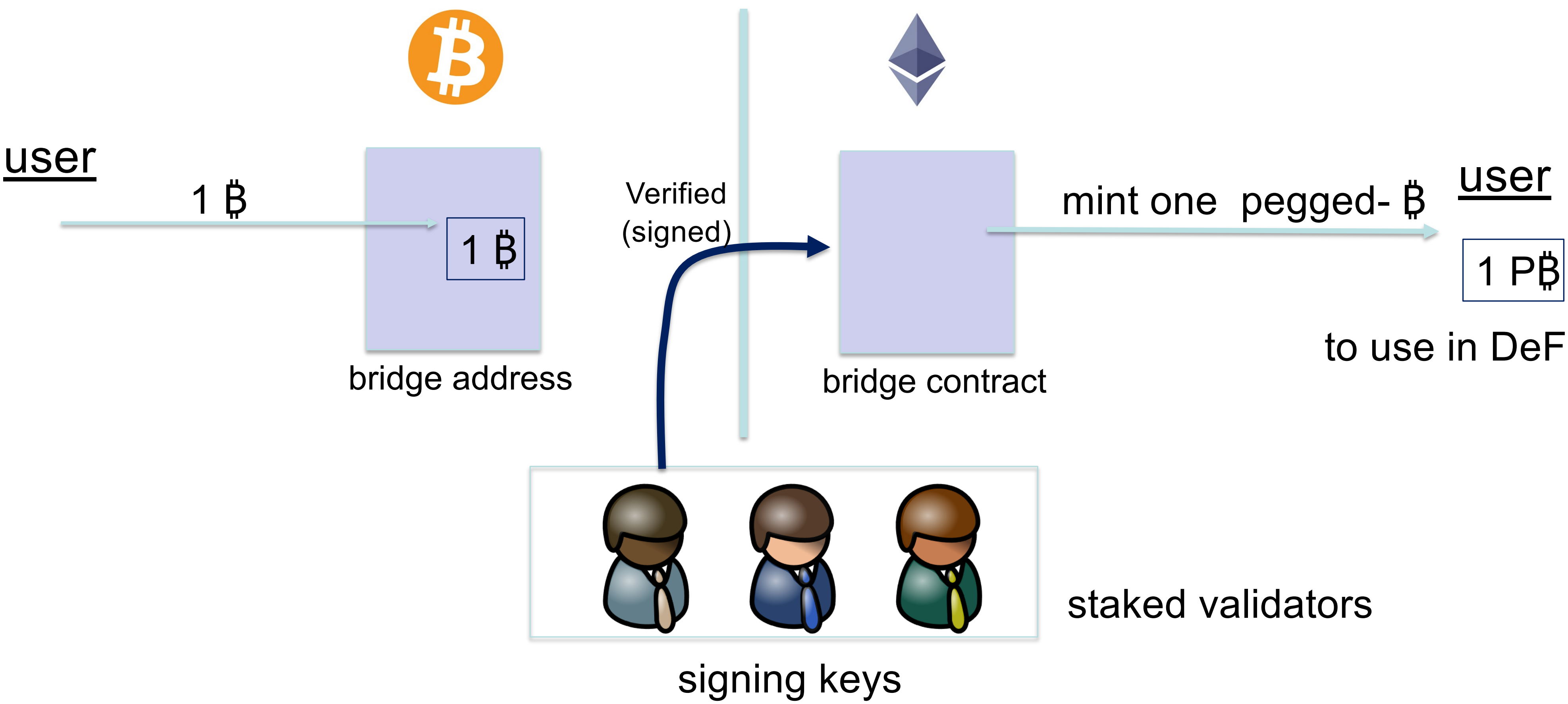
Interoperability

- **Interoperability**:
  - a user owns funds or assets on one blockchain system.
    Goal: enable the user to move funds and/or assets to another system.
- **Composability**:
  - enable a DAPP on one blockchain to call a DAPP on another

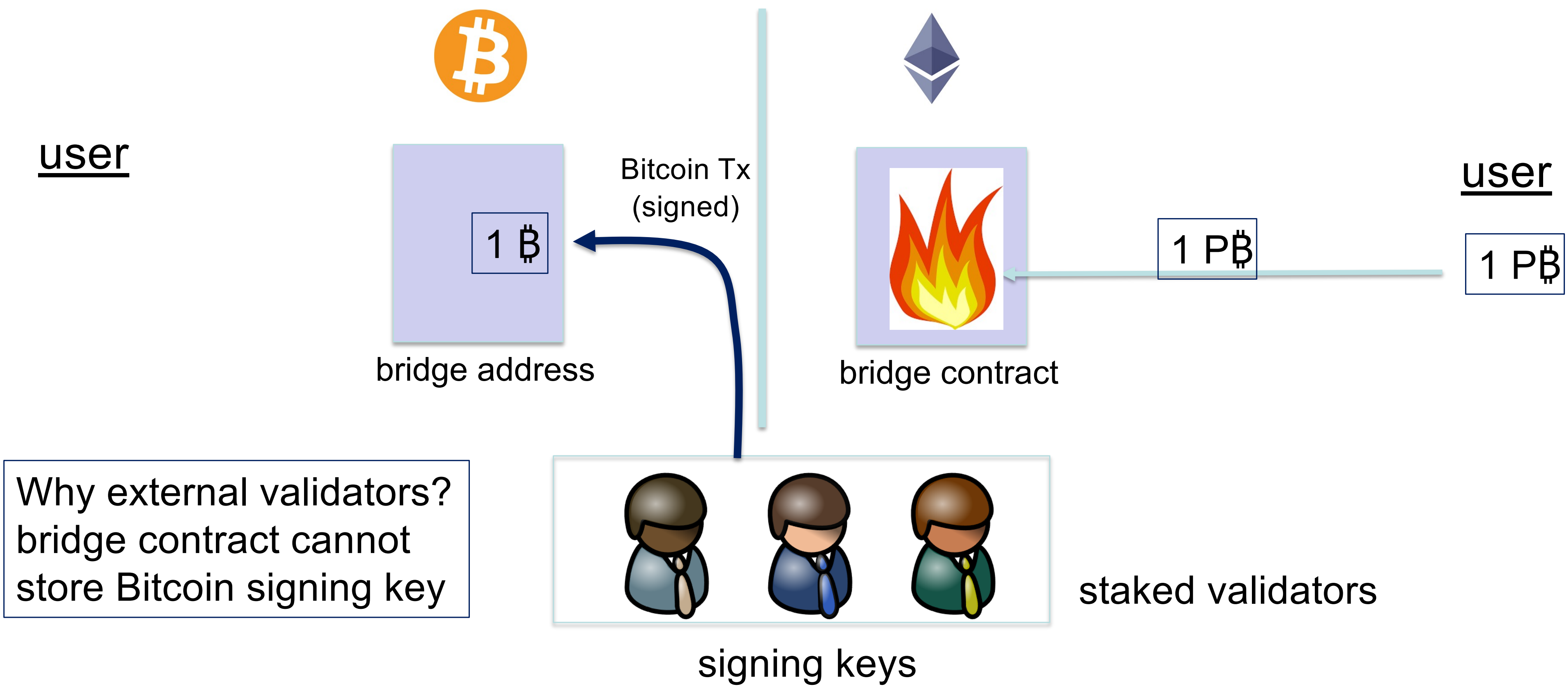
Both are easy if the entire world used Ethereum

- In reality: many blockchain systems that need to interoperate
- Several cross-chain protocols: XCMP, IBC, …

How to move assets? Building a federated bridge (simplified)



How to move assets? Building a federated bridge (simplified)



End of lecture: quick review

Cryptographic primitives:

- Hash functions: committing to large amounts of data
- Digital signatures: authorizing actions

Scaling the blockchain

- Payment channels and Rollups

Interoperability: via bridges and pegged coins.


END OF TOPIC
https://defi-learning.org/