

# Scaling the blockchain part II: Rollups

Yu Feng

University of California, Santa Barbara



# Scaling the blockchain: the problem

Transaction rates (Tx/sec):

- Bitcoin: can process about 7 (Tx/sec)
  - Ethereum: can process about 15 (Tx/sec)
  - The visa network: can process up to 24,000 (Tx/sec)
- 
- Tx Fees fluctuate:  
2\$ to 60\$ for simple Tx

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

# How to process more Tx per second

Many ideas:

- Use a faster consensus protocol
- Parallelize: split the chain into independent shards
- Today: Rollups, move the work somewhere else
- Payment channels: reduce the need to touch the chain
  - Requires locking up funds; mostly designed for payments.

reduces  
composability

# Recall: a basic layer-1 blockchain

Can handle 15 Tx/sec ...



Tx<sub>A</sub>



Tx<sub>B</sub>

⋮

World state: balances, storage, etc.

A layer-1 blockchain  
(e.g., Ethereum)

current world state



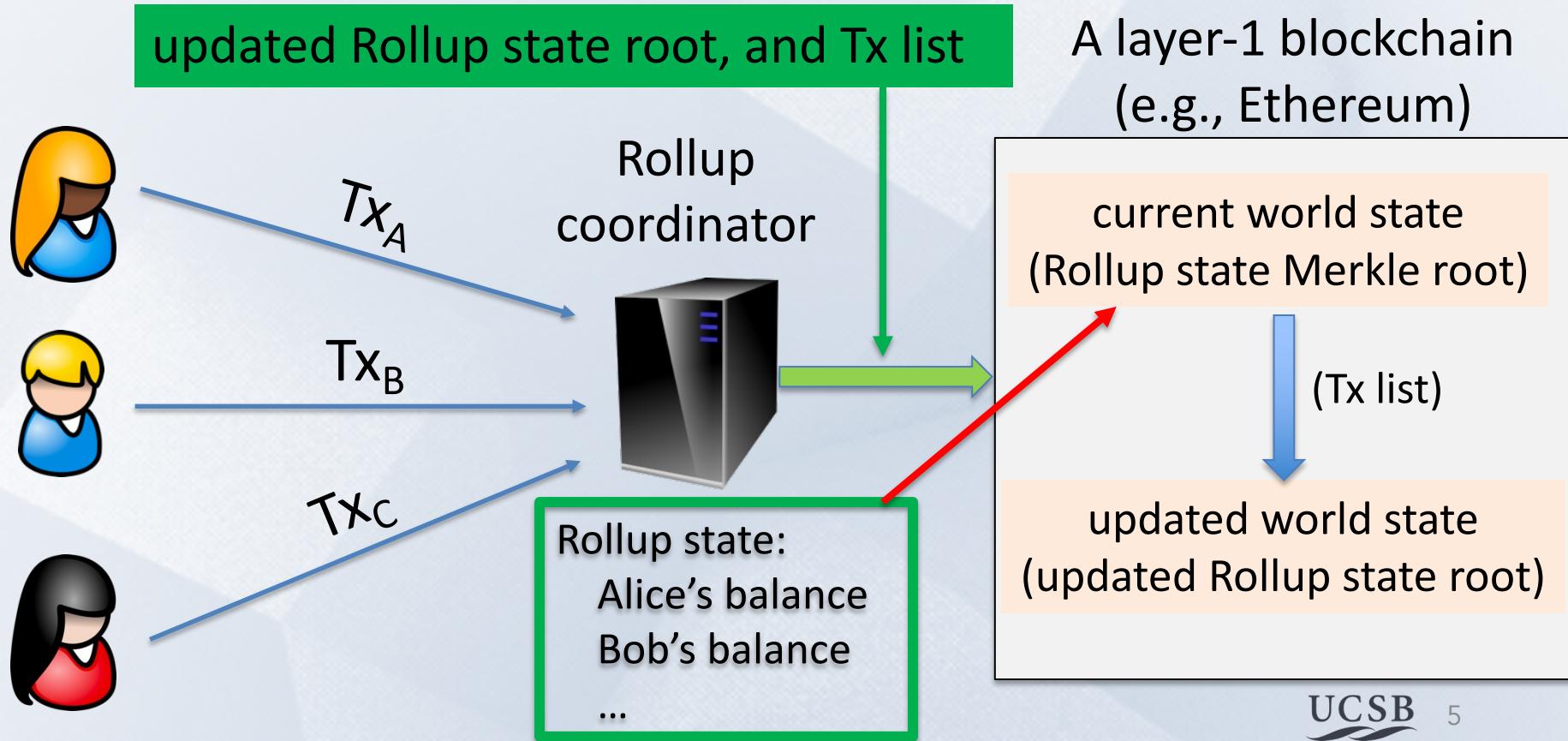
updated world state



updated world state



# Rollup idea 1: batch many Tx into one



# Rollup idea 1: batch many Tx into one

## Key point:

- *Hundreds* of transactions on Rollup state are batched into a *single* transaction on layer-1

⇒ 100x speed up in Tx/sec

- Let's see how ...

Rollup state:  
Alice's balance  
Bob's balance  
...

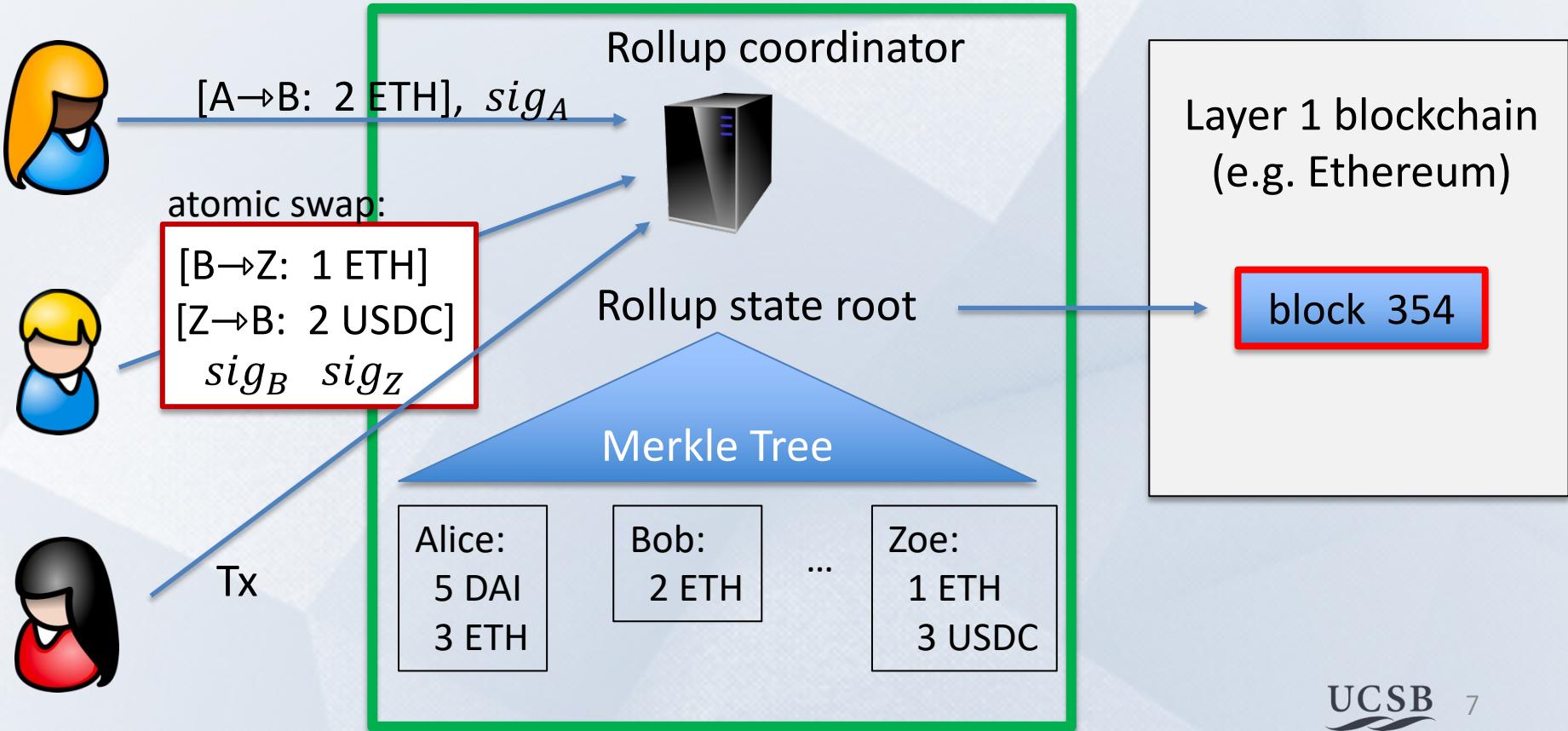
A layer-1 blockchain  
(e.g., Ethereum)

current world state  
(Rollup state Merkle root)

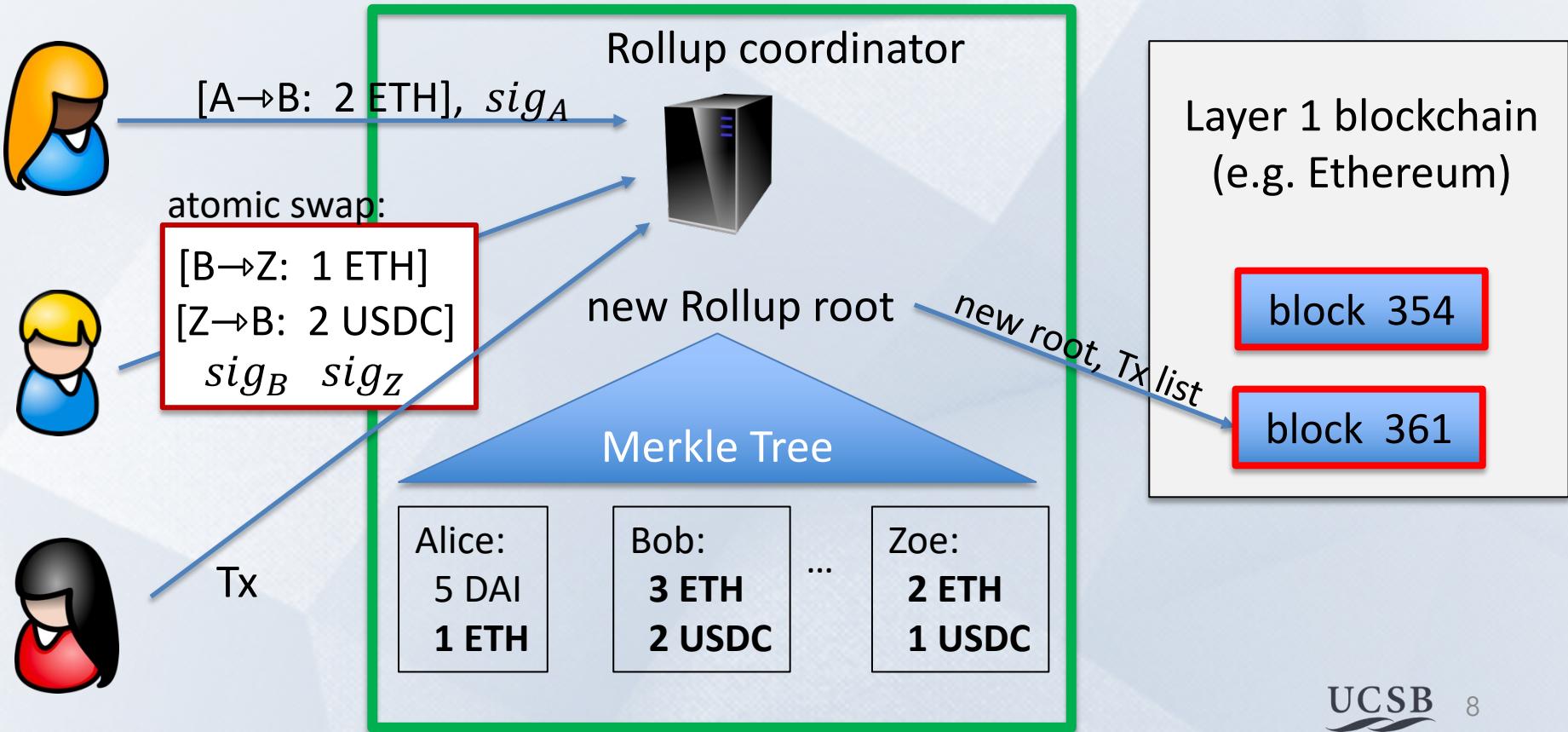
(Tx list)

updated world state  
(updated Rollup state root)

# Rollup operation (simplified)

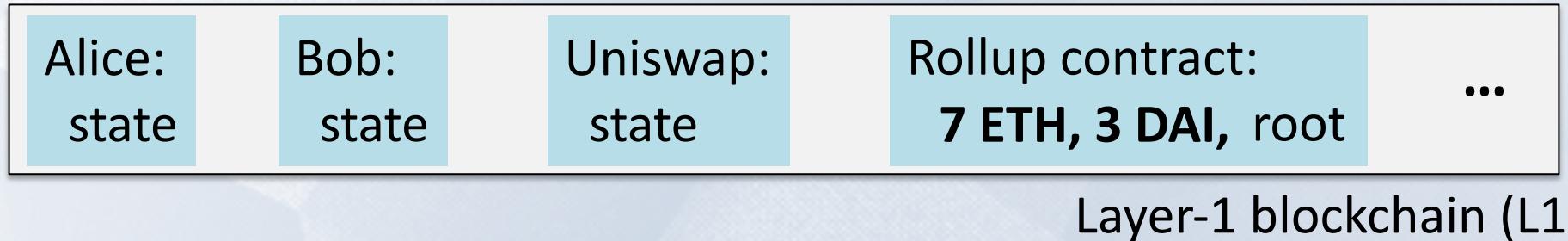
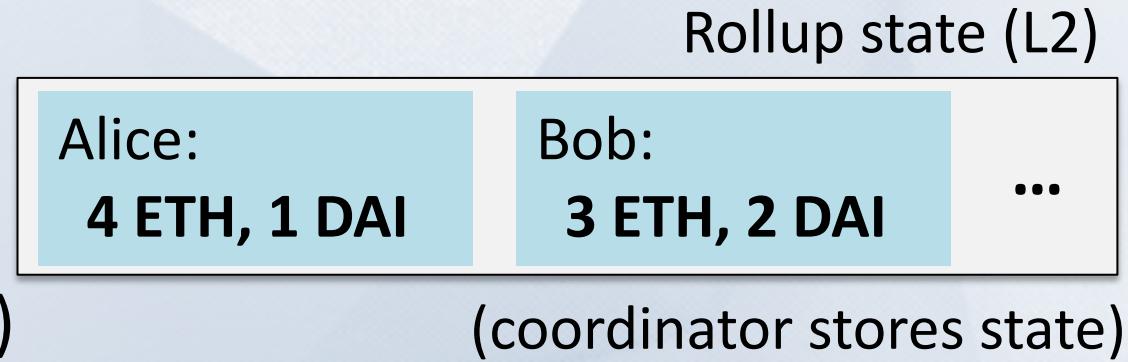


# Rollup operation (simplified)



# In more detail

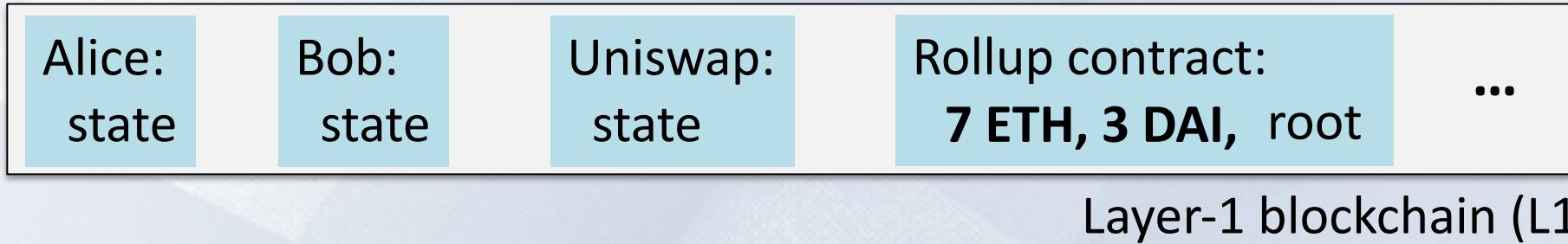
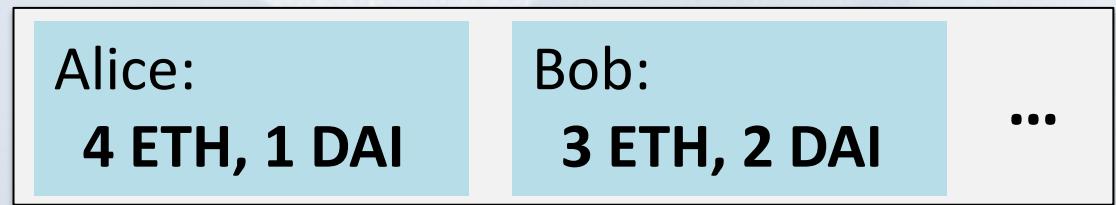
Rollup contract on  
layer-1 holds assets  
of all Rollup accounts  
(and Merkle state root)



# Transfers inside Rollup are easy (L2 → L2)



[A→B: 2 ETH],  $sig_A$   
(with hundreds of Tx)



# Transfers inside Rollup are easy ( $L2 \rightarrow L2$ )

Coordinator updates root on Rollup contract

Rollup state ( $L2$ )



[A → B: 2 ETH],  $sig_A$   
(with hundreds of Tx)

Alice:

**2 ETH, 1 DAI**

Bob:

**5 ETH, 2 DAI**

...

new Merkle root,  
Tx list

Alice:  
state

Bob:  
state

Uniswap:  
state

Rollup contract:  
**7 ETH, 3 DAI, root**

...

Layer-1 blockchain ( $L1$ )

# Transferring funds into Rollup ( $L1 \rightarrow L2$ )

Alice issues an L1 Tx: slow and expensive

Rollup state (L2)



[2 ETH],  $\sigma_A$

Alice: <b>2 ETH, 1 DAI</b>	Bob: <b>5 ETH, 2 DAI</b>	...
-------------------------------	-----------------------------	-----

Alice:  
state

Bob:  
state

Uniswap:  
state

Rollup contract:  
**7 ETH, 3 DAI, root**

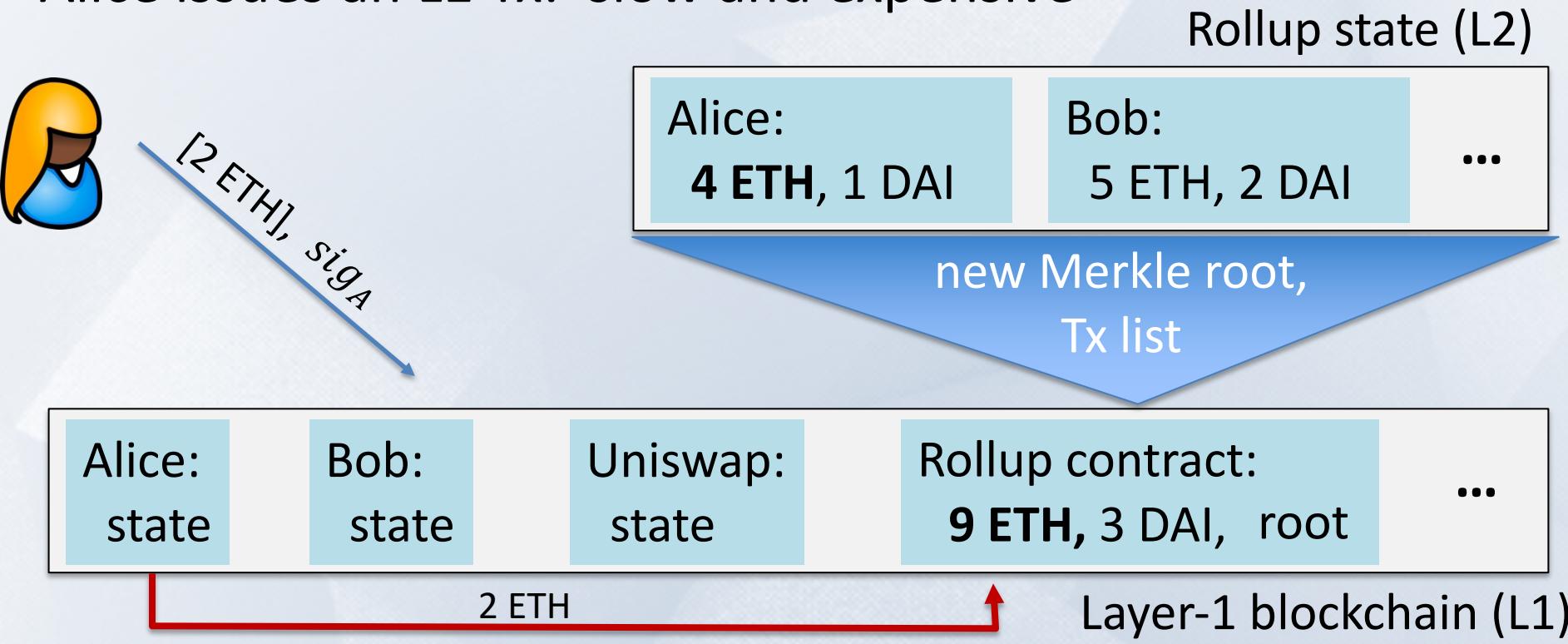
...

2 ETH

Layer-1 blockchain (L1)

# Transferring funds into Rollup ( $L1 \rightarrow L2$ )

Alice issues an L1 Tx: slow and expensive

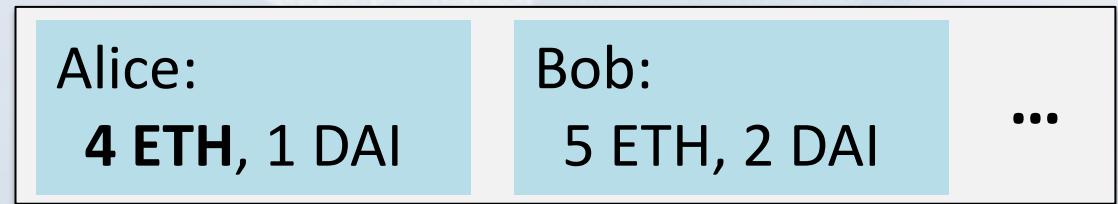


# Transferring funds out of Rollup (L2 → L1)

Requires extra gas on L1 to process transfer



[withdraw 4 ETH],  $sig_A$   
(plus hundreds of Tx)



Alice:  
state

Bob:  
state

Uniswap:  
state

Rollup contract:  
**9 ETH, 3 DAI, root**

...

...

Layer-1 blockchain (L1)

# Transferring funds out of Rollup ( $L2 \rightarrow L1$ )

Requires extra gas on L1 to process transfer



[withdraw 4 ETH],  $sig_A$   
(plus hundreds of Tx)

Rollup state		
Alice: <b>0 ETH, 1 DAI</b>	Bob: <b>5 ETH, 2 DAI</b>	...

new Merkle root,  
Tx list

Alice: state	Bob: state	Uniswap: state	Rollup contract: <b>5 ETH, 3 DAI, root</b>	...
-----------------	---------------	-------------------	---	-----

4 ETH

Layer-1 blockchain (L1)

# Summary: transferring Rollup assets

Transactions within a Rollup are easy:

- Batch settlement on L1 network (e.g., Ethereum)

Moving funds into or out of Rollup system ( $L1 \leftrightarrow L2$ ) is expensive:

- Requires posting more data on L1 network  $\Rightarrow$  higher Tx fees.

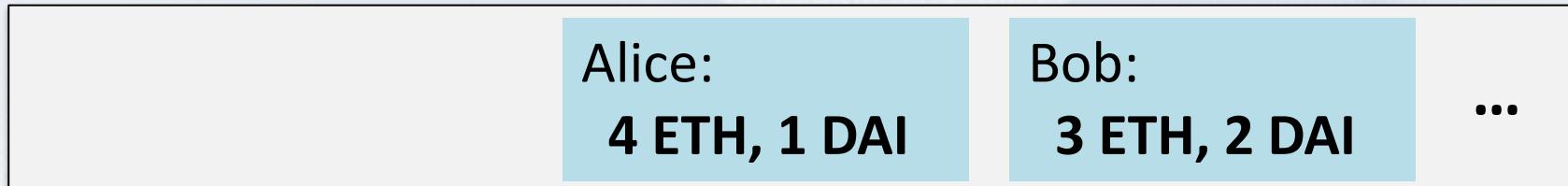
Moving funds from one Rollup system to another ( $L2 \leftrightarrow L2$ )

- Either via L1 network (expensive),  
or via a direct  $L2 \leftrightarrow L2$  bridge (cheap)

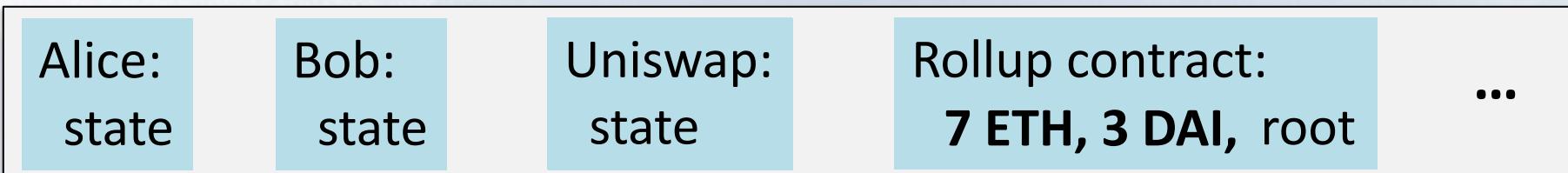
# Running contracts on a Rollup?

Two copies of Uniswap

Rollup state (L2)



⇒ Rollup users can cheaply interact with Uniswap on Rollup



Layer-1 blockchain (L1)

# Running contracts on a Rollup?

Rollup state (L2)			
Uniswap: state	Alice: <b>4 ETH, 1 DAI</b>	Bob: <b>3 ETH, 2 DAI</b>	...

Coordinator maintains state of all contracts on Rollup system:

- It updates the Uniswap Merkle leaf after every Tx to Uniswap
- Writes updated Rollup state Merkle root to L1 chain

# Running contracts on a Rollup?

Rollup state (L2)

Uniswap: state	Alice: <b>4 ETH, 1 DAI</b>	Bob: <b>3 ETH, 2 DAI</b>	...
-------------------	-------------------------------	-----------------------------	-----

Rollup functions as Ethereum, but ...

- It relies on the L1 chain to attest to the current Rollup state

# How to send Tx to the coordinator

Enduser configures its wallet to send Tx to the RPC points of the selected Rollup.

The screenshot shows the Metamask settings interface. On the left, a sidebar lists options: General, Advanced, Contacts, Security & privacy, Alerts, Networks, Experimental, and About. The 'Networks' option is selected and highlighted with a blue background. The main content area is titled 'Settings' and has a search bar at the top. Below it, the 'Networks' section is titled 'Add a network'. It contains a list of popular custom networks: Arbitrum One, Avalanche Network C-Chain, BNB Smart Chain (previously Binance Smart Chain Mainnet), Fantom Opera, Harmony Mainnet Shard 0, and Optimism. Each network entry includes a small icon, the name, and an 'Add' button to its right. The 'Add' button for Arbitrum One is circled in red, and the 'Add' button for Optimism is also circled in red.

(by default Metamask sends Tx to the Ethereum Mainnet RPC points)

# The role of the Coordinator

The Coordinator has multiple tasks:

- **Sequence** incoming Tx from Rollup users into a stream of Tx
  - ⇒ can extract MEV from searchers, in addition to Tx fees
  - ⇒ very profitable to be a Rollup coordinator
- **Execute** the stream of Tx on the latest Rollup state
- **Push updates** to the L1 chain

**Shared coordinator:** one coordinator for multiple Rollups (not today)

# Coordinator architectures

A centralized coordinator:

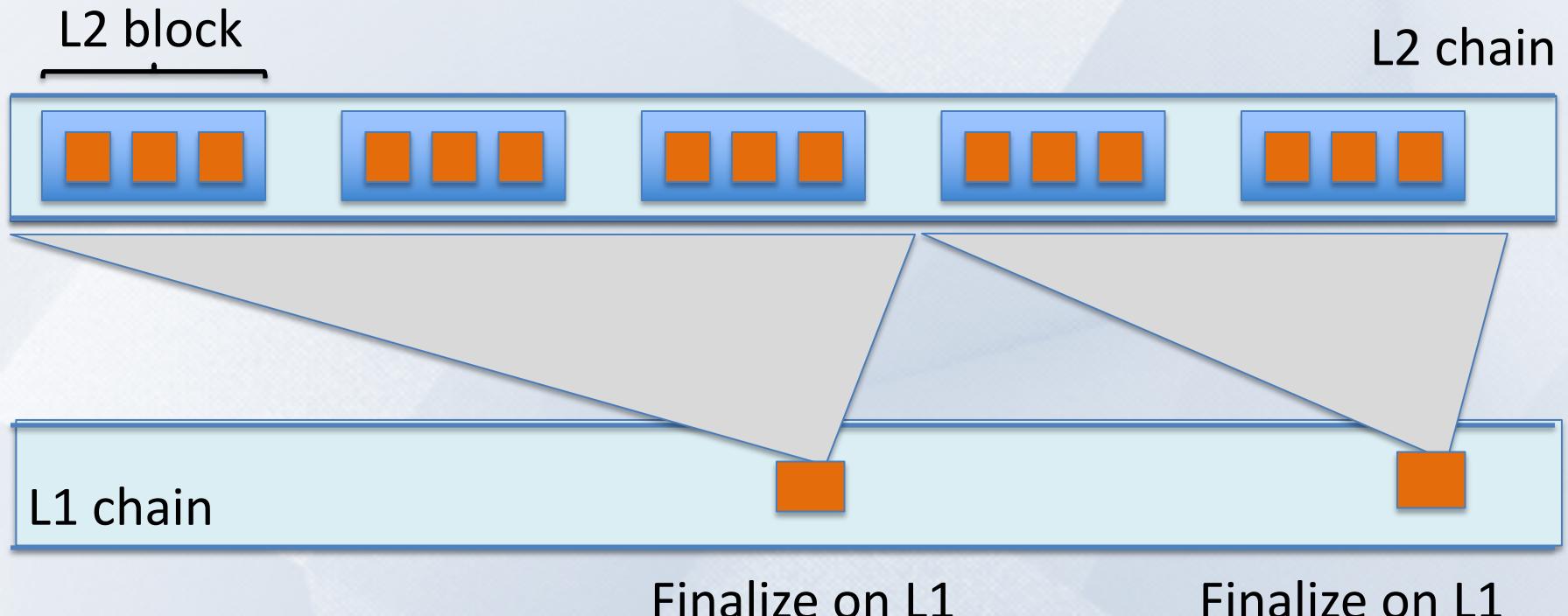
- availability and censorship concerns,  
... but cannot steal assets (as we will see)

A decentralized coordinator:

- a set of parties that run a fast consensus protocol
- At every epoch one party is chosen to sequence, execute, and push updates to the L1

Importantly: L1 provides ground truth of the Rollup state

# Tx rate on L2 is higher than on L1



# An example (StarkNet -- using validity proofs)

Block				
Number	Hash	Status	Num. of Txs	Age
PENDING	PENDING	PENDING	64	3min
13011	0x0432...2380	ACCEPTED_ON_L2	82	8min
13010	0x0492...f0d1	ACCEPTED_ON_L2	122	15min
13009	0x0081...b7af	ACCEPTED_ON_L2	127	24min
...				
12868	0x060c...15eb	ACCEPTED_ON_L2	58	8h
12867	0x0654...3b0f	ACCEPTED_ON_L1	72	9h
12866	0x0779...57d6	ACCEPTED_ON_L1	63	9h
12865	0x06ae...943f	ACCEPTED_ON_L1	97	9h

Source: starkscan.co

Tx posted on L1 (Ethereum) about every eight hours

# Not so simple ...



# Problems ...

**Problem 1:** what if coordinator is dishonest?

- It could steal funds from the Rollup contract
- It could issue fake Tx on behalf of users

**Problem 2:** what if coordinator stops providing service?

- If Rollup state is lost, how can users issue Tx?  
... can't compute updated Rollup Merkle root.

# Problem 1: what if coordinator is dishonest?

Can coordinator steal funds from Rollup users?

No! L1 chain verifies that Rollup state updates are valid.

⇒ all Tx are valid and properly signed by the Rollup users

Challenge: how to do this cheaply ?? (with little gas on L1)

Alice:  
state

Bob:  
state

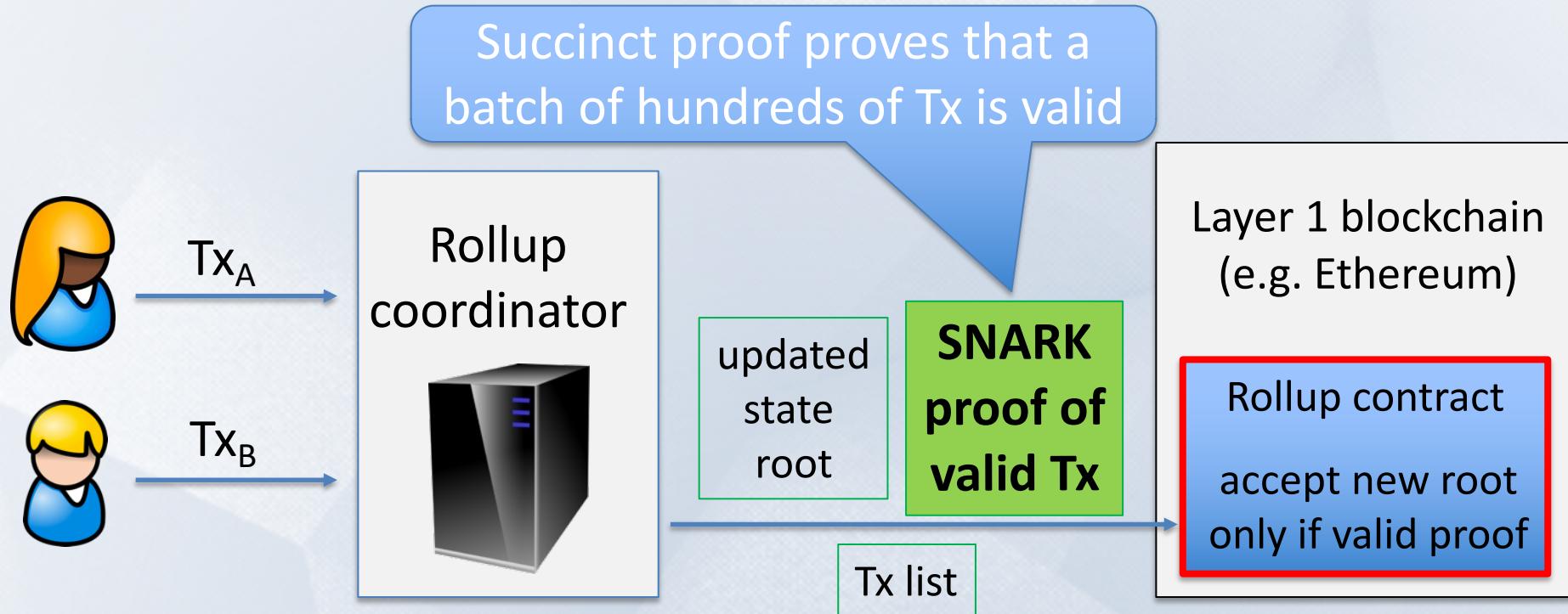
Rollup contract:  
**7 ETH, 3 DAI, root**

...

Layer-1 blockchain (L1)

# Verifying Rollup state updates

Approach 1: validity proofs (called a zk-Rollup)



# What the SNARK proof proves

SNARK proof is **short** and **fast** to verify:

⇒ Cheap to verify proof on the slow L1 chain (with EVM support)  
(usually not a zero knowledge proof)

**Public statement:** (old state root, new state root, Tx list)

**Witness:** (state of each touched account pre- and post- batch,  
Merkle proofs for touched accounts, user sigs)

SNARK proof proves that:

- (1) all user sigs on Tx are valid,
- (2) all Merkle proofs are valid,
- (3) post-state is the result of applying Tx list to pre-state

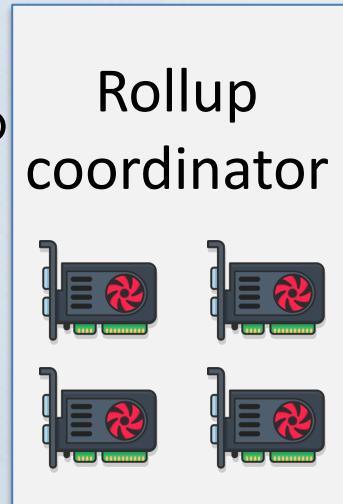
# zkEVM

When a contract (e.g. Uniswap) runs on a Rollup:

- coordinator builds a SNARK proof of correct execution of an EVM program ⇒ called a zkEVM
- Generating proof is a heavyweight computation  
... verifying proof is fast

Flavors of a zkEVM:

- Prove that EVM bytecode ran correctly  
(Polygon zkEVM, Scroll)
- Compile Solidity to a SNARK-friendly circuit  
(MatterLabs)



# The end result

Rollup contract on L1 ensures coordinator cannot cheat:

- all submitted Tx must have been properly signed by users
  - all state updates are valid
- ⇒ Rollup contract on L1 will accept any update with a valid proof
- ⇒ Anyone can act as a coordinator (with enough compute power)

# Verifying Rollup state updates

Approach 2: fault proofs (called an optimistic Rollup)

- Coordinator deposits stake in escrow on L1 Rollup contract
- Operation: Coordinator submits state updates to L1 w/o a proof
- If update is invalid: anyone has seven days to submit a fault proof
  - Successful fault proof means coordinator gets slashed on L1
  - Unsuccessful fault proof costs complainer a fee

Challenge: how to prove a fault to the Rollup contract on L1 ??

Naively: L1 can re-execute all Tx in batch  $\Rightarrow$  expensive and slow

# Fault Proof game



coordinator

pre-root

Tx list



fault claim

claimed  
post-root

Coordinator computes Merkle tree of all states.  
Sends Merkle root to L1

different  
post-root

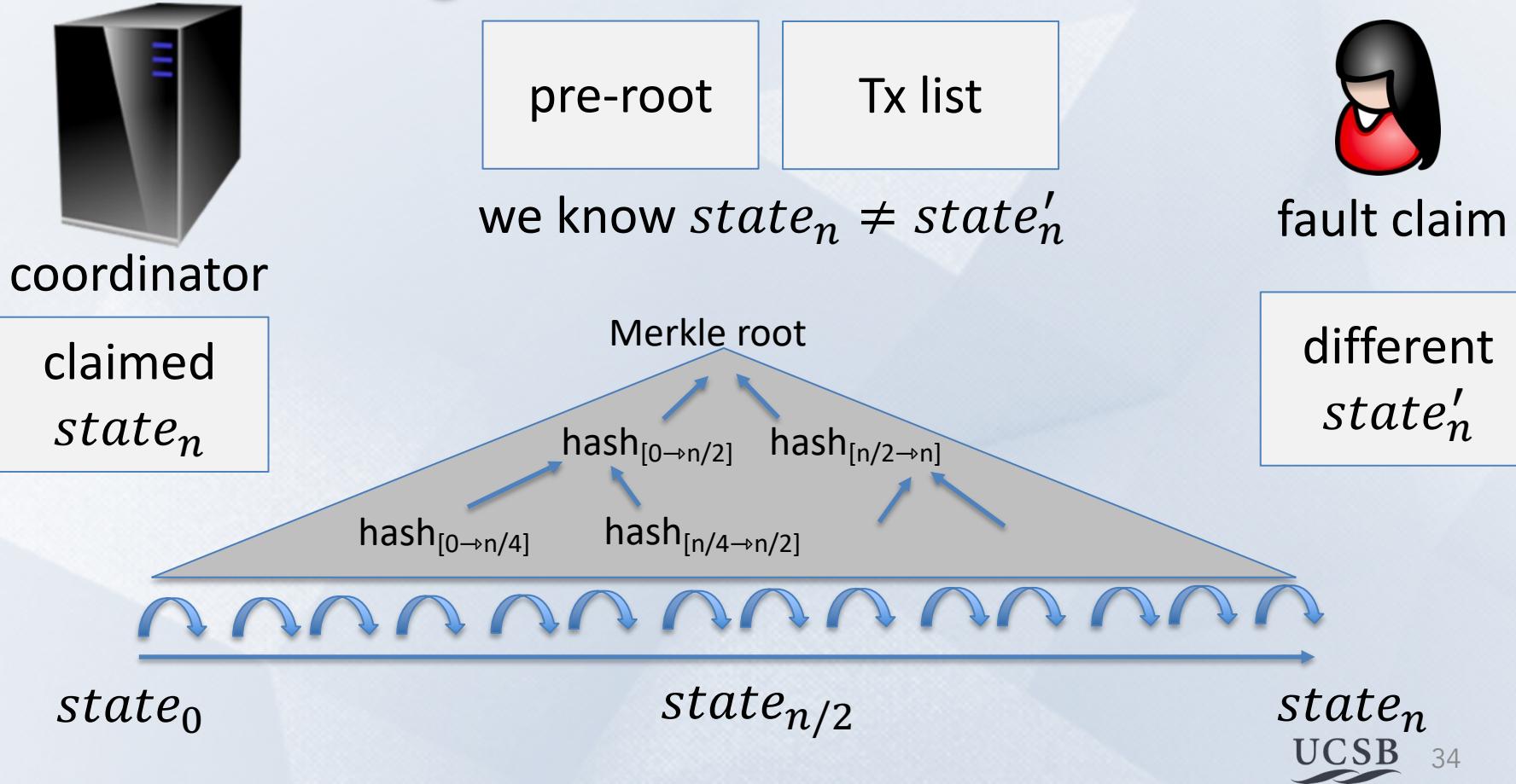
pre-  
state



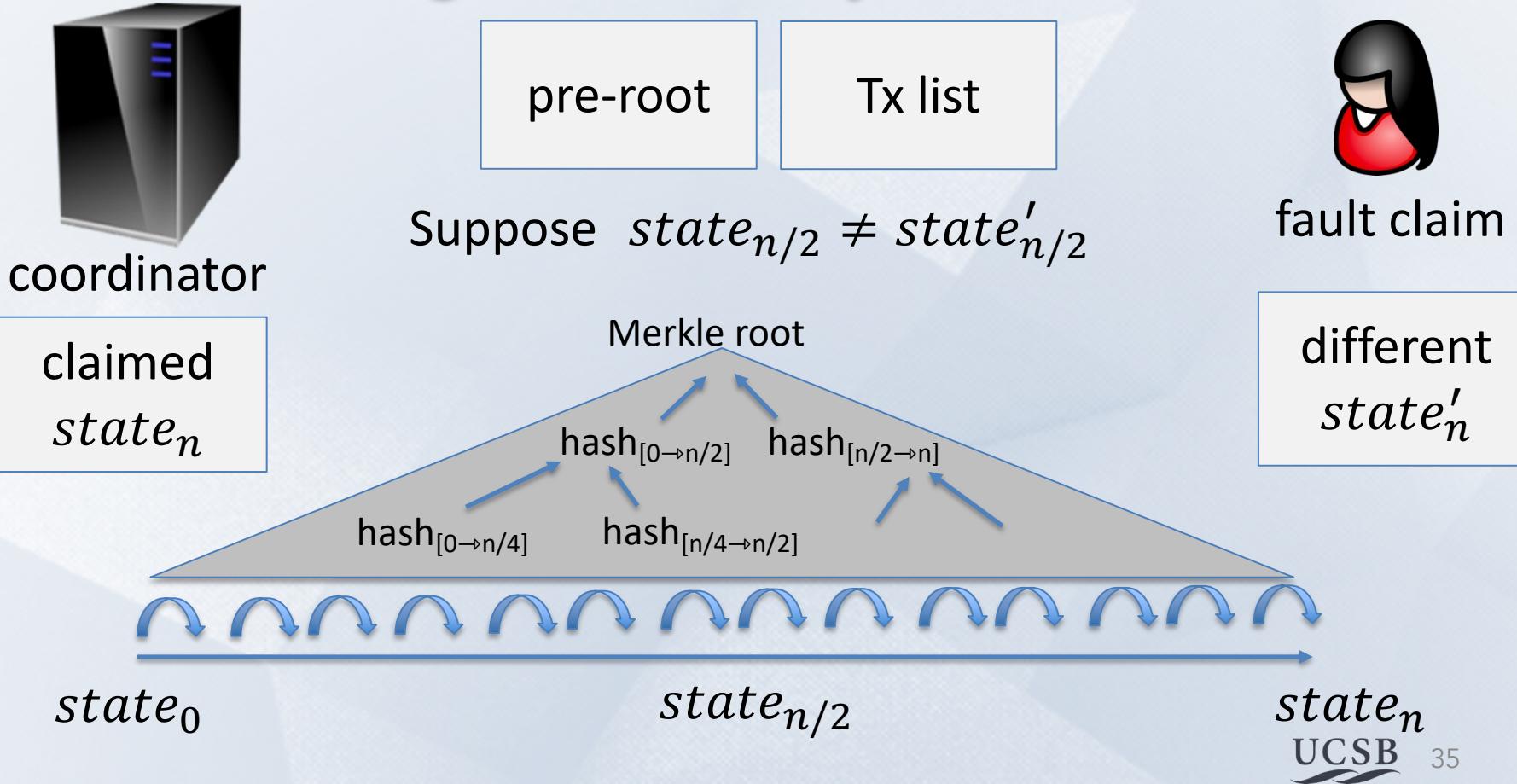
break computation into small steps

post-  
state

# Fault Proof game



# Fault Proof game: binary search



# Fault Proof game: binary search

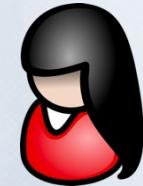


coordinator

claimed  
 $state_n$

pre-root

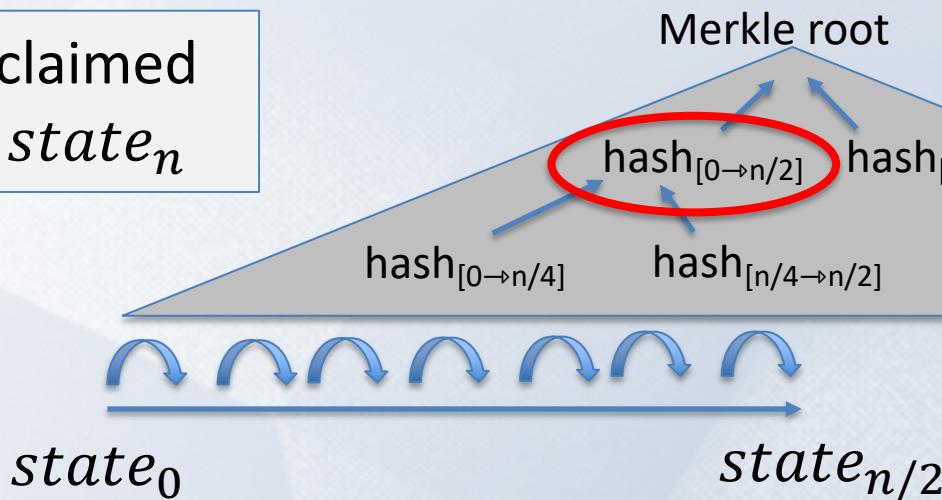
Tx list



fault claim

different  
 $state'_n$

Suppose  $state_{n/2} \neq state'_{n/2}$



Coordinator sends  $hash_{[0 \rightarrow n/2]}$  to L1  
Alice sends "left" to L1

# Fault Proof game: binary search



coordinator

claimed  
 $state_n$

pre-root

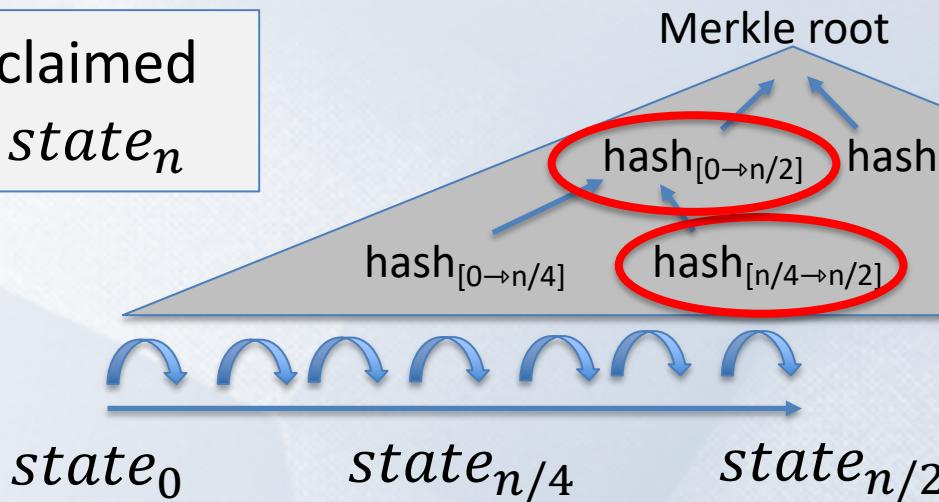
Tx list



fault claim

different  
 $state'_n$

Suppose  $state_{n/4} = state'_{n/4}$



Coordinator sends  $hash_{[n/4 \rightarrow n/2]}$  to L1  
Alice sends "right" to L1

# Fault Proof game: binary search



coordinator

claimed  
 $state_n$

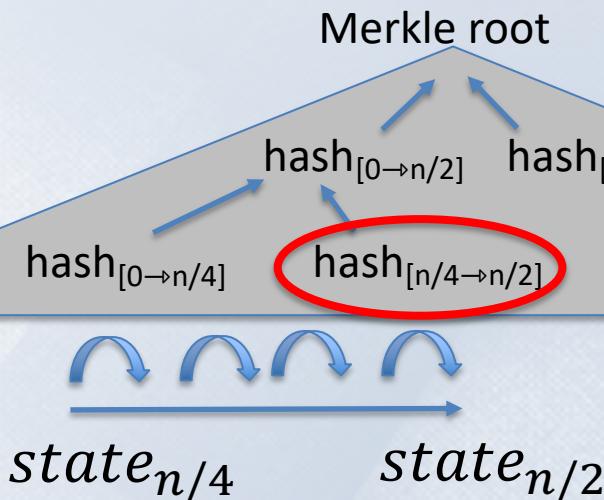
pre-root

Tx list



fault claim

different  
 $state'_n$

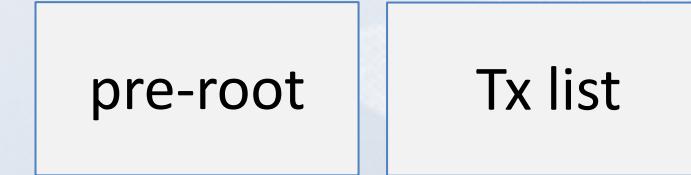


Coordinator sends  $hash_{[n/4 \rightarrow n/2]}$  to L1  
Alice sends "right" to L1

# Fault Proof game: binary search



coordinator



After  $\log_2 n$  rounds:

- L1 has  $state_i$  and  $state_{i+1}$  from coordinator
- $state_i = state'_i$  and  $state_{i+1} \neq state'_{i+1}$

or game times out because one player defects

⇒ Now L1 can verify fault proof by checking one computation step!

# A simpler alternative



pre-root

Tx list



claimed  
 $state_n$

different  
 $state'_n$

- Alice submits to L1 contract a SNARK proof that  $state_n$  is invalid
- L1 verifies SNARK, and if valid, slashes coordinator
  - ⇒ SNARK is only needed in a rare fault event

# Some difficulties with optimistic approach

- (1) Transactions only settle after 7 days (after fault window expires)
- Alice needs to wait 7 days to withdraw funds from Rollup  
(Rollup contract will only send her the funds after 7 days)

[For fungible tokens, a 3<sup>rd</sup> party can advance the funds to Alice after checking validity of Alice's withdraw Tx. Does not apply to non-fungible tokens.]

- (2) Suppose a successful fault proof 4 days after batch is posted  
⇒ all subsequent Tx need to be reprocessed

# The end result

Can easily port any smart contract to an optimistic Rollup

- The Rollup EVM can be enhanced with new features (opcodes)

High Tx throughput: in principle, up to 4000 tx/s

- No need for special hardware at the coordinator

Anyone can act as a coordinator and a verifier

Downside: 7 day finality delay

... ok, so coordinator cannot submit invalid Tx.

## Problem 2: centralized coordinator, what if it stops providing service?

Solution: setup a new coordinator

... but need the latest Rollup state

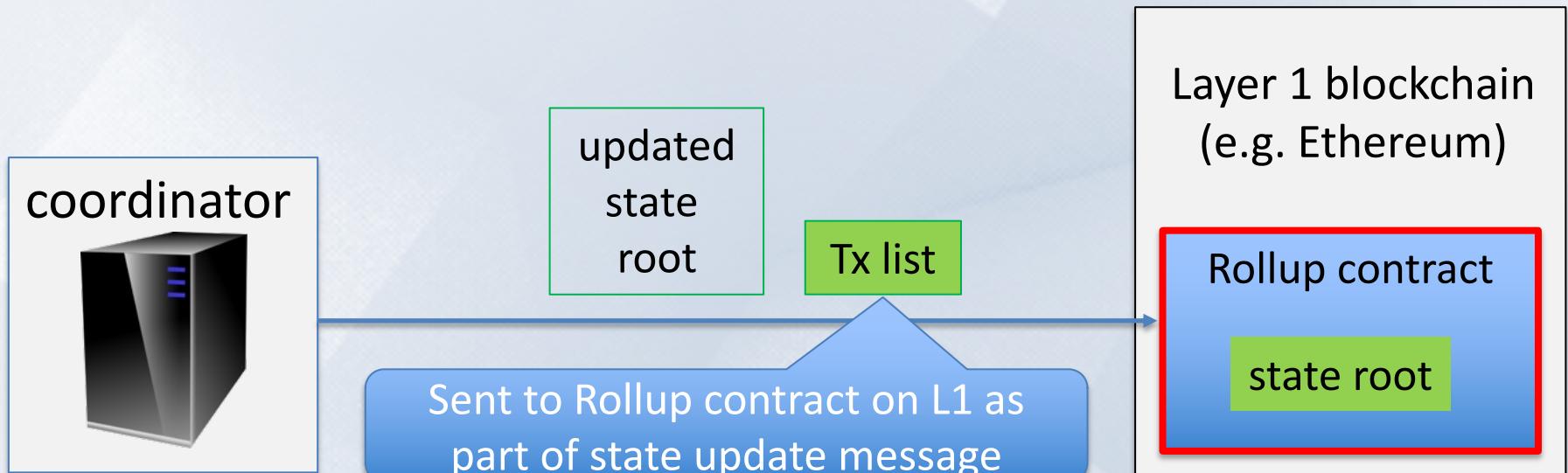
Where to get state?? The **data availability problem**



# Ensuring Rollup state is always available

The definition of a Rollup:

Rollup state can always be reconstructed from data on the L1 chain



# Ensuring Rollup state is always available

To reconstruct current Rollup state:

- Read all Rollup update messages and re-execute Tx.  
⇒ anyone can become a coordinator
- Rollups use L1 for data storage

## What to store?

- For zk-Rollup: send Tx summary to L1, without user signatures  
(SNARK proof proves validity of signatures)
- For optimistic: need to send Tx summary \*and\* signatures to L1

# Ensuring Rollup state is always available

The downside: **expensive**

- Tx list is sent as calldata: 16 gas per non-zero byte  
(EIP-4844: store Tx list as a cheap blob)

Can we do better?

# Data Availability Committee (DAC)

To further reduce Tx fees:

- **Store L2 state root** (small) on the L1 chain
- **Store Tx data** (large) with a Data Availability Committee (DAC):
  - a set of nodes trusted to keep the data available
  - cheaper than storage on L1
  - L1 accepts an update only if all DAC members sign it
    - ⇒ ensures that all DAC members accepted Tx data

Setting up a new coordinator depends on availability of the DAC

# Validium

Validium: an L2 using a DAC and validity proofs (SNARKs)

- Well suited for lower value assets.
- Potential privacy benefits ... only DAC members see Tx data

An example: StarkEx uses a five member DAC

- Users can choose between Validium or Rollup modes

(Tx data off-L1-chain vs. Tx data on-L1-chain)

cheaper Tx fees,  
but only secure as DAC

More expensive Tx,  
but same as L1 security

# Summary: types of L2

Scaling the blockchain: Payment channels and Rollups (L2 scaling)

security	SNARK validity proofs	Fraud proofs
availability	Tx data on L1 chain	zkRollup  optimistic Rollup, 7 day finality
	Tx data in a DAC	Validium (reduced fees, but higher risk)  "Plasma"

# Volume of some L2 systems

	<u>Tx Volume/day</u>	<u>average fee/tx</u>	(on Nov. 27, 2023)
• Ethereum:	1077K Tx	7.8 USD/Tx	
• Arbitrum:	676K Tx	0.30 USD/Tx	(optimistic Rollup)
• Optimism:	284K Tx	0.26 USD/Tx	(optimistic Rollup)
• StarkNet:	537K Tx	0.56 USD/Tx	(zkRollup)

# Can coordinator censor a Tx?

What if coordinators refuse to process a Tx?

What to do? One option:

- enduser can post Tx directly to the L1 Rollup contract
- The L1 Rollup contract will then refuse to accept updates until an update includes that Tx
  - ⇒ censorship will cause the entire Rollup to freeze

# SNARK recursion

Layer 3 and beyond ...



# SNARK recursion

Two level recursion: **proving knowledge of a proof**

Use  $V'(vp', x, \pi')$  to verify final proof  $\pi'$

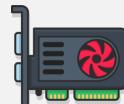
public:  $x$

witness:  $w$

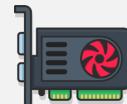
proves  $P$  knows  $w$  s.t.  
 $C(x, w) = 0$

proves  $P'$  knows  $\pi$  s.t.  
 $V(vp, x, \pi) = yes$

SNARK  
prover  $P$



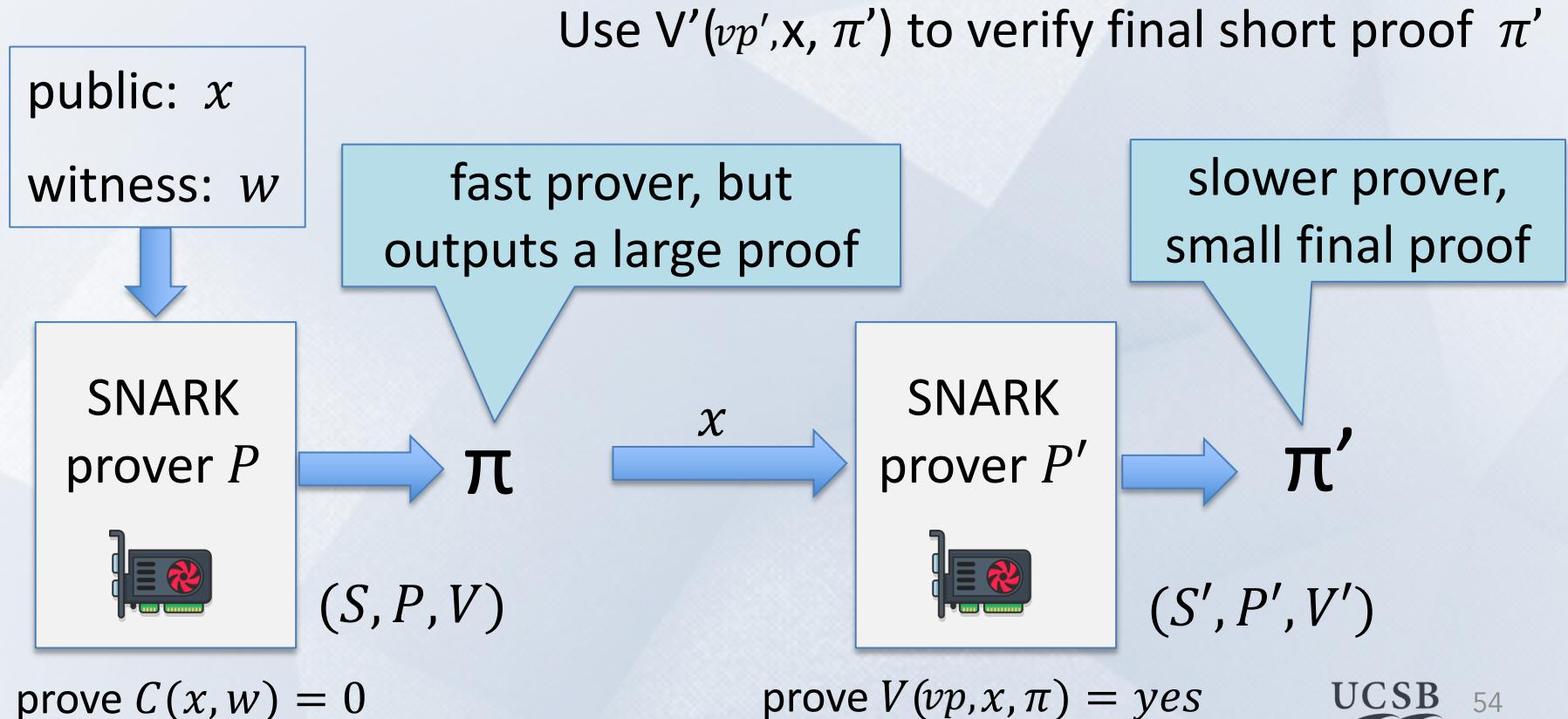
SNARK  
prover  $P'$



$(S, P, V)$

$(S', P', V')$

# Application 1: proof compression



# Application 2: Layer three and beyond

Alice:  
state

Bob:  
state

...

L3 Rollup state (any VM)

L3 Rollup contract:  
state root

Alice:  
**2 ETH, 1 DAI**

Bob:  
**5 ETH, 2 DAI**

...

L2 Rollup state

Alice:  
state

Uniswap:  
state

Rollup contract:  
**7 ETH, 3 DAI, root**

...

Layer-1 blockchain (L1)

# Layer three and beyond

One L2 coordinator can support many L3s

- each L3 can run a custom VM with its own features
- L3 chains can communicate with each other through L2

Each L3 coordinator submits Tx list and SNARK proof to L2

- L2 coordinator: collects batch of proofs,
  - builds a proof  $\pi$  that it has a batch of valid proofs, and
  - submits the single proof  $\pi$  and updated root to L1 chain.

# END OF LECTURE

Next lecture: final topics

