Recursive SNARKs

Dan Boneh
... but first, more on Rollups
Review: Rollup core idea

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

updated Rollup state root, and Tx list

Rollup coordinator

Tx_A

Tx_B

Tx_C

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

current world state
(Rollup state Merkle root)

(updated Rollup state root)

(Tx list)
### The two parts of Rollup

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

<table>
<thead>
<tr>
<th>Alice: state</th>
<th>Bob: state</th>
<th>Rollup contract:</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 ETH, 1 DAI</td>
<td>3 ETH, 2 DAI</td>
<td>7 ETH, 3 DAI, root</td>
</tr>
</tbody>
</table>

**coordinator (a server): Rollup state (L2)**

- Alice: 4 ETH, 1 DAI
- Bob: 3 ETH, 2 DAI
- ...
How to send Tx to the coordinator

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

(by default Metamask sends Tx to the Ethereum Mainnet RPC points)
**Problem 1:** what if coordinator is dishonest?

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

⇒ **solution:** validity proofs (zk-Rollup) or fraud proofs (opt. Rollup)

| immediate finality, high compute | 7-day finality, low compute |
An example (StarkNet -- using validity proofs)

<table>
<thead>
<tr>
<th>Block</th>
<th>Number</th>
<th>Hash</th>
<th>Status</th>
<th>Num. of Txs</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PENDING</td>
<td>PENDING</td>
<td>PENDING</td>
<td>64</td>
<td>3min</td>
</tr>
<tr>
<td>13011</td>
<td>0x0432_2380</td>
<td>ACCEPTED_ON_L2</td>
<td>82</td>
<td>8min</td>
<td></td>
</tr>
<tr>
<td>13010</td>
<td>0x0492_f0d1</td>
<td>ACCEPTED_ON_L2</td>
<td>122</td>
<td>15min</td>
<td></td>
</tr>
<tr>
<td>13009</td>
<td>0x0081_b7af</td>
<td>ACCEPTED_ON_L2</td>
<td>127</td>
<td>24min</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12868</td>
<td>0x060c_15eb</td>
<td>ACCEPTED_ON_L2</td>
<td>58</td>
<td>8h</td>
<td></td>
</tr>
<tr>
<td>12867</td>
<td>0x0654_3b0f</td>
<td>ACCEPTED_ON_L1</td>
<td>72</td>
<td>9h</td>
<td></td>
</tr>
<tr>
<td>12866</td>
<td>0x0779_57d6</td>
<td>ACCEPTED_ON_L1</td>
<td>63</td>
<td>9h</td>
<td></td>
</tr>
<tr>
<td>12865</td>
<td>0x06ae_943f</td>
<td>ACCEPTED_ON_L1</td>
<td>97</td>
<td>9h</td>
<td></td>
</tr>
</tbody>
</table>

Tx posted on L1 (Ethereum) about every eight hours

Source: starkscan.co
An example (Optimism -- using fraud proofs)

<table>
<thead>
<tr>
<th>Txn Batch</th>
<th>Age</th>
<th>Batch Size</th>
<th>L1 Txn Hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>328411</td>
<td>2 mins ago</td>
<td>109</td>
<td>0xbb358889959cf83413...</td>
</tr>
<tr>
<td>328410</td>
<td>2 mins ago</td>
<td>91</td>
<td>0x8398475c9b7179ebfe...</td>
</tr>
<tr>
<td>328409</td>
<td>3 mins ago</td>
<td>85</td>
<td>0x3264a772e220beca85...</td>
</tr>
<tr>
<td>328408</td>
<td>3 mins ago</td>
<td>106</td>
<td>0xa92bd044f576a87c1...</td>
</tr>
<tr>
<td>328407</td>
<td>4 mins ago</td>
<td>101</td>
<td>0x302cda229ed83d570e...</td>
</tr>
<tr>
<td>328406</td>
<td>4 mins ago</td>
<td>79</td>
<td>0x0f205018c4a289af9d7...</td>
</tr>
<tr>
<td>328405</td>
<td>5 mins ago</td>
<td>113</td>
<td>0xedbe2e0706cb06c3cb...</td>
</tr>
<tr>
<td>328404</td>
<td>5 mins ago</td>
<td>120</td>
<td>0xffaa82d2f006f519a892...</td>
</tr>
</tbody>
</table>

Shows batch posted on L1 (Ethereum)

Source: optimistic.etherscan.io
Problem 1: what if coordinator is dishonest?
• It could steal funds from the Rollup contract
• It could issue fake Tx on behalf of users
⇒ solution: validity proofs (zk-Rollup) or fraud proofs (opt. Rollup)

Problem 2: what if coordinator stops providing service?
• If Rollup state is lost, how can we initialize a new coordinator?
Ensuring Rollup state is always available

The definition of a Rollup:
Rollup state can always be reconstructed from data on the L1 chain.

Layer 1 blockchain (e.g. Ethereum)
- Rollup contract
  - state root
- Tx list

Coordinator
- Sent to Rollup contract on L1 as part of state update message
- Updated state root
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 signatures
  (SNARK proof proves validity of signatures)
• For optimistic: need to send Tx summary *and* signatures to L1

... but note EIP-4444
Ensuring Rollup state is always available

The downside: **expensive**
- Tx list is sent as calldata: 16 gas per non-zero byte
  
  (EIP-4488 aims to support Rollups by reducing to 3 gas/byte)

In practice:
- Optimistic Rollups fee/Tx: 3-8 times lower than Ethereum L1
- zk-Rollup fee/Tx: 40-100 times lower than Ethereum L1

Can we do even better?
To further reduce Tx fees:

• **Store L2 state root** (small) on the L1 chain

• **Store Tx data** (large) with a Data Availability Committee (**DAC**):
  • comprises 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
    \[\Rightarrow\text{ensures that all DAC members accepted Tx data}\]

Setting up a new coordinator depends on availability of the DAC
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)

<table>
<thead>
<tr>
<th>security</th>
<th>SNARK validity proofs</th>
<th>Fraud proofs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tx data on L1 chain</td>
<td>zkRollup</td>
<td>optimistic Rollup, 7-day finality</td>
</tr>
<tr>
<td>Tx data in a DAC</td>
<td>Validium (reduced fees, but higher risk)</td>
<td>&quot;Plasma&quot;</td>
</tr>
<tr>
<td>System</td>
<td>Tx Volume/day</td>
<td>Average Fee/tx</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Ethereum</td>
<td>1013K Tx</td>
<td>2.71 USD/Tx</td>
</tr>
<tr>
<td>Arbitrum</td>
<td>345K Tx</td>
<td>0.08 USD/Tx</td>
</tr>
<tr>
<td>Optimism</td>
<td>303K Tx</td>
<td>0.13 USD/Tx</td>
</tr>
<tr>
<td>StarkNet</td>
<td>14K Tx</td>
<td>0.22 USD/Tx</td>
</tr>
</tbody>
</table>
Can coordinator censor a Tx?

What if coordinators refuse to process my 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 from a coordinator 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**

public: $x$

witness: $w$

proves $P$ knows $w$ s.t.

$C(x, w) = 0$

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

proves $P'$ knows $\pi$ s.t.

$V(vp, x, \pi) = yes$

(S, P, V) \rightarrow \pi \xrightarrow{x} \pi' \rightarrow (S', P', V')
Application 1: proof compression

public: \( x \)

witness: \( w \)

SNARK prover \( P \)

\( (S, P, V) \)

prove \( C(x, w) = 0 \)

fast prover, but outputs a large proof

\( \pi \)

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

slower prover, small final proof

SNARK prover \( P' \)

\( (S', P', V') \)

prove \( V(vp, x, \pi) = yes \)
Application 2: Layer three and beyond

L2 Rollup state

Alice: state
Bob: state

L3 Rollup contract: state root

Alice: 2 ETH, 1 DAI
Bob: 5 ETH, 2 DAI

L3 Rollup state (any VM)

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.

⇒ Scaling factor 100 × 100
Application 3: L2 with private Tx (simplified)

Only Alice knows her own state $a$ and $r_a$.

- Coordinator does not know account balances

(only Alice knows her committed account balances)

L2 Rollup state: hidden balances

Alice: $h_a = H(\text{state}_a, r_a)$

Bob: $h_b = H(\text{state}_b, r_b)$
Alice want to pay Bob 2 ETH: \[ Tx: \ [A \to B: 2 \text{ ETH}, \ \text{sig}_A \] \]

- compute updated state’_a and send \( Tx \) to Bob (privately)
- choose random \( r'_a \) and set \( h'_a \leftarrow H(\text{state'}_a, r'_a) \)
- build proof \( \pi_a \) that \( h'_a \) is a valid update to Alice’s state
- Send \( (h'_a, \pi_a) \) to L2 coordinator

Alice: \[ h_a = H(\text{state}_a, r_a) \]

[state commitment]

Bob: \[ h_b = H(\text{state}_b, r_b) \]

[state commitment]
Bob receives $Tx = [A \rightarrow B: 2 \text{ ETH}, \text{ sig}_A]$ from Alice

- compute updated state $'b$
- choose random $r'_b$ and set $h'_b \leftarrow H(\text{state}'_b, r'_b)$
- build proof $\pi_b$ that $h'_b$ is a valid update to Bob’s state
- Send $(h'_b, \pi_b)$ to L2 coordinator

Application 3: L2 with private Tx (simplified)

Alice: $h_a = H(\text{state}_a, r_a)$

Bob: $h_b = H(\text{state}_b, r_b)$

[state commitment]
Collect a batch of transactions from users \( \{(h'_i, \pi_i)\} \):

- Update Merkle leaves to new committed states
- build a proof \( \pi' \) that it has a batch of valid proofs for a consistent set of transactions, and
- submit a single proof \( \pi' \) and updated root to L1 chain.

**Application 3: L2 with private Tx (simplified)**

Alice:  \( h'_a = H(\text{state}'_a, r'_a) \)  

Bob:  \( h'_b = H(\text{state}'_b, r'_b) \)  

[state commitment]  

[proof \( \pi' \), new root, Tx List]
Only Alice knows her balance. Only Bob knows his balance.
... they can transact without revealing amounts
... also transact with a public contract (public code and state).

Note: as described, no privacy for Alice when withdrawing from L2

Alice: \( h'_a = H(\text{state'}_a, r'_a) \)  
Bob: \( h'_b = H(\text{state'}_b, r'_b) \)

\[ h'_a = H(\text{state'}_a, r'_a) \]
\[ h'_b = H(\text{state'}_b, r'_b) \]

proof \( \pi' \), new root, Tx List
Danger: if Alice loses here $r_a$, she loses access to her funds on L2

Alice: $h'_a = H(\text{state}'_a, r'_a)$

Bob: $h'_b = H(\text{state}'_b, r'_b)$

[state commitment]

proof $\pi'$, new root, Tx List
Final ZK topics
Commercial interest in SNARKs

Many more building applications on top ...
In this setup, a single reliable PC can monitor the operation of a herd of supercomputers working with unreliable software.

“Checking Computations in Polylogarithmic Time”
Why so much commercial interest?

Babai-Fortnow-Levin-Szegedy 1991:

*a slow and expensive computer*

In this setup, a single reliable PC can monitor the operation of a herd of supercomputers working with unreliable software.

“Checking Computations in Polylogarithmic Time”
Why so much commercial interest?

Babai-Fortnow-Levin-Szegedy 1991:

*an L1 blockchain*

In this setup, a single reliable PC can monitor the operation of a herd of supercomputers working with unreliable software. *coordinators*

“Checking Computations in Polylogarithmic Time”
Blockchains drive the development of SNARKs:

zkRollup, zkBridge, zkCreditScore, zkTaxes, ...

... but **many** non-blockchain applications
Ukraine conflict: Many misleading images have been shared online.

Fact-checking videos and pictures from Ukraine

Since Russia's invasion on 24 February, and pictures of attacks on Ukrainian cities have appeared online. The BBC's Alistair McCauley explains how to spot fake images.
C2PA: a standard for content provenance

Sony Unlocks In-Camera Forgery-Proof Technology

04 Aug, 2022

embedded certified signing key $sk$

verify metadata by checking sig

C2PA

location

timestamp

signature
A problem: post-processing

Newspapers often process the photos before publishing:

- Resize (1500 \times 1000), Crop, Grayscale (AP lists allowed ops)

The problem: laptop cannot verify signature on processed photo

C2PA “solution”: editing software will sign processed photo to certify edits
A solution using ZK proofs (SNARKs) (with T. Datta)

Editing software attaches a proof $\pi$ to photo that:

I know a triple $(\text{Orig}, \text{Ops}, \text{Sig})$ such that

1. $\text{Sig}$ is a valid C2PA signature on $\text{Orig}$
2. photo is the result of applying $\text{Ops}$ to $\text{Orig}$
3. metadata(photo) = metadata($\text{Orig}$)

$\Rightarrow$ Laptop verifies $\pi$ and shows metadata to user
Performance

Proof size: 200-400 bytes. Verification time: 2 ms. (in browser)

Proof generation time by newspaper:

- Resize \((3000 \times 3000 \rightarrow 1500 \times 1500)\): 84 sec.
- Crop \((3000 \times 3000 \rightarrow 1500 \times 1500)\): 60 sec.

What about video??

See also: PhotoProof by Naveh & Tromer (2016)
The future: a market for ZK provers

Anyone with a GPU will be paid to create ZK proofs

market

selects provers and distributes rewards

prover 1
\[ \pi_1 \]

prover 2
\[ \pi_2 \]

prover 3
\[ \pi \]
• **Lots** more to work on:
  • **Better provers**: faster, lower memory footprint, shorter proofs, quantum resistant, no trusted setup, distributed witness.
  • **New applications** for SNARKs and zk-SNARKs
Recap: current application areas

1. Finance (DeFi):
   • new financial instruments, exchanges, lending, ...

2. Managing digital assets (NFTs)
   • Assured provenance

3. Decentralized organizations (DAOs):
   • DAOs for investment, for donations, for collecting art, etc.
   • Governance: group decision making
What is a DAO?

- A Dapp deployed on-chain at a specific address
- Anyone (globally) can send funds to DAO treasury
- Anyone can submit a proposal to DAO
  - participants vote
  - approved → proposal executes

(SafeSnap: trustless on-chain execution of off-chain votes)
Examples of DAOs

There are currently about 6500 DAOs managed on Snapshot

• **Collector DAOs**: PleasrDAO, flamingoDAO, ConstitutionDAO, ...
  (see art collection at [https://gallery.so/pleasrdao](https://gallery.so/pleasrdao))

PleasrDAO: 103 members.

• Manages a treasury, has full time employees.
• Deliberations over what to acquire over telegram.
Examples of DAOs

There are currently about 6500 DAOs managed on Snapshot

• Collector DAOs: PleasrDAO, flamingoDAO, ConstitutionDAO, ...

• Charity DAO: gitcoin (42K members), ...

Proposal ID 21: This proposal looks to ratify the allocation of 30,000 GTC from the Community Treasury to the MMM workstream.
Examples of DAOs

There are currently about 6500 DAOs managed on Snapshot

- **Collector DAOs:** PleasrDAO, flamingoDAO, ConstitutionDAO, ...
- **Charity DAO:** gitcoin, ...
- **Protocol DAO:** manages operation of a specific protocol
  - Uniswap DAO (29K members), Compound (4K members), ...
- **Social DAO:** FWB, ...
- **Investment DAO:** many
## Example: Uniswap proposals

<table>
<thead>
<tr>
<th>Proposal</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add 1 Basis Point Fee Tier</td>
<td>executed</td>
</tr>
<tr>
<td>TLDR: Uniswap should add a 1bps fee tier with 1 tick spacing. This change is straightforward from a</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Upgrade Governance Contract to Compound's Governor Bravo</td>
<td>executed</td>
</tr>
<tr>
<td>Previous Discussion: <a href="https://gov.uniswap.org/t/temperature-check-upgrade-gove...">Temperature Check</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Community-Enabled Analytics</td>
<td>canceled</td>
</tr>
<tr>
<td><em>Past discussion:</em> <a href="https://gov.uniswap.org/t/temperature-check-larger-grant-pro">Temperature Check</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>DeFi Education Fund</td>
<td>executed</td>
</tr>
<tr>
<td># (Previously known as: DeFi Political Defense Fund) Past discussion: [Temperature Check ](http</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Reduce the UNI proposal submission threshold to 2.5M</td>
<td>executed</td>
</tr>
<tr>
<td>This proposal lowers the UNI proposal submission threshold from 10M UNI to 2.5M UNI. Uniswap's gove</td>
<td></td>
</tr>
</tbody>
</table>
How to build a DAO

Three key decisions:

• What is the community for the DAO?

• How is membership managed?
  Many available tools: Syndicate, Juicebox, Colony, ...
  can anyone join, or does the community vote?

• How to do governance? What is controlled by governance?
Many DAO governance experiments

Who can vote?   How to vote?   What voting mechanism?

Lightspeed Democracy: What web3 organizations can learn from the history of governance

by Andrew Hall and Porter Smith
June 29, 2022

DAOs: a platform for experimenting with governance mechanisms
Governance methods

One token one vote: (most common)
- Members receive tokens based on their contribution.
- Everyone can vote.

Frequently implemented using one of OpenZeppelin’s Governor contracts (Solidity code)

```solidity
_castVote(proposalID, voter, support, reason);
```

Problem: direct democracy does not scale.
Poor participation rate

For all but one project: participation rate < 5%

What to do? delegation
Supported in Governor contract

Voting rate = # Tokens voted / Total tokens in existence
These 5 DAOs sampled for convenience
Source: Boneh and Hall (super preliminary ongoing research)
Delegation example: element

≈300 addresses delegated tokens to this address
Private DAO treasury

2021: an auction for a physical copy of the constitution. (Sotheby's auction house)

**ConstitutionDAO:**

- Formed in Nov. 2021 to participate in auction.
- Raised $46.3M from about 20K participants worldwide
- Lost to another bidder who bid $43M

bidder knew that ConstitutionDAO could not outbid it

How to participate in an auction when everyone knows your treasury??
The design:
One DAO platform manages many DAOs:
- a single Ethereum contract (e.g., JuiceBox)

**DAO manager:** sets up a DAO by publishing a DAO public key (pk)

**Contributor:** sends funds to platform with a “blinded DAO-pk”

Contract records contribution
  - ⇒ an observer learns nothing about which DAO received the funds
  - ⇒ only learns total amount stored on the platform as a whole

DAO manager can later use its secret key to claim funds sent to its DAO
Many other DAO privacy questions …

- **Private DAO participation**: keep participant list private
- **Private voting**: keep who voted how on each proposal private
- **Private delegations**

... while complying with all relevant laws.

Some of these questions are solved by general privacy platforms such as Aztec, Aleo, and others.
Next lecture: MEV and bridging