Scaling II: Rollup

Benedikt Bünz
Lightning network

- Low TX fees
- Instant payments
- Routing through intermediaries
Downsides of Payment/State Channels

- Everyone needs to be online
  - Mitigated by watchtowers
  - Hubs need to be online
- Capital is locked up
  - Funds in one channel can’t be used in different channel
  - If network is separated transactions are not possible
- Only Peer to Peer payments
  - No multi party contracts channels
- TX to fund/close
Blockchain Layers

Layer 3: user facing tools  (cloud servers)
Layer 2: applications  (DAPPs, smart contracts)
Layer 1.5: compute layer  (blockchain computer)
Layer 1: consensus layer

Bitcoin/Ethereum combine ordering (layer 1) and verification (1.5)
What if we can outsource verification? Makes consensus cheaper
Idea: Aggregate Transactions

- Payment channels move more transactions offchain
- Idea: Combine Transaction, Rollup Server verifies

TX1: A->B 5ETH
TX2: C->D 2ETH
TX3: D->B 1ETH

TX Agg: TX1, TX2, TX3

Server (untrusted)
Blockchain
Smaller than sum of TX
Recap: The Ethereum blockchain
Recap: Merkle tree  
(Merkle 1989)

Goal:
• commit to list $S$
• Later prove $S[i] = m_i$

To prove $S[4] = m_4$, proof $\pi = (m_3, y_1, y_6)$

length of $\pi$: $\log_2 |S|$
Recap State Commitment

Every contract has an associated storage array $S[]$:

$S[0], S[1], \ldots, S[2^{256}-1]$: each cell holds 32 bytes, init to 0.

Account storage root: **Merkle Patricia Tree hash** of $S[]$

- Cannot compute full Merkle tree hash: $2^{256}$ leaves

|--------------|--------------|--------------|--------------|

<table>
<thead>
<tr>
<th>0</th>
<th>0, a</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10, d</td>
</tr>
<tr>
<td>0</td>
<td>⊥, b</td>
</tr>
<tr>
<td>1</td>
<td>⊥, c</td>
</tr>
</tbody>
</table>

Time to compute root hash: $\leq 2 \times |S|$

$|S| = \# \text{non-zero cells}$
Merke (Patricia) Tree Proofs

- Logarithmic in tree height
- Given proof for $i \rightarrow$ Possible to update $S[i]$ and recompute root
- Given proof for $i$, proof for $j$ and update of $S[j]$ it’s possible to update proof for $S[i]$
- Exclusion proofs possible in Patricia Trees
Rollup State $S$

$S[A's \text{ PK}] = \{3 \text{ ETH, nonce}\}$

$S[B's \text{ PK}] = \{2 \text{ ETH, nonce}\}$

$S[C's \text{ PK}] = \{10 \text{ ETH, nonce}\}$

$S[D's \text{ PK}] = \{1 \text{ ETH, nonce}\}$
Rollup Deposit

Users

TX Deposit:

Proof that A’s PK ∉ S given root
3 ETH transfer

Rollup Smart Contract

1. Checks Proof
2. Updates root such that S[A’s PK]={3 ETH, 0}
Rollup Withdraw

Users
TX Withdraw:

Proof that S[A’s PK]={3 ETH, nonce} given root
Destination Address NewA
Signature by A

Rollup Smart Contract

1. Checks Proof
2. Checks Signature
3. Sends 3 ETH to NewA
Rollup Transfer

TX Transfer

Users

TX Transfer:

Proof that given root
S[A’s PK]={3 ETH, 0}
S[B’s PK]={2 ETH, 0}
Transfer amount 2 ETH
Signature by A

Rollup Smart Contract

1. Checks Proofs
2. Checks Signature
3. Set
   1. S[A’s PK]={1 ETH, 1}
   2. S[B’s PK]={4 ETH, 1}

Space saved but no computation
Provides Proof/SNARK that given public inputs (rootHash, key, value) it knows private inputs (path) such that function outputs true

SNARK is short/easy to check
SNARK: a **Succinct ARgument of Knowledge**

A **succinct preprocessing argument system** is a triple \((S, P, V)\):

- \(S(C) \rightarrow\) public parameters \((S_p, S_v)\)
- \(P(S_p, x, w) \rightarrow\) short proof \(\pi\); \(|\pi| = O(\log(|C|), \lambda)\)
- \(V(S_v, x, \pi)\); time(V) = \(O(|x|, \log(|C|), \lambda)\)

If \((S, P, V)\) is **succinct** and **zero-knowledge** then we say that it is a **zk-SNARK**

\(|\pi| = 500\) bytes

time(V) = 500k Gas
ZKRollup

• Merkelize Transactions
• SNARK proves that given transactions I know signatures such that state transition $S \rightarrow S'$ valid
• Publish transaction diff on chain.
• No signatures per transaction.
• 500k gas + data cost for on chain diff
**ZKRollup (Validity Rollup)**

**Coordinator does:**

1. Applies TXs to $S$ resulting in $S'$
2. Produces $\text{root}' = \text{Commit}(S')$
3. Produces SNARK $\pi$ that $\exists \text{txs}$ such that $\text{root}'$ is correct update to state $S$ committed in $\text{root}'$
ZKRollup

Users → Transactions → Server Stores $S'$ → Rollup Smart Contract

- Commitment to $S'$
- Smart Contract does:
  1. Verify $\pi$ given root and $\text{root}'$
  2. If accept then set root ← root'

Rollup Smart Contract

Smart contract still allows “manual” withdrawals
Data Availability Problem

Update must be valid!
What if Server does not reveal data?

Can’t update Merkle proofs
Can’t withdraw!
Publish diff on chain

Transactions with signatures → Server Stores \( S' \) → \( \pi \) \( txlist \) → Rollup Smart Contract

\[ \text{Txlist} = [{A \rightarrow B \ 3}, \ {C \rightarrow D \ 2}, \ {D \rightarrow B \ 1}] \]

No signatures, Sender, Receiver, Amount only in Calldata (not stored) <100 bytes per tx ~400 gas/tx, SNARK verification ~1500 gas/tx (if full)

Full Block 3600 rollup tx vs 570 normal tx (6x speedup)
zkRollup stats

- ZKRollup is cheaper than onchain tx
- Can scale to max ~300tx/s now, 1000tx/s soon
- Vs. max 40-50tx/s on mainnet
- Cost dominated by SNARK verification
  - Will get cheaper precompiles
- Finality ~ Blockchain finality (no instant transfer)
Multiple Assets

Very easy to support many assets
Simply add asset field to TX
Hardly increases SNARK complexity

Txlist= [{A-> B 3 ETH}, {C-> D 2 DAI}, {D-> B 1 BAT}]
Transaction List/Atomic Swaps

Support transaction list that are executed together
Transactions need to be signed by all senders
Can’t execute part of transaction only all together!

Enables atomic swaps: Alice swaps with Bob 3 ETH for 2 DAI
Txlist= [{A-> B 3 ETH and B-> A 2 DAI}, {D-> B 1 BAT}]
Buy 3 ETH for 5 DAI

Sell 3 ETH for 5 DAI

Exchanges match orders
Classical exchanges also store funds

Order book

give | get
---|---
Alice | 3 ETH | 5 DAI
Bob | 5 DAI | 3 ETH
Carol | 4 BAT | 10 DAI
Rolled up Exchange

Submit orders

Has orderbook
Matches orders
Rolls up transactions as atomic swaps

Txs Root, $\pi$

Root of balance tree

Exchange trusted for honest matching

Verifies TX
Rolled up Exchange v2

Submit orders
Updates orderbook tree on chain and proves correct matching

Txs Root, $\pi$
Root of balance tree
Root of orderbook tree

Verifies TX

Benefit: No trust
Downside: Every order creates rollup TX, No instant matching
Rolling up Smart Contracts

• zkRollup works best for simple transfers
• zkRollup for the EVM?
  • Roll up generic smart contract transactions?
  • Create SNARK where the circuit implements the EVM
  • More expensive on the server
  • Soon to be a reality for a subset of the EVM (zkEVM)
• Can we support smart contract rollups today?
What if we remove the SNARK?

Idea: Instead of proving correctness, prove fraud!

New Role: Validator checks correctness, provides fraud proofs
Optimistic Rollup

- Server updates transaction root
- Server puts a large bond into escrow
- If transaction update is invalid users/validators provide *fraud proof*
- Successful fraud proof means bond gets *slashed*
  - Part to validator providing proof part gets burned
- Unsuccessful fraud proof costs validator money
- How to proof fraud?
Fraud Proofs

1. Stores S agrees on root
2. Applies txlist to S to compute S’
3. Computes root” from S’
4. If root’≠root” call “Fraud”

Problem: Validator doesn’t know what’s in root’
Idea: Server and Validator find first point of disagreement

Break down computation of $S'$ into small steps, e.g. cycles on a VM
Validator does the same
Let $S_i$ be Server’s intermediate states and $S'_i$ the validator’s
Server and Validator run interactive binary search

Checks whether $S_{n/2} = S'_{n/2}$
If no disagreement in first half
Otherwise in second
Referee Delegation

Repeat protocol for $\log_2(n)$ steps
End with agreement on $S_i$ and disagreement on $S_{i+1}$ and $S'_{i+1}$

Smart Contract checks transition between $S_i$ and $S_{i+1}$ and declares winner
Problem: Checks take a long time

- $\log_2(n)$ messages (1 hash per message)
- 1 Verification step on smart contract
- If either party timeouts declares winner
- Looser gets *slashed*, Winner rewarded
- Problem: $\log_2(n) \times$timeout
- No incentive to cheat
- But: Long wait till finalization! (7 days)
Pipelined Assertions

Server can build on states before timeouts

If prior state invalid, all subsequent bonds are slashed
Pipelined Assertions

Server can claim prior state not valid and continue given this.

If no successful fraud proof then reward gets slashed.
Insurance of Rollup -> Instant Finality

- Rollup is not instant (unlike lightning)
- But if server is trusted then giving them transaction -> finality
- Idea: Use insurance to achieve finality
- Server signs insurance
- If transaction not included in next (few) blocks insurance can be used to get insurance premium
- Works for zk and optimistic rollup
- Does not work for NFTs (directly)
• Live and implemented (Optimism and Arbitrum)
• You can port arbitrary smart contracts (OVM)
• Works well if honest rollup server
  • Fraud proofs protection if malicious server
• Up to ~4000 tx/s on ETH 1.0
• Important that one independent validator exists
• 7 day finality wait
A combined view of rollups

Standard L1 chains: every miner must verify every posted Tx

Rollup server: compresses a thousand Tx into one on-chain proof (SNARK or Fraud)
Data Availability

Update must be valid!
User must know state transition
Posting state diff on chain limits rollup benefits

Can we keep the data off chain?
Off Chain Rollups (Validum and Plasma)

- Idea: Create a separate “cheap” chain for the data

User

Gets state/
Sends tx

POS Consensus chain
Stores roll up state

Post new state root/
rollup proof

Independent of #TX

Ethereum: Stores state root
Conflict Resolution: On Chain requests

- Idea: Create a separate “cheap” chain for the data

POS Consensus chain
- Stores roll up state

User
- State request
- Respond with state/
- Slashed if no response

Ethereum: Stores state root
On Chain vs Off Chain Data availability

• Off Chain is much cheaper and independent of #tx
  • Only limitation is data consensus
• Data consensus not trusted for security
  • Can’t steal your money
• Data consensus is trusted for availability
  • Can lock up your money (bribery attack)
  • Can increase fees
• Economic incentives can mitigate issues
• For high value transfer use on chain rollup for low value use off chain rollup
## 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 summary on L1 chain</td>
<td>zkRollup blockchain finality, only simple transfers (now)</td>
<td>optimistic Rollup 7 day finality Instant transfers</td>
</tr>
<tr>
<td>Tx summary off chain</td>
<td>Vallidium large #tx but vulnerable to lock up attacks</td>
<td>&quot;Plasma&quot; Largest #tx but lock up attacks and long finality</td>
</tr>
</tbody>
</table>

**availability**

- **2 by 2 rollup**: A method for scaling blockchain by leveraging off-chain transactions to improve both security and availability.
END OF LECTURE

Next lecture: Recursive SNARKs