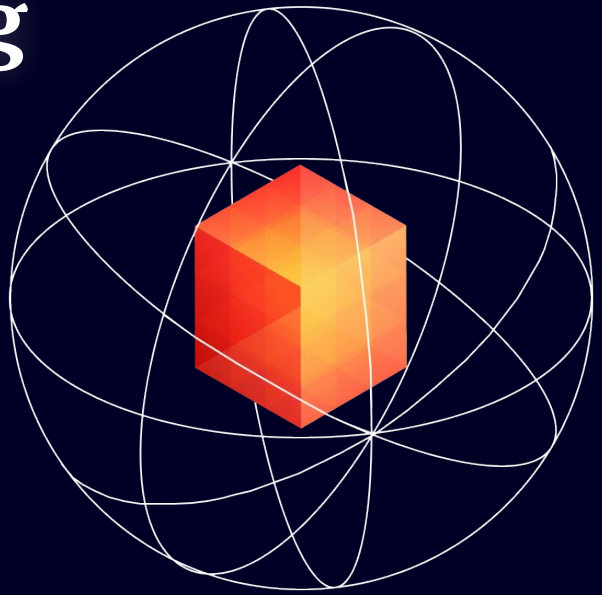
STARKWARE

Beyond Recursive Proving for Validity Rollups

Advanced Proving for Starknet

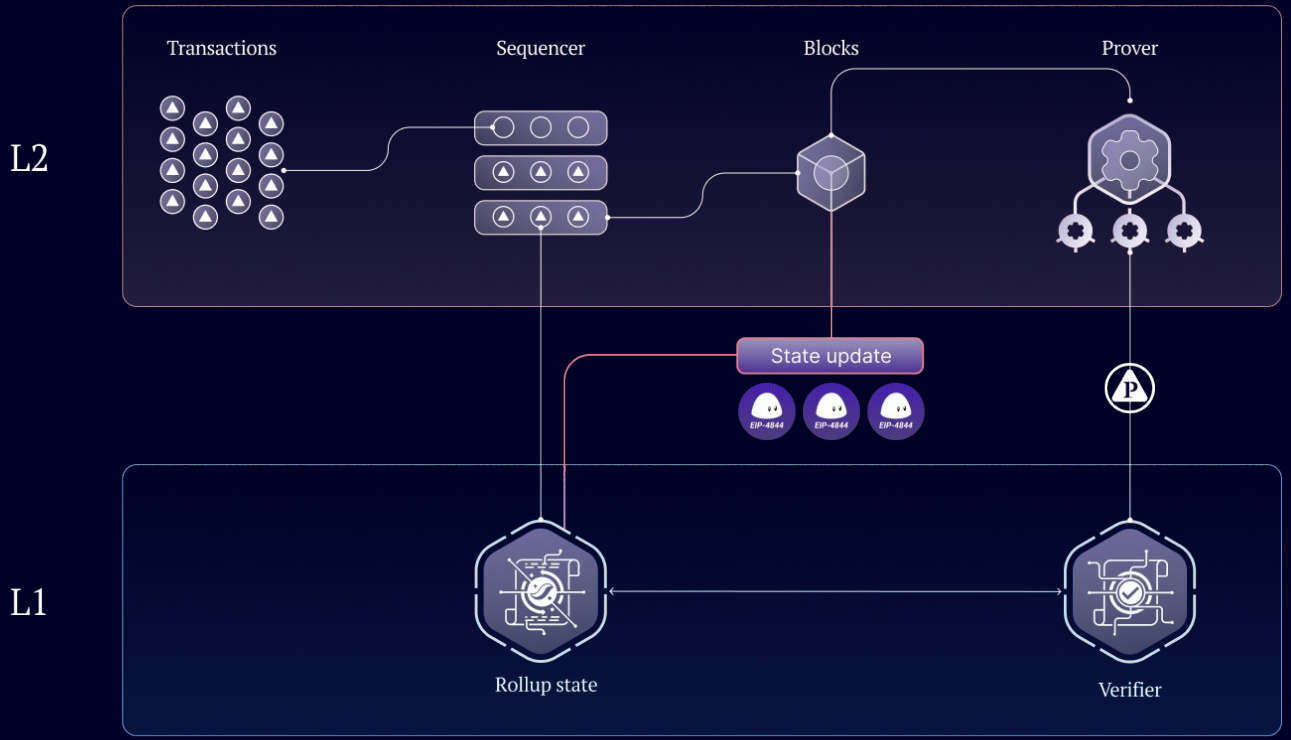


Gidi Kaempfer
VP Engineering | StarkWare

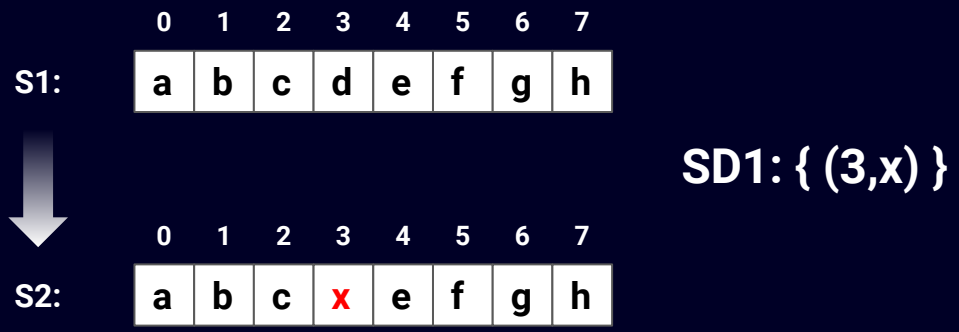
Section 1

Basics of Validity Rollups

Basics of L2 Validity Rollups



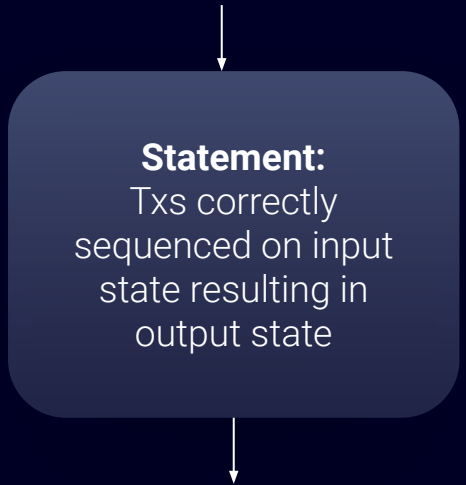
What is State Diff?



SD1: { (3,x) }

Per Block Proving

State A, txs



State B,
State Diff

Per Block on L1

- Verify block proof
- Publish state diff
- Update state (+ messaging)

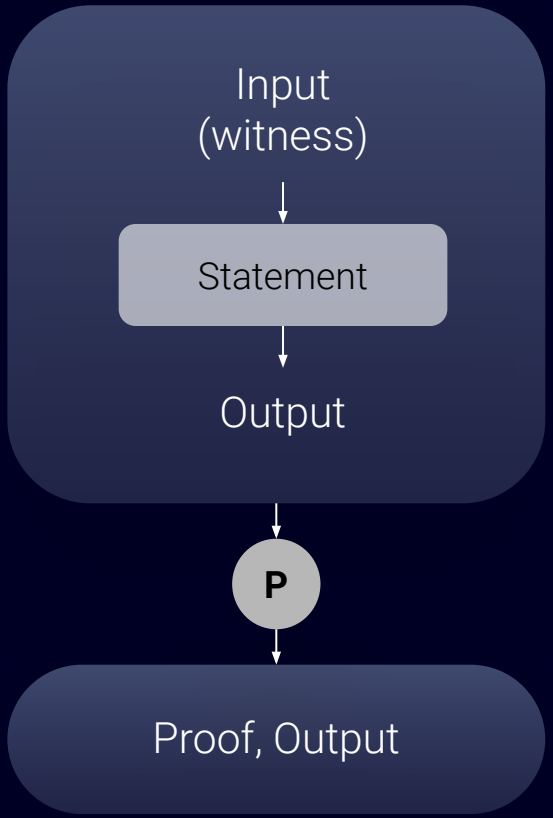
Costly unless blocks are big (can mean long block times)

Section 2

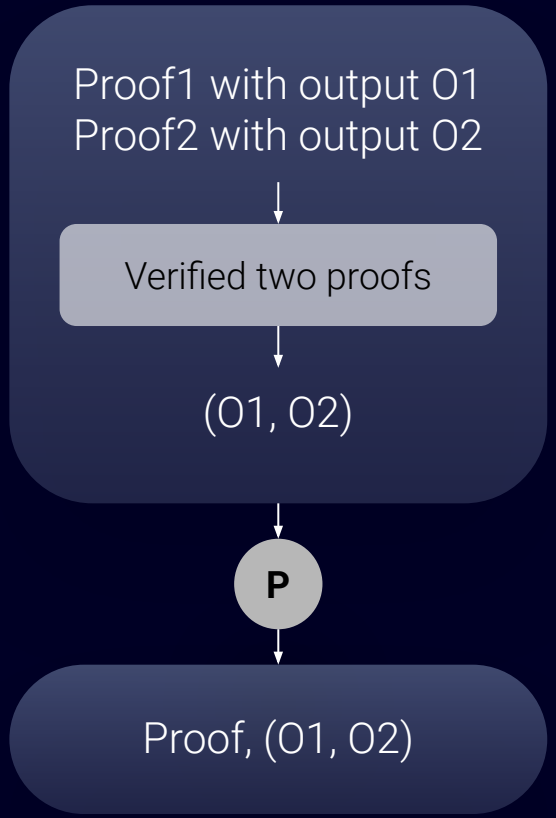
Validity Rollups with Recursive Proving

STARKWARE

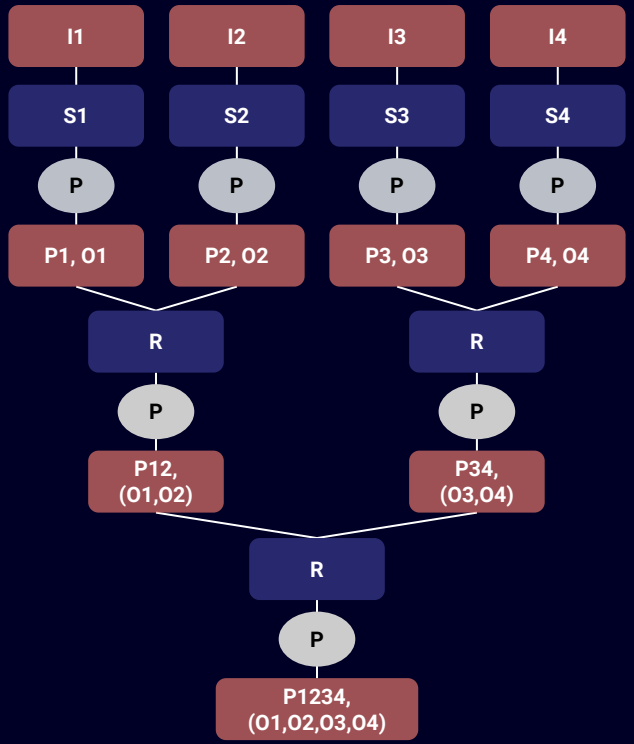
Generic Proving



Recursive Proving

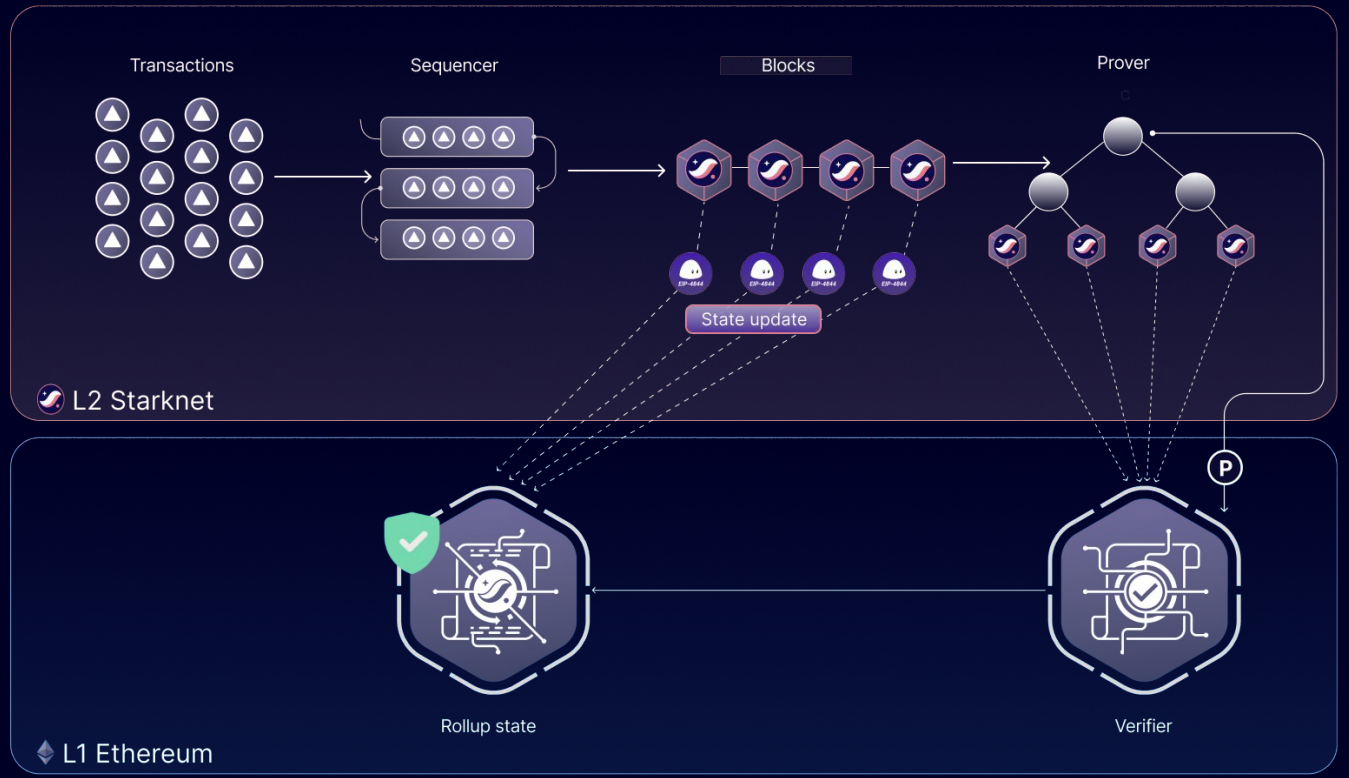


One Proof for Many Statements

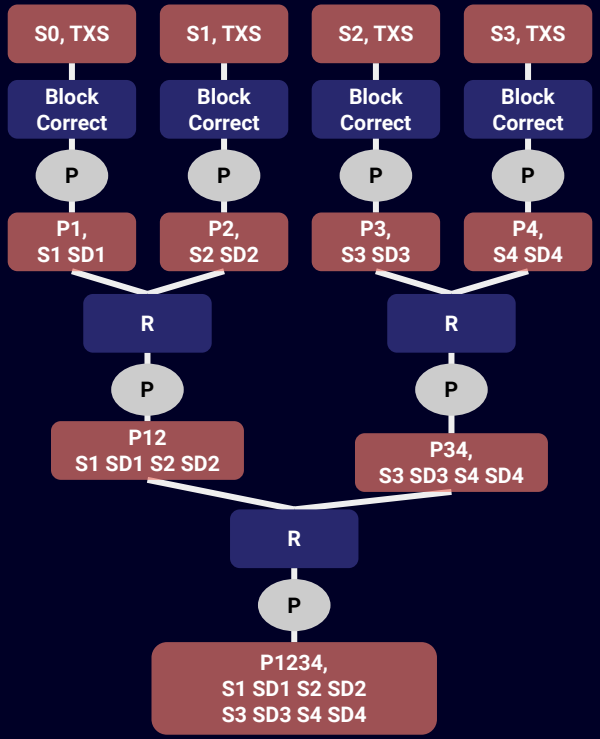


STARKWARE

L2 Validity Rollup with Recursive Proving



Validity Rollup With Recursion



Per Recursion on L1:

- Verify block proof

Per Block on L1:

- Publish state diff
- Update state $S_i \rightarrow S_{i+1}$ (+ messaging)

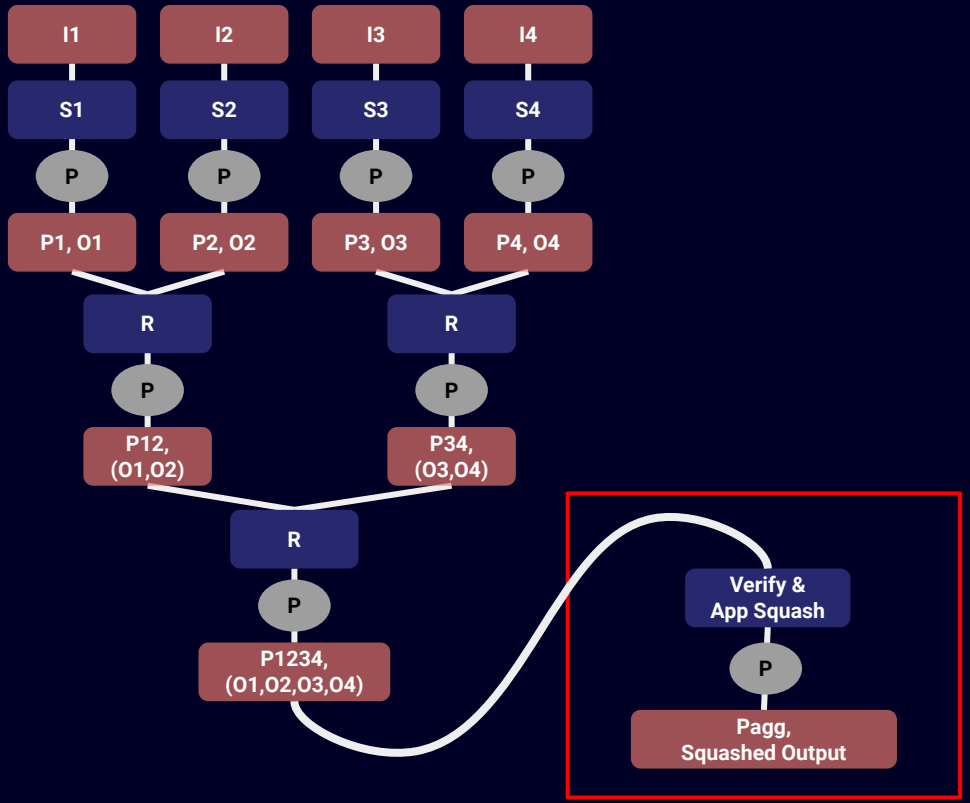
Less Costly unless blocks are very small (e.g., we want fast finality)

We would like to optimize per block updates

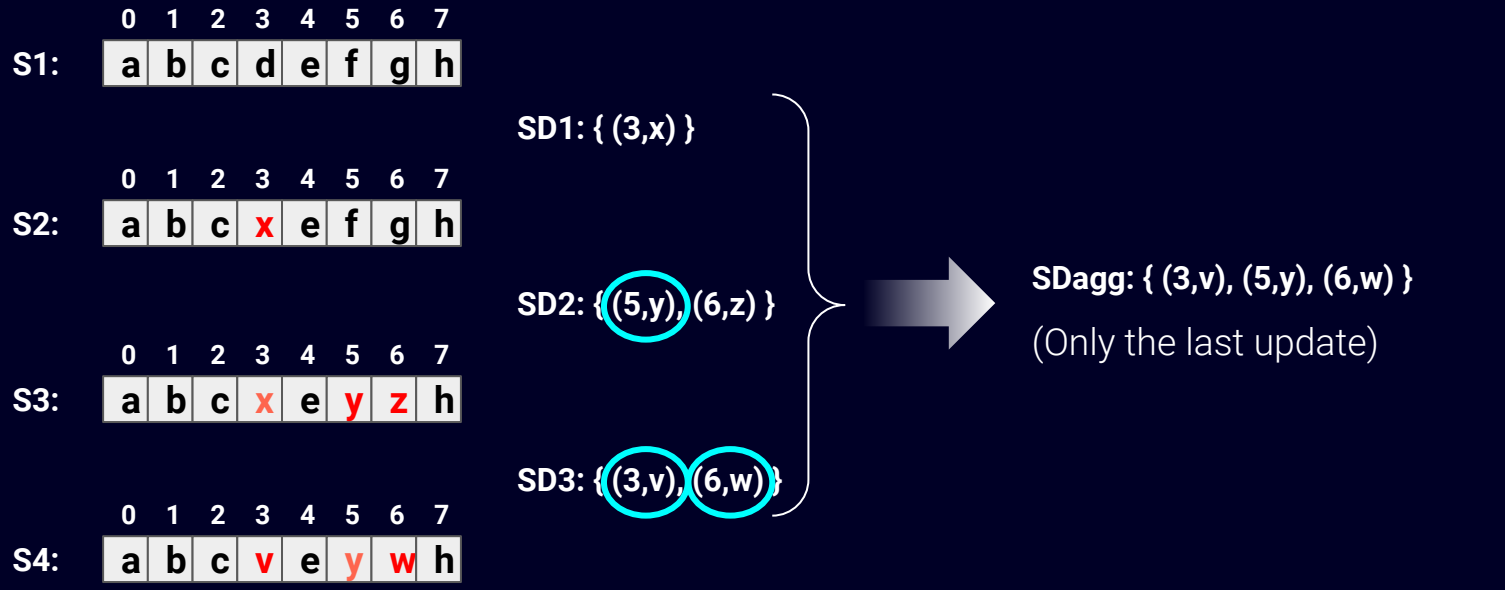
Section 3

Validity Rollups with Applicative Recursion

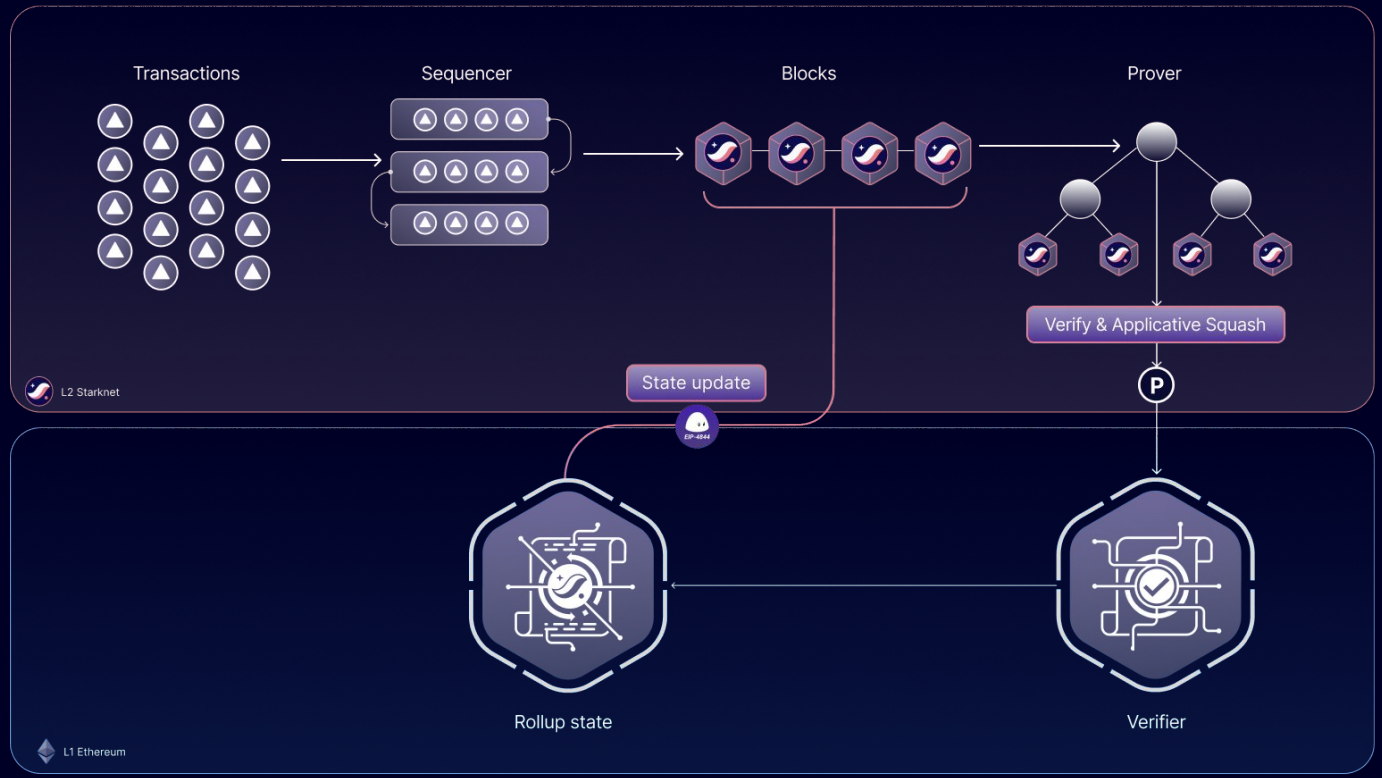
One Proof for Many Statements + Applicative Squash



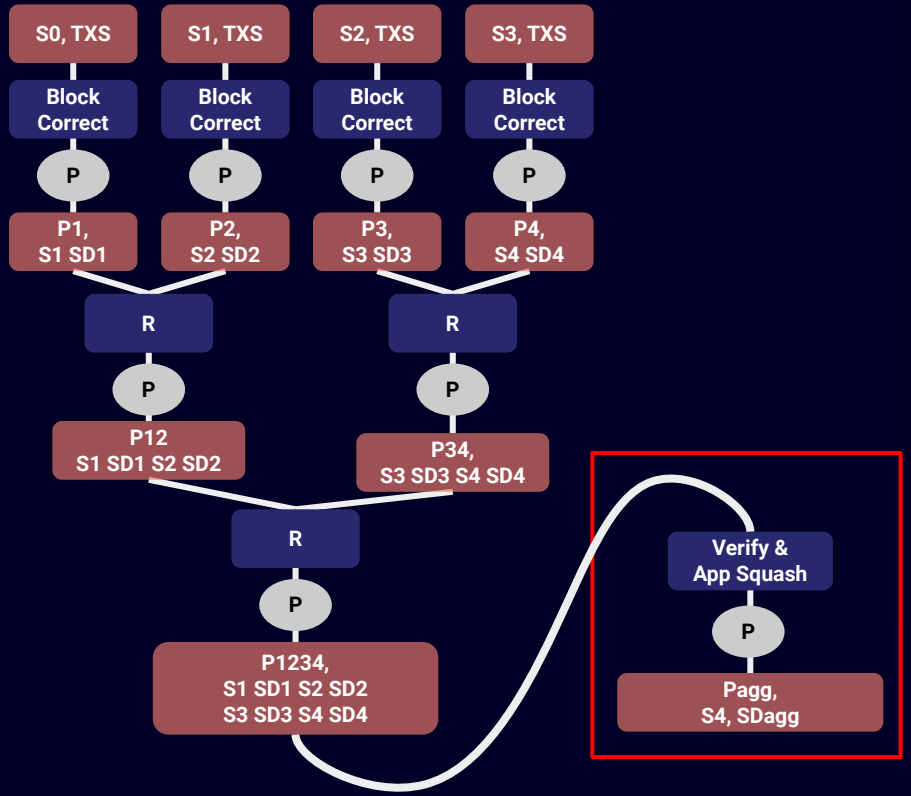
State Diff Aggregation



L2 Validity Rollup with Applicative Recursion



Validity Rollup With Applicative Recursion



Per Recursion on L1:

- Verify block proof (Pagg)

Per Recursion on L1:

- Publish state diff
- Update final state directly (+ messaging)

The more blocks per recursion, the lower the L1 cost
L2 finality can be very fast!

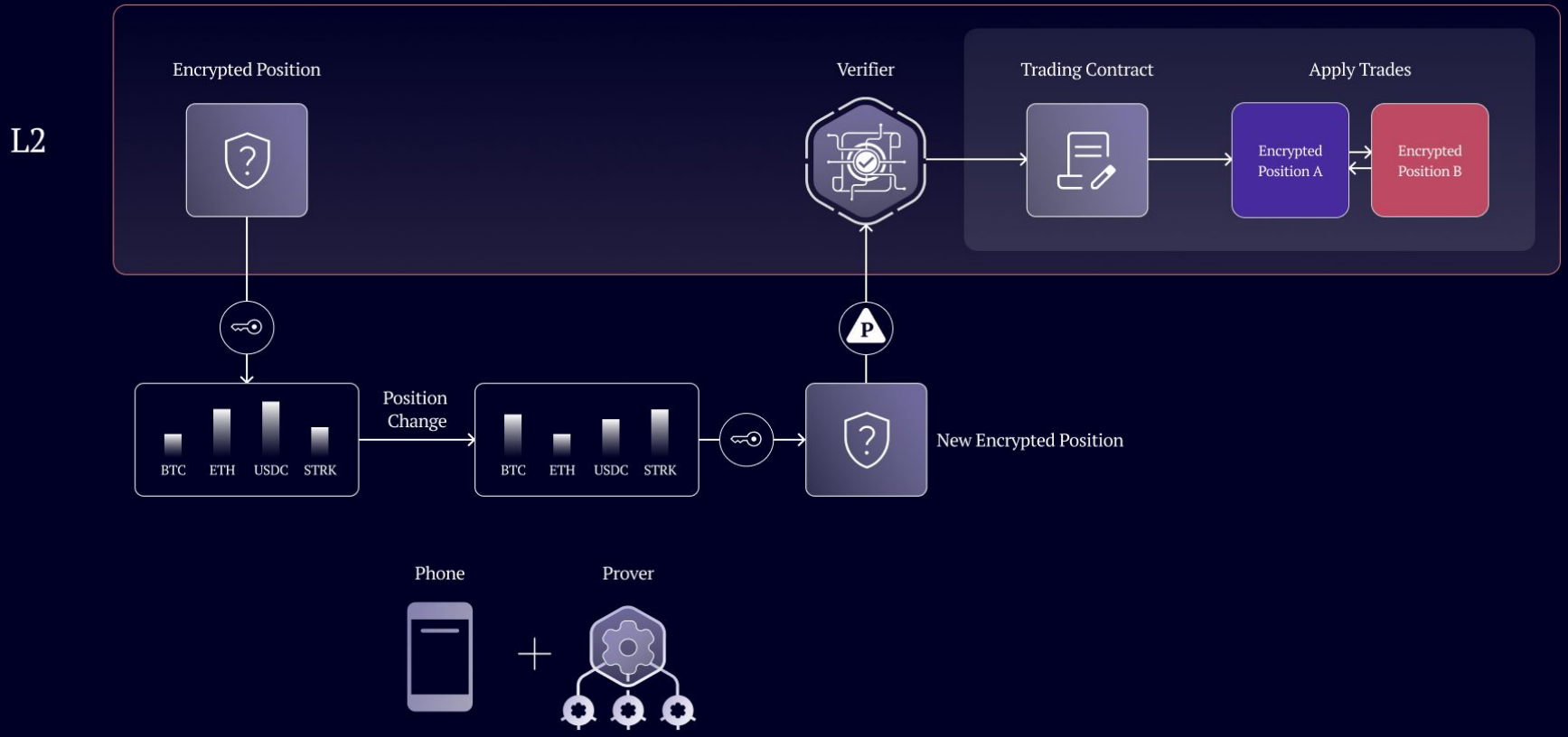
What else can Applicative Recursion do?

- Stateless Data Compression
  - Represent the data more efficiently
- Remove the need for state updates on L1:
  - S1 → S2, S2 → S3, S3 → S4 becomes S1 → S4
- Segmentation of large statements
  - E.g., prove that the state of a VM at the end of one segment is the state of the VM at the beginning of another segment
- Your idea here :)

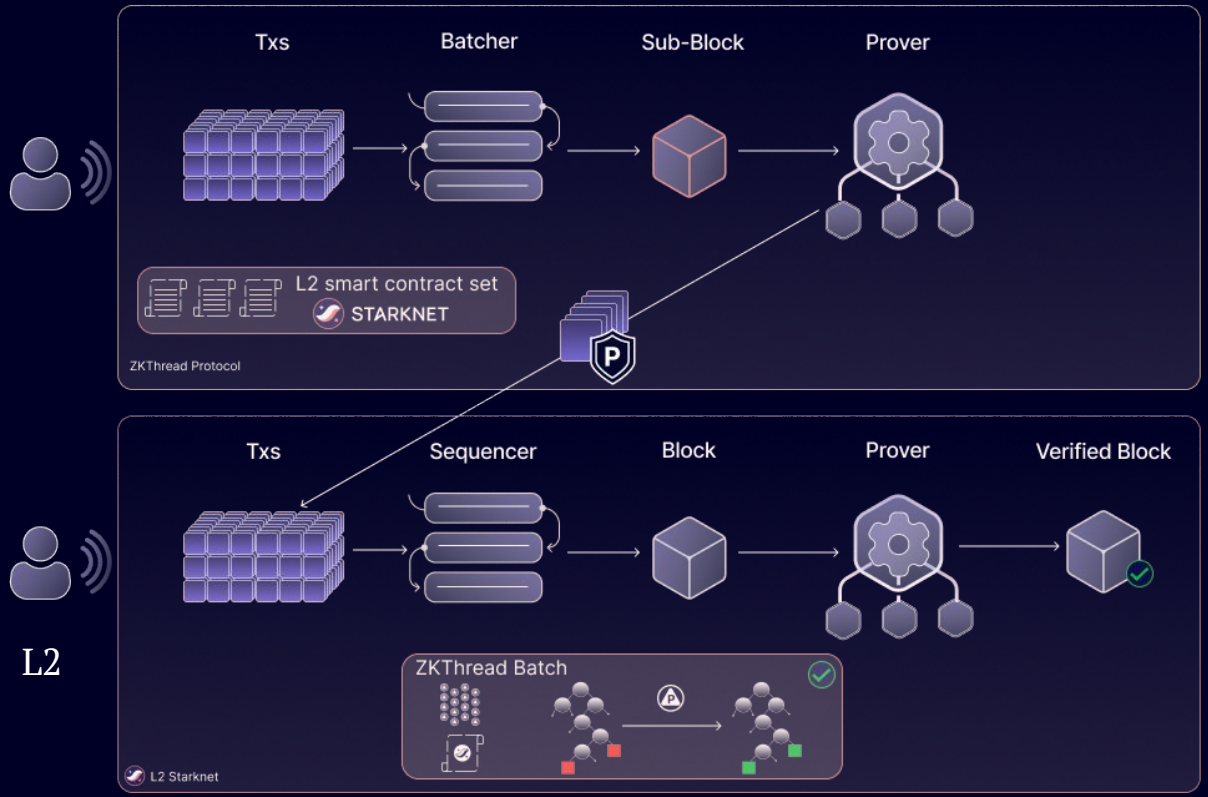
Section 4

Other Use Cases for Offchain Proofs

Offchain Proofs for Privacy



Offchain Proofs for ZK Threads



STARKWARE

Thank you!

Gideon Kaempfer
VP Engineering, StarkWare
gidi@starkware.co
@gkaempfer