NameHash Labs

ENS Service Provider Update
March 14, 2024

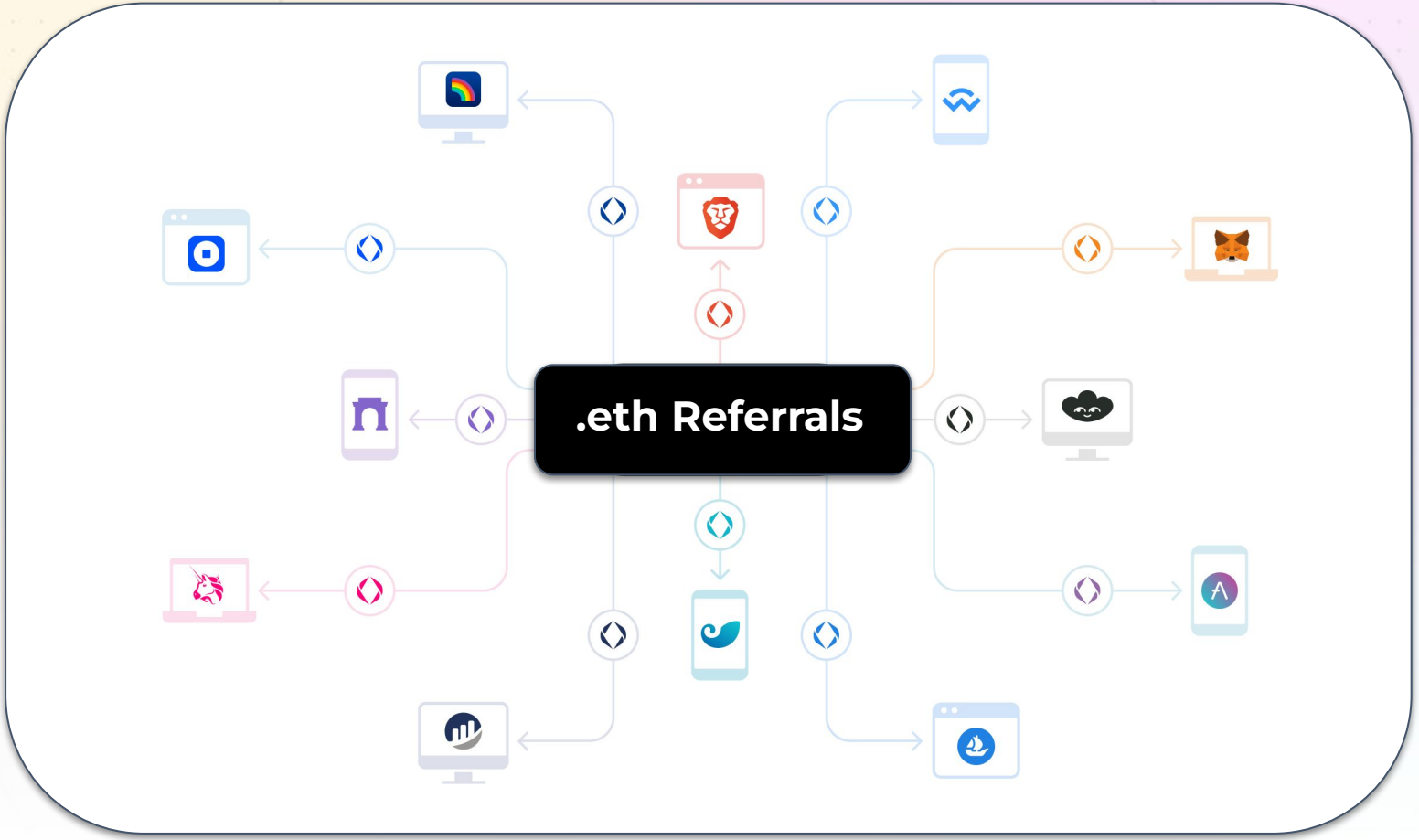
.eth Referral Program R&D Update

Referral Rewards Accounting & Distributions using ZK-Proofs (with Axiom)

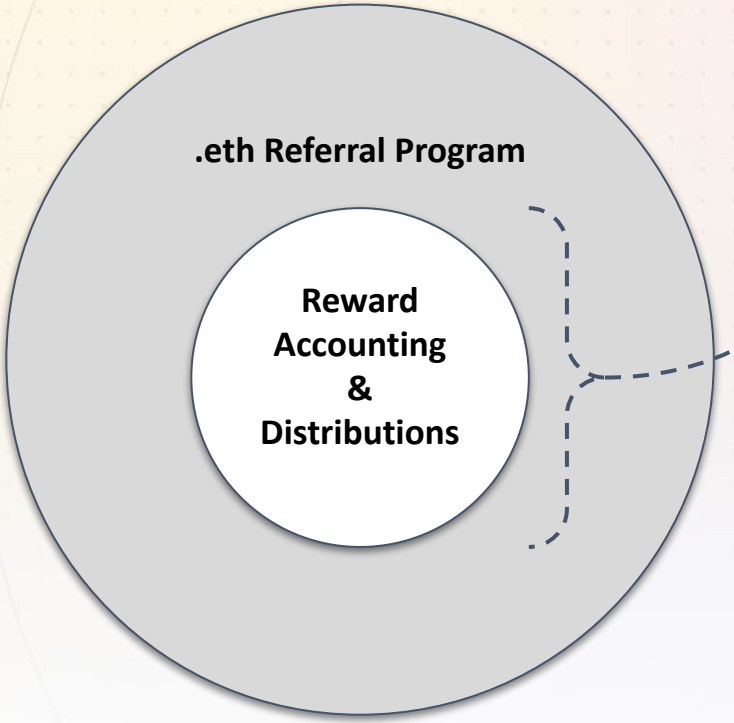
Our Mission

Help ENS grow

Strategy: Improve Incentives for Integrators



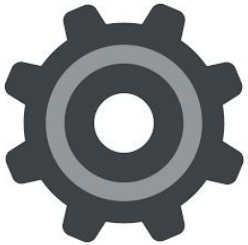
R&D Update - *Current Area of Focus*



Current R&D Focus

1. *How to **track referrer reward balances** in a trusted & decentralized way?*
2. *How to **distribute referrer rewards** in a trusted & decentralized way?*

Goals Overview - *More than just purely technical*



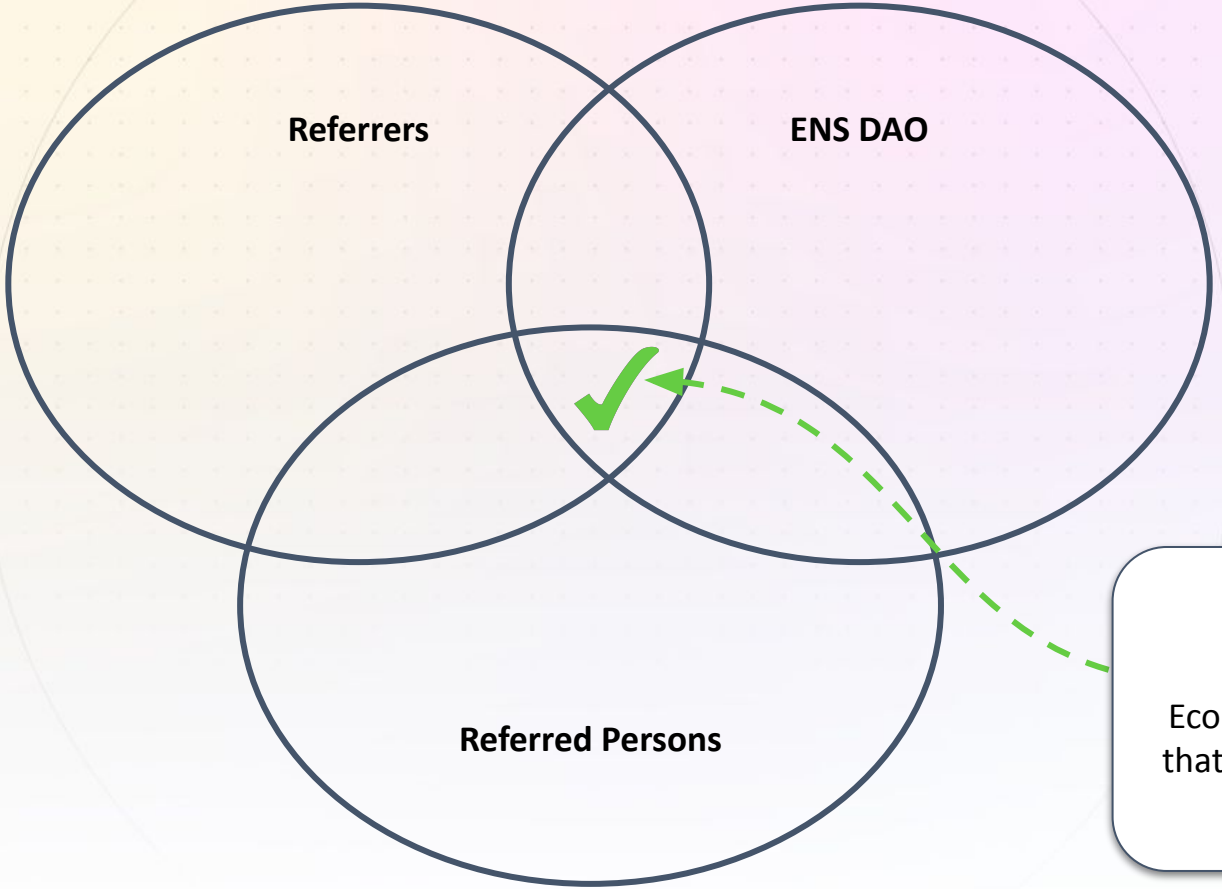
Technical Feasibility

Economic Incentive Alignment

Relatively trivial on its own

Both goals are critical for program success

Economic Incentive Alignments - Key Parties

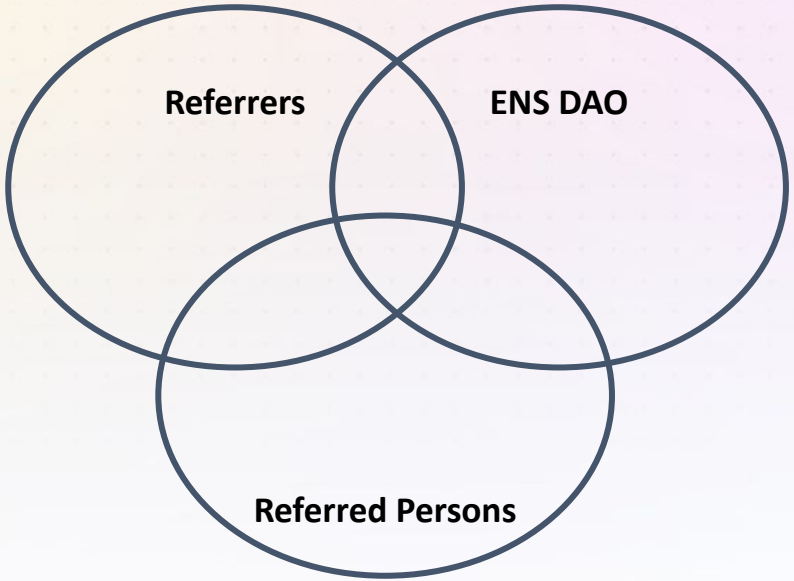


The Goal

Economic incentives that align to work for all parties

Economic Incentive Alignments - Key Goals By Party

- ✓ *Revenue share on registration & renewals*
- ✓ *No mandatory pricing disadvantage*



- ✓ *Grow .eth by incentivizing referrals*
- ✓ *Mitigate rewards for "unproductive" referrals*

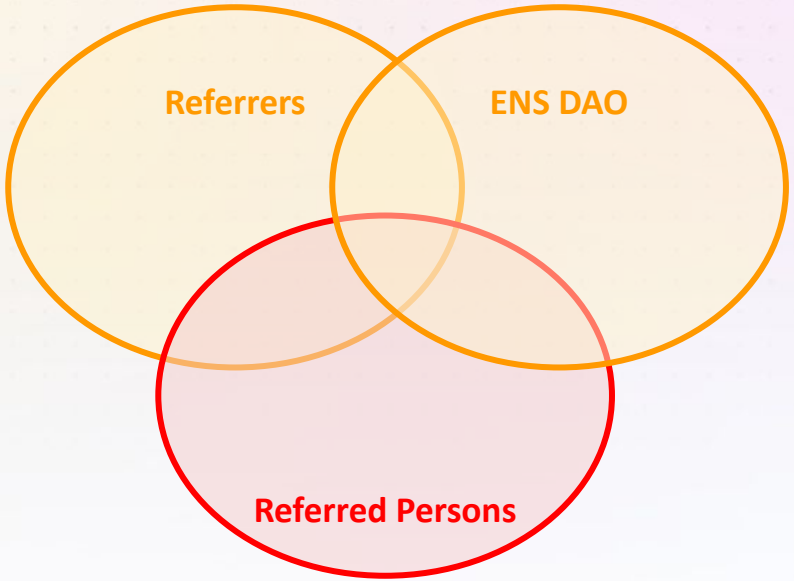
✓ *Pay no extra gas fees to be referred*

.eth Referrals - Naive Strategy A - Immediate Reward Distribution

- ■ *Update .eth Registrar Controllers*
- ■ *Immediate Reward Distributions*
- ■ *Immediate Accounting Updates*
- ✓ *Specialized Revshare Rules*

✅ *Revenue share on registration & renewals*

❌ *No mandatory pricing disadvantage*



? *Grow .eth by incentivizing referrals*

? *Mitigate rewards for "unproductive" referrals*

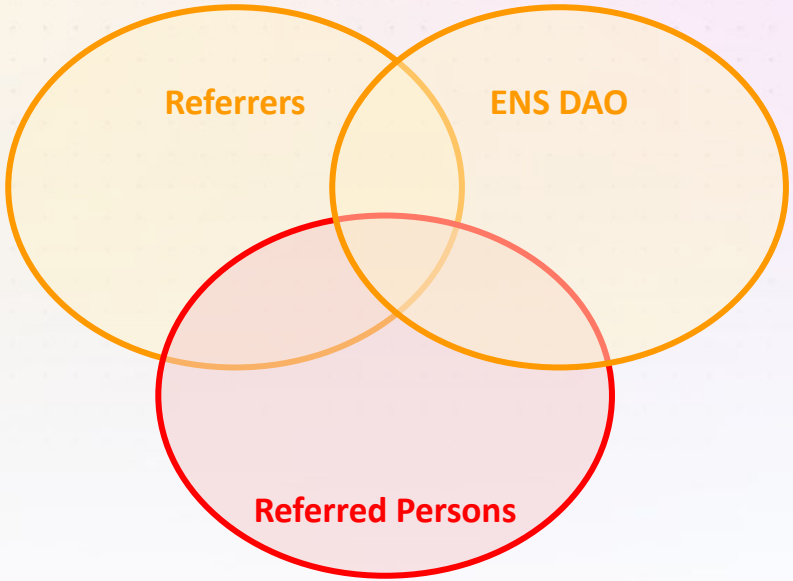
❌ *Pay no extra gas fees to be referred*

.eth Referrals - Naive Strategy B - Explicit Internal Ledger

- ■ *Update .eth Registrar Controllers*
- ✓ *Deferred Reward Distributions*
- ■ *Immediate Accounting Updates*
- ✓ *Specialized Revshare Rules*

✅ *Revenue share on registration & renewals*

❌ *No mandatory pricing disadvantage*



❓ *Grow .eth by incentivizing referrals*

✅ *Mitigate rewards for "unproductive" referrals*

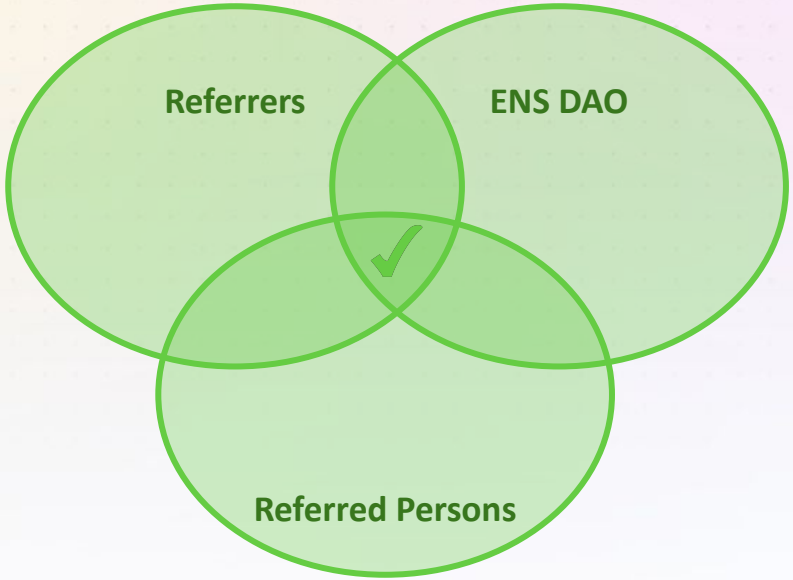
❌ *Pay no extra gas fees to be referred*

.eth Referral Program - ZK Strategy R&D

- ✓ *Unchanged .eth Registrar Controllers*
- ✓ *Deferred Reward Distributions*
- ✓ *Deferred Accounting Updates*
- ✓ *Specialized Revshare Rules*

✅ *Revenue share on registration & renewals*

✅ *No mandatory pricing disadvantage*



✅ *Grow .eth by incentivizing referrals*

✅ *Mitigate rewards for "unproductive" referrals*

✅ *Pay no extra gas fees to be referred*

ZK Strategy R&D - Collaboration with Axiom

What is Axiom?

ZK-verified access to historic on-chain data

Specify ZK data and compute in Typescript

- Try the Typescript SDK at **repl.axiom.xyz**
- Export a **client-side ZK prover** to specify queries

Query the entire history of Ethereum on-chain

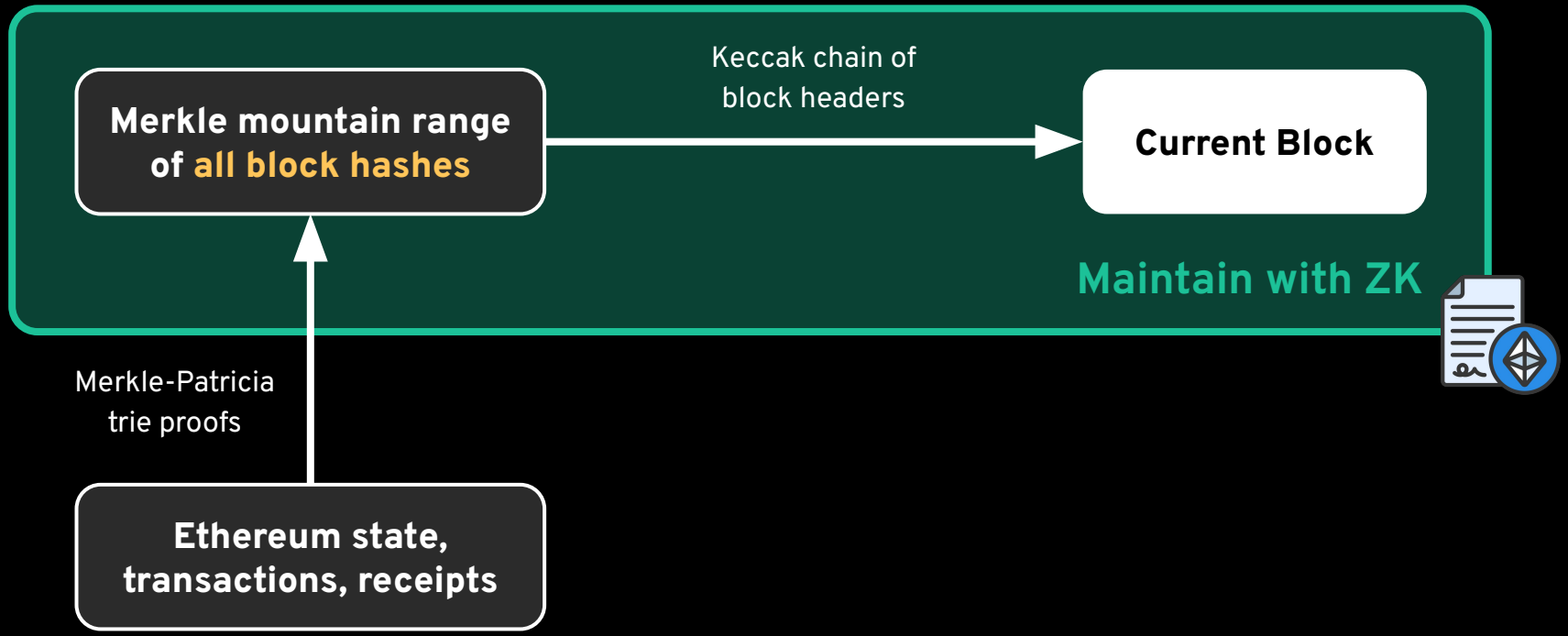
- Block headers, accounts, contract storage
- Transactions, receipts, Solidity mappings

Receive ZK-verified outputs in a contract callback

- No ZK-related contract deploys
- Results proven to be valid using zero-knowledge proofs

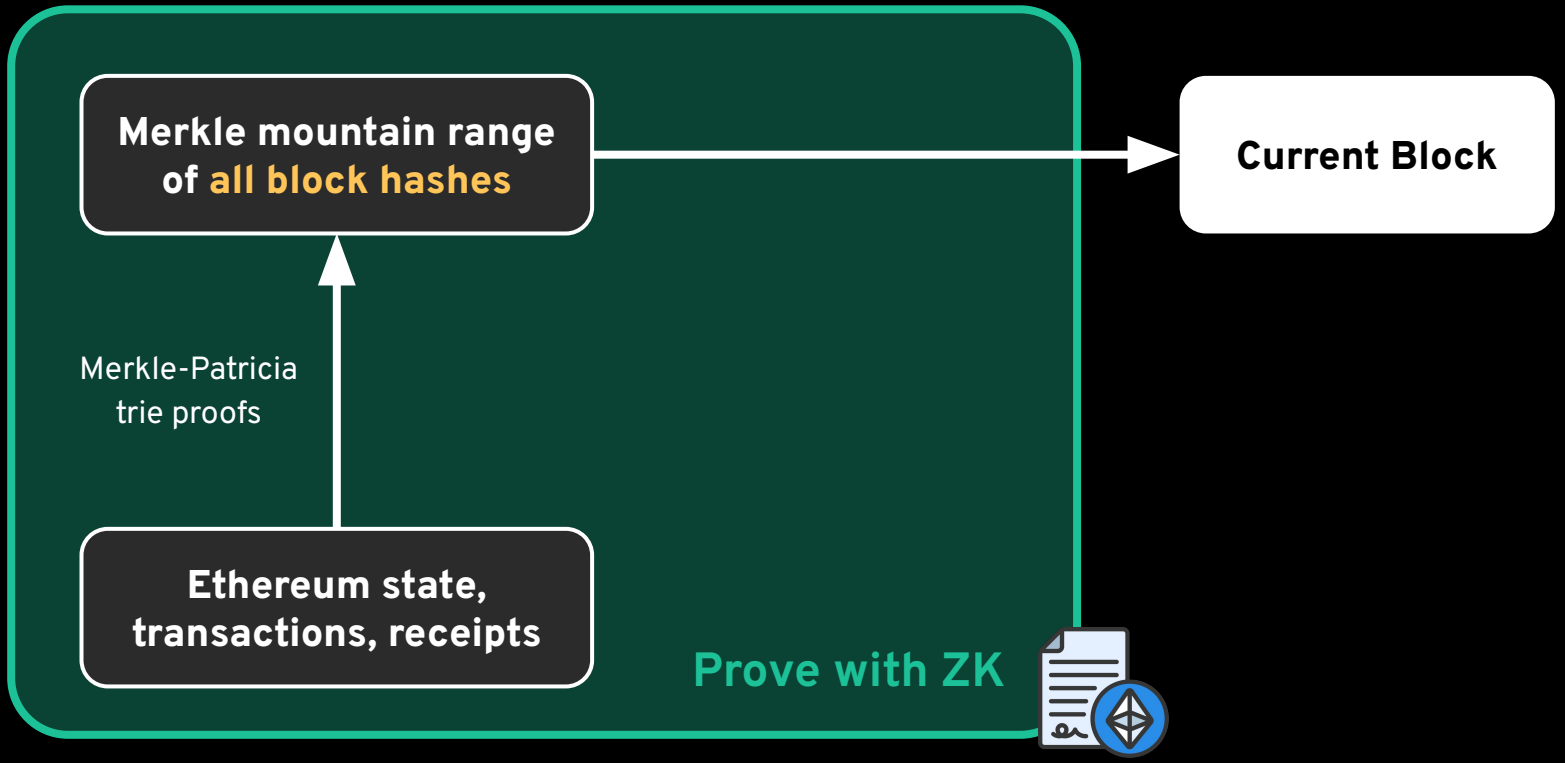
```
import {
  add, sub, mul, div, checkLessThan, addToCallback, CircuitValue,
  CircuitValue256, constant, witness, getAccount,
} from "@axiom-crypto/client";

export interface CircuitInputs {
  blockNumber: CircuitValue;
  address: CircuitValue;
}

export const defaultInputs = {
  "blockNumber": 4000000,
  "address": "0xEaa455e4291742eC362Bc21a8C46E5F2b5ed4701"
};

export const circuit = async (inputs: CircuitInputs) => {
  const samples = 8;
  const spacing = 900;

  if (inputs.blockNumber.value() <= (samples * spacing)) {
    throw new Error("Block number too low");
  }
  checkLessThan(mul(samples, spacing), inputs.blockNumber);

  let sampledAccounts = new Array(samples);
  for (let i = 0; i < samples; i++) {
    const sampleBlockNumber: CircuitValue = sub(inputs.blockNumber, mul(spacing, i));
    const account = getAccount(sampleBlockNumber, inputs.address);
    sampledAccounts[i] = account;
  }

  let total = constant(0);
  for (const account of sampledAccounts) {
    const balance: CircuitValue256 = await account.balance();
    total = add(total, balance.toCircuitValue());
  }

  const average = div(total, samples);

  addToCallback(inputs.blockNumber);
  addToCallback(inputs.address);
  addToCallback(average);
};
```

How does Axiom Work?



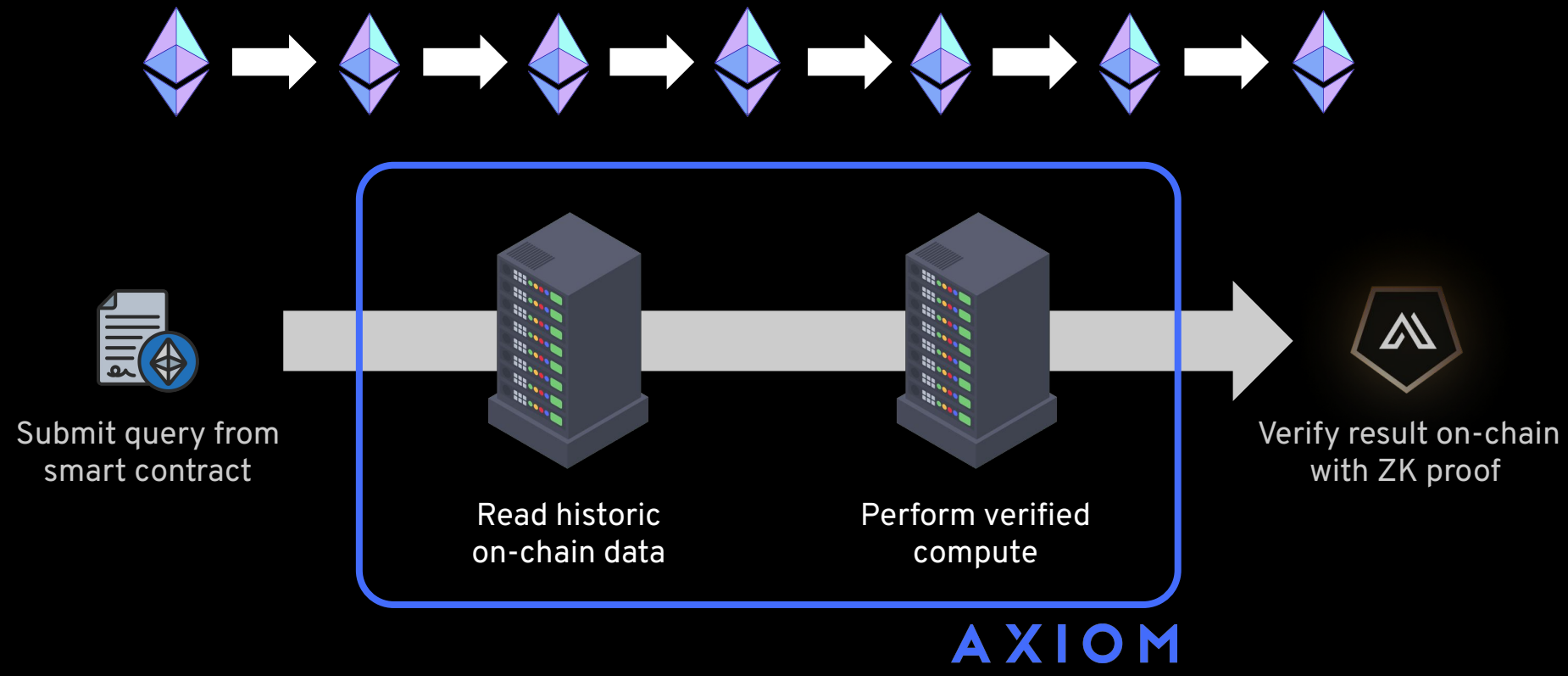
We cache block hashes back to genesis in a **Merkle mountain range**

How does Axiom Work?



We prove **all historical on-chain data** into this Merkle mountain range.

How does Axiom Work?



Using Axiom for .eth Referrals

Prove .eth registrations and renewals in ZK

User registrations and renewals use the existing ETHRegistrarController

- Frontends optionally inject a referrerId into the duration / expiry parameters.
- No changes are required to the on-chain registration / renewal flows or contracts.

Referrers claim by proving registration / renewal fees attributed to their referrerId with Axiom

- Fee amounts read from NameRegistered and NameRenewed events
- Premium fees and temporary premiums are excluded from the computation
- Trustless accounting ensured by validating data access in ZK

New ENSReferrals smart contract distributes rewards

- Axiom provides ZK-verified fees attributed to a referrer claim in a callback
- Double claims are excluded by a check in ENSReferrals
- Reward rates / policies can be controlled entirely in a smart contract

AXIOM

Join The Discussion with NameHash Labs

.eth Referrals Prototype

github.com/namehash/ens-referrals

Learn more about Axiom's Incentives Framework

Prototype Axiom Incentives Framework

github.com/axiom-crypto/axiom-incentives

General Axiom developer documentation

docs.axiom.xyz

@NameHashLabs
namehashlabs.org