ABCDE ZK Hacker Camp Jul 29, 2023

Lecture 2 — Axiom: The ZK Coprocessor for Ethereum

Instructor: Yi Sun *Scribes: Leo Fang*

1 Overview

This lecture was given by Dr. Yi Sun, an assistant professor at the University of Chicago and the founder of Axiom. He first addressed the issue of smart contracts being unable to query historical data efficiently, which also limits data-rich applications on the blockchain. He then introduced the features and architecture of Axiom and explained the steps to use Axiom from a developer's perspective. Finally, he discussed the ZK technologies behind Axiom, especially proof recursion and aggregation.

2 Painful Data Tradeoffs

Smart contracts nowadays face difficulties in efficiently accessing historical data, such as transaction and state information. For example, when using Opensea to search for a Pudgy Penguins NFT, you can't view all historical price and owners of the NFT. This limitation is not a flaw in design, but rather a necessity to maintain decentralization. If smart contracts could easily access the entire history of Ethereum, all archive nodes would need to be full nodes.

Developers then confront painful tradeoffs related to data: *Either pay more or reduce security.* You can store more data in state which leads to higher cost; you can also use trusted oracles based on certain assumptions. Scaling on-chain data access today is caught in a dilemma.

3 What Does Axiom Work

Axiom provides a solution to accessing on-chain history in a more efficient manner by utilizing cryptography independent of consensus mechanisms. Native access to history within EVM can be excessively costly due to the need for obtaining millions of intermediate block headers and constructing merkle proofs.

To address this challenge, Axiom introduces the practicality of historic data access through ZK techniques, leveraging the scalability and composability offered by ZK. The approach involves caching the entire history using Merkle Mountain Range up until the Ethereum genesis block, and performing computations using custom ZK circuits. Besides, Axiom uses parallized ZKP to aggregating historical block headers which can drastically reduce data size.

The workflow for utilizing Axiom involves reading and verifying computations, followed by the proof generation. Importantly, every result obtained through Axiom maintains cryptographic security

equivalent to Ethereum. Furthermore, computations can be performed without the limitations imposed by the blockchain's VM.

[axi]

4 Axiom for Developers

In this section, Prof. Sun introduced Axiom's SDK, providing a comprehensive overview of its functionalities, including installation, setup, query building, query submission, proof generation, and utilizing data within your smart contract. For more detailed information, please refer to the Axiom documentation.

This workflow enables developers to create a variety of data-rich dApps. In terms of **Identity and Governance**, possibilities include the development of autonomous airdrops and on-chain loyalty systems. Additionally, **trustless oracles**, such as Historic Uniswap LP share pricing and NFT transacted floor prices, can be implemented. Furthermore, it is possible to establish an **on-chain Reputation system** while safeguarding privacy, such as the Proof of Whale (proving ownership of at least 5 NFTs from a specific collection and having burned at least 100 ETH in gas).

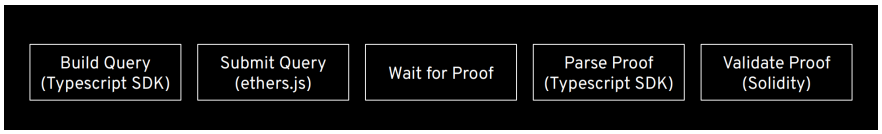

Figure 1: Axiom Workflow

5 ZKPs behind Axiom

In this section, we will delve into the ZKP primitives utilized in Axiom, focusing on three key parts: **parsing RLP serialization**, **Merkle-Patricia trie inclusion**, and **proof aggregation**.

5.1 Parsing RLP Serialization

All data in Ethereum is serialized using **Recursive Length Prefix**(RLP), which is a method for serializing nested byte arrays. Each RLP-serialized data consists of a prefix byte and optional length bytes that precede the actual data. These bytes indicate the length of the subsequent field.

Next, we employ **Random Linear Combination**(RLC) to arithmetize these arrays with variable lengths. Upon committing to arrays $a[i]$ and $b[i]$, we randomly sample a point r and define:

$$RLC(a[i], r) := (len(a), a[k]r^k + a[k-1]r^{k-1} + \ldots + a[0])$$

We then verify if $RLC(a[i], r) = RLC(b[i], r)$. If this condition holds, it confirms that $a[i] = b[i]$.

5.2 Merkle-Patricia Trie Inclusion

Merkle Patricia Trie (MPT) is a commonly used data structure in Ethereum for efficient storage and retrieval of large-scale key-value data. It is a data structure based on a Trie, with the incorporation of Merkle Tree concepts to enable data integrity verification.

The key feature of the MPT is its construction of a tree-like structure by encoding keys as prefixes. Each node in the tree represents a prefix of a key. In the MPT, each node stores a data item encoded in RLP.

The concept of Merkle Trees is applied to each node in the MPT to ensure data integrity. Each node includes a hash value computed from its child nodes. By recursively calculating the hash values of nodes, data tampering can be detected. Here we will use **Keccak hash**, which is computationally expensive in our ZKP construction.

5.3 Proof Aggregation

To reduce the proof size, we employ aggregation by combining the block header and MPT proofs. We mainly use non-native elliptic curve arithmetic and MSM to build this proof aggregation. Given proofs $\pi_1, \pi_2, \ldots, \pi_n$, we construct a recursive proof as follows:

$$\pi := \pi_1 \wedge \pi_2 \wedge \ldots \wedge \pi_n$$

The proof system construction is based on `halo2` with KZG backend (PSE fork) with modular setup, which is illustrated in the diagram below:

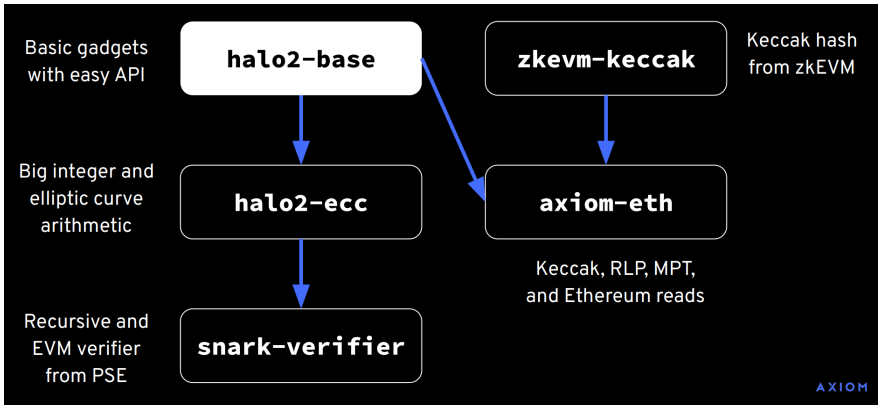

Figure 2: How does Axiom Build the Proof System?

You can learn more about the code implementations in this repository.

6 Vision

Nowadays, Axiom has already achieved several milestones, including trustlessly reading historical block headers, accounts, account storage, transactions, and receipts, and then computing via custom ZK circuits.

However, Scaling Data-rich Applications with ZK technologies is still on the road. Dr.Sun identified a potential optimal implementation path as follows:

- Trustless reads to all historic on-chain data. (including the beacon chain)
- `View` function simulation via zkEVM proofs. (we need type-1 zkEVM, with archive node running in ZK)
- Arbitrary compute via ZK-native VM.

Looking ahead, with advancements in technologies like Axiom, accessing and utilizing data on Ethereum will become more seamless. This opens up exciting possibilities for the development of data-rich applications, including machine learning models, recommender systems, and various other innovative solutions.

References

[axi] Axiom Documentation. https://docs.axiom.xyz/. August 11, 2023.