TLDR

: We just published our new paper on building a ZKP bridge for full Ethereum node with our open source code. We analyze why building a ZKP bridge for Ethereum full node is hard and how we address these challenges. Our bridge has many major improvements over the zkBridge paper, more decentralized, requires no worker-to-worker communication and is suitable for emerging prover markets.

Motivation

:

Building bridges on ZKP for Ethereum full node is hard as it requires enormous computation power. zkBridge is the first paper that sketches a layout on how to build such a bridge and Polyhedra Network (which has many authors of the paper) is one of the very few teams that claims to prove Ethereum full node.

The problems with current ZK Bridge for Ethereum full node:

- It's closed source, and does not have full community review. Given so many bridge hacks, it's hard for us to trust a trustless bridge that is in a black box.
- Public key aggregation is only a small part of work (in fact, less than 3% of computation work). We still need to prove the validity of each validator and the RANDAO.
- Solution is centralized and can only run in a data center.

This motivates us to build a decentralized trustless bridge for Ethereum full node from scratch. Our paper is public for community feedback. Our prototype code is open source for anyone to modify and optimize.

The problems

:

At the time of writing, Ethereum mainnet has about one million validator nodes. For simplicity, we assume that there are one million active validators in Ethereum and each slot has 32,000 validators throughout this article.

To understand why proving Ethereum full node is hard, one needs to understand its underlying consensus protocol. Each of these one million validators is required to attest to exactly one Ethereum block in one epoch. Each epoch has 32 slots and each slot might contain at most one Ethereum block. There are roughly 32,000 validators producing 32,000 signatures in one block time frame. These signatures are aggregated and then included into the Merkle tree of the block.

To prove that one block is valid with regard to current beacon state, here is a list of necessary but not exhaustive tasks a prover needs to prove:

- All these 32,000's signatures should be aggregated (in different committees) and valid under the new block hash.
- Each of these signatures belongs to an active validator under the previous beacon state.
- Each validator index must abide by an RANDAO algorithm which randomly shuffles and selects a set of validators for each epoch.

The challenge of proving Ethereum full node is generating proof for 32,000 nodes. The second and third tasks require proving multiple SHA256 hashes. To prove the second task for one validator, a prover needs to show that the validator's public key is part of a Merkle path whose root is the previous beacon state hash. Ethereum consensus uses SHA256 for hashing and the current best SHA256 proof requires about 27,000 constraints. Each path length from validator info leaf to the root is between 50 and 60. We are looking at 1,350,000 constraints per Merkle path per validator. Multiplying this number by 32,000 yields 43,2 billion constraints.

Proving the third task is even more challenging as each RANDAO to get the proposer index for a block requires multiple SHA256 hashes. We estimate that proving all validator indexes are correct under RANDAO requires between 100 billion to 1 trillion gates. Even the lower range estimation would exceed the computational capacity of the current best proving system. Note that these are not the complete list of problems we have to prove for Ethereum full node but the ones we want to highlight.

Our Contribution

Our contributions of Sisu protocol in this paper is as follow:

- We introduce an algorithm to prove that the majority (more than 2/3) of validators in Ethereum network in one epoch are validators under the beacon state in the previous epoch and they are all pairwise different. A naive approach compares all pairs of validators public key and runs in O(n^2). We propose an algorithm that runs on O(n) in circuit by leveraging special properties of the validator set.

We argue that by proving the uniqueness of the majority of validators in one epoch we can replace the RANDAO algorithm of Ethereum consensus with our new method. To verify one block, the RANDAO algorithm defines a specific set of validators while Sisu protocol requires any set of valid validators. However, within one epoch the set of validators of RANDAO is the same as the set of validators in Sisu protocol. This set is the set of active validators in Ethereum network. The cost of attacking Sisu protocol for this part is equivalent to the cost of manipulating the majority of Ethereum networks.

- We propose a new approach that requires no worker to worker and minimal traffic between worker and master in our distributed system. This allows Sisu to scale better compared to the design in zkBridge.
- Our protocol is built on a distributed GKR system using a general circuit. Compared to a layered circuit, the general circuit reduces the number of layers in the circuit and achieves 3-4 times speed up in performance.
- Our circuit uses a new type of gate called accumulation gates that allows more than 2 input gates connecting to the same output gate without modifying the GKR evaluation function. Though the number of evaluations remains the same, accumulation gate significantly reduces the number of layers and the number of gates per layer.
- We incorporate GPU hardware acceleration in our implementation with multiple optimization strategies. We first implement Sisu protocol in CPU, identify the part that could run in parallel and gradually port them to GPU.

Sisu protocol is designed toward decentralized proving. We make a deliberate trade-off of having decentralization at the expense of increasing the number of constraints. However, with the emerging prover markets we expect ZK applications in the future could run on hundreds and even thousands of GPUs. Any protocol that can be divided into independent provers could take advantage of distributed proving systems.

Background & Our Protocol

If you are new to the ZK world or mostly uses ZK framework like Circom, I would recommend you to read on the following concepts before reading our protocol:

- GKR
- GKR for general circuit.
- Libra paper: introduces a new method to calculate multilinear sumcheck in linear time using 2 rounds bookkeeping tables.
- Virgo: a transparent protocol to allow a verifier to delegate the computation at input layer to prover using VPD (verifiable Polynomial Delegation).
- zkBridge: a paper that uses a distributed version of Virgo to build general zkBridge.

This list could be overwhelming, so take your time to digest them.

After reading on this list, you can start reading on Sisu protocol. Our protocol is based on Virgo with some major differences:

- We use a distributed general circuit instead of a distributed layered circuit. The formula to calculate multilinear sumcheck is different.
- To remove worker-to-worker communication, we use cluster shared mempool for a group of workers and increase the number of Merkle paths that the verifier has to validate. This is a deliberate trade-off as worker machines can run independently and easier to scale to a decentralized network.
- A new associative hash function is introduced to prove that validators in different blocks are pairwise distinct. Previously, this was not possible since proof is generated for one block at a time.
- A new accumulation gate is introduced to reduce the number of gates & layers. Our SHA256 circuit has only 4 layers to prove.

Our protocol has 2 phases: phase 1 runs distributed sumcheck and distributed polynomial commitment. Phase 2 uses Groth16 to emulate the verifier of Phase 1. Both phases can have overlapping execution.

We are thinking about using Halo2 to replace Groth16 for better performance. However, our engineers are more familiar with Circom and decide to use it for a prototype.

More detail is in our paper.

Evaluation

We use machines with Intel(R) 8 cores i7-7700 @3.60GHz

CPU, 32GB RAM and Geforce GTX 1080 (8GB VRAM) GPU. Our GPU is an 8 year old model. We chose the modest hardware configuration because they provide the best RAM and compute power per dollar.

One Geforce GTX 1080 GPU can handle generating proof for 16 validators comfortably within 12s in our prototype. The circuit for these 16 validators has about 30M gates in phase 1.

At full optimization, one such GPU can handle between 32-64 validators. 32,000 validators will need 512-1000 such GPUs with a total circuit size of 15-30 billion gates.

Our longest operation is the witness generation time in Groth16 as it takes dozen of minutes to execute. We are using rapidsnark to generate witness which is not under our control. We are looking into tactics to optimize this process, either by optimizing rapidsnark or write our own version.

Conclusion

We present an open analysis and open source code for Ethereum full node. Our approach can apply for general ZK applications with parallel structures with billions of gates. More importantly, we design Sisu protocol to be decentralized and highly suitable for running with distributed prover markets. Feedback is welcome.

If you want to contribute/ support/ fund the project, feel free to contact me at [my_first_name].[my_last_name]@sisu.network

or reach out to me on Twitter/X at billypham09.

Billy Pham