Special thanks to TXRX research team, @AgeManning, @protolambda, @djrtwo, @benjaminion, @5chdn

The Eth2 community has long speculated that validator privacy will be an issue. For background, refer to this issue @jannikluhn opened on validator privacy almost 2 years ago:

- https://github.com/ethresearch/p2p/issues/5

@liochon suggests the following solution here:

- Cryptographic sortition: possible solution with zk-snark

and @JustinDrake improves on it:

- Low-overhead secret single-leader election

Despite all this, Eth2 still doesn't provide privacy preserving (alliteration

) options for validators. To be fair, no one has demonstrated a method of exposing validator IP addresses. As a result, the problem has been somewhat limited to the existential realm. With the Phase 0 launch growing closer, I have been giving the following question a fair amount of thought:

What is the simplest way to deanonymize validators?

This post is dedicated to exploring this question.

Data Collection

The data for the following analysis was collected on the Witti Testnet from Wednesday, June 10, 2020 through Thursday, June 11 by a single network agent designed specifically to crawl and collect data on the eth2 network. The network agent uses Sigma Prime's implementation of discv5 and the gossipsub implementation they contributed torust-libp2p.

The data collection was done in two phases. First, the network agent crawled the Witti testnet in order to locate most/all of the testnet nodes. In order to optimize the DHT crawl, some minor modifications were made to Discv5 params:

- MAX_FINDNODE_REQUESTS:

7

- MAX_NODES_PER_BUCKET:

1024

Crawl Summary

- total node count:

134

- validating node count:

78

- non-validating node count:

56

Note: the ENR attnets field was used to determine if a node hosts validators

Next, nodes discovered during the crawl were used to select peers and begin logging gossip messages. Minor modifications were made to gossipsub params:

- mesh_n_high:

set to the estimated number of validating nodes

- mesh_n_low:

set to 1/3 of the estimated number of validating nodes

- mesh_n:

set to 2/3 of the estimated number of validating nodes

- gossip_lazy:

set to 0

In addition, the gossipsub LRU cache was remove to enable the logging of duplicate messages.

Note: Blocks were the only gossip messages logged.

Gossip Summary

- Starting slot:

105787

- Ending slot:

112867

- Number of slots:

7080

- Number of peers:

17

- Number of peers validating:

11

- Number of peers not validating:

6

Data from the DHT crawl was joined with Gossip data to create a dataset with the following fields:

Data Analysis

Given the data collected, do you think it is possible to determine (with a high degree of confidence) the ip address associated with any of the active validators?

Let’s start by looking for peers that always notify the agent first with respect to blocks created by particular proposer indexes.

[

962×347 29.7 KB

](https://ethresear.ch/uploads/default/original/2X/b/b334344b34efce8c9390f3385c5974c7f6972875.png)

Our peers change and anomalies happen so it’s probably okay if a particular peer isn’t ALWAYS the first. Next, we need to transform peer_id into a categorical variable that can be included in a visual analysis.

As you can see, peer_id can be conveniently mapped to a categorical variable peer_id_cat. This makes it easier to plot (and even use in models). Since we are paying attention to what peer is first to deliver a block, it’s probably a good idea to track what peers are active and when.

[

700×450 14.1 KB

](https://ethresear.ch/uploads/default/original/2X/3/30ec2e2d7e3849e96856e2ecb2decc4cff7cad58.png)

This gantt chart gives us a rough idea when/if the peer is still actively sending the agent blocks.

Now we are ready to look at some different views of proposer index vs. the first peer id notify the agent of the corresponding blocks.

[

1003×489 124 KB

](https://ethresear.ch/uploads/default/original/2X/d/dbff6ded3ab5a8efc775695c0351bc0d60f1bec6.png)

Notice how there seems to be a large consecutive sequence of proposer indexes associated with a single peer id? If I deposited a bunch of eth in order to activate 128 validators, then wouldn't they have consecutive indexes in the validator registry? How convenient…for me. Let's zoom in.

[

1003×489 17.1 KB

](https://ethresear.ch/uploads/default/original/2X/1/1a7a462ebc2bea957a22395a060d405bfad56401.png)

Just like before, the x-axis is proposer index, but this time the y-axis represents peer-id. The more times a proposer is selected and the same peer notifies the agent, then the fatter the line. If many different peers have been the first to notify the agent, then it will just look like the walls are melting (aka noise).

Finale

[

786×670 189 KB

](https://ethresear.ch/uploads/default/original/2X/3/383b98f770d26a3acc0a5f954134c3596d6cdbd7.png)

The diagram above outlines my thought process. No models. Just plots. This only scratches the surface of what's possible.

Denouement

I was looking at the plots above and realized that I should probably verify my guess with known validator indices. Remember peer-id: 16Uiu2HAmK3aw5p4Uw7RYRFeUcL4u1pg3u6JN8MnyT5wRshNLvHqU

(aka peer_id_cat = 6)? I looked up the associated IP, saw it was in Berlin and assumed it was Afri. He was generous enough to share the public keys of his 384 validators so that I could validate my methodology.

If Afri has 384 validators, then why was I only able to predict 128? Simple. The agent wasn't peered with Afri's other validating nodes so the data didn't provide strong signal. This indicates that this methodology provides some resistance to false positives.

Suggestion(s)

Batched deposits resulting in consecutive validator indices running on the same node is a dead giveaway. This can be easily exploited. At a minimum, we should suggest some best practices for splitting keys across nodes.

Future Work

- follow-up post on DHT analysis
- follow-up post on Gossip analysis
- derive a model to output ip address and probability for a given validator index
- look at other network messages besides just blocks
- take a closer look at the relationship between message_size and arrival time variance.