A wrap-up of twitter spaces we had last Friday, “Decentralized and permissionless proving design.”

Many thanks to the discussion participants who joined us to share their perspectives, thoughts, and experience:Joe, Omer, Hugo, Brecht, Nick, Daniel, Alex, and Misha.

Part 1: definitions

Permissionless

- Anyone can sign up or register for this.
- There is no central authority that controls the access.
- Crucial for market efficiency as the proof computation demand can be spread across the market and hardware with extra capacity to obtain proofs at the lowest cost.
- On the opposite, if there are only twenty provers whitelisted by the project, it might result in collusion between these provers for proof (i) cost increase, (ii) impact on block ordering through proof withholding for a specific block so the block is not finalized for a long time, (iii) if the provers are in one location, there might be a regulation risk, (iv) permissioned provers can become a bottleneck for the rest of the system to be decentralized.

Liveness

- In general, it depends on the specific design of a specific rollup. For example, if block proposal and block proving are separated – then the chain is live as long as new blocks are proposed. Furthermore, the security of a ZK-Rollup doesn’t depend on the prover decentralization (the worst thing the prover can do – not to generate the proof on time that will make ux worse because of longer time to finality but has no impact on security.
- However, proving time impacts (i) withdrawals – how long users have to wait to withdraw their funds to L1, (ii) all cross-chain communications between not only L1 <> L2 but also other L2s. So the faster the proof is generated, the better the UX.

Decentralization

- A large number of different provers run by different operators. In a perfect case, they are geographically distributed, run their machine with different hardware and software setups, etc.
- Its importance depends on a specific rollup and its goals. For some specific cases, a rollup might prefer to have a centralized mechanism with a limited number of provers. However, for most general-purpose rollups it seems reasonable to aim for having a decentralized prover network.
- Going further, we can decentralize not only the operators but also a proof construction. That is, instead of generating one proof for a large monolithic block, we can break it down into tiny pieces of computation, and proofs for different pieces will be generated by different provers.
- This computation split is also connected with hardware requirements, which will be much lower in the case of a block split. Furthermore, it will allow rollups to distribute more evenly and smoothly the proof generation load between different provers. As in the case of managing very powerful and efficient hardware, what should it do in case of not winning the current block? Wait until it finally wins the block?
- We can assume that the network is decentralized enough if, in case of unexpected incidents or black swans, the network can respond fast and continue generating proofs.
- There might be a two-level proof system (with the secondary proof market) with one level more decentralized and another level less decentralized. That is, if we have a secondary proof market where a lot of provers generate proofs for small pieces of computations, then there might be just several aggregating provers who build proofs of proofs.
- However, speaking about prover decentralization, it’s crucial to understand the origination of provers’ addresses to be able to differentiate different entities or individuals running provers.
- At the same time, prover reward calculation should stay as simple as possible. For example, it can be done through a smart-contract.

Part 2: three approaches to decentralize the prover

Disclaimer: there are many more than three approaches. The three mentioned below are just the most often observable today. And for each of them there are many variations.

Proof racing – FRFS approach: the fastest proof is accepted. Under this approach, provers have the greatest incentive to be as fast as possible. The advantage is that it allows us to get proof pretty fast and at a low cost. The problem is that the single fastest operator wins every slot, so it can result in pure centralization. The drawback is that a lot of provers work in parallel, which results in a waste of computations.

Proof mining (similar to classical PoW) – a bunch of ZKPs are generated by different provers until a specific condition is met. Whoever met this condition gets the right to generate the next proof. The issue with the waste of computational power is similar to the proof racing case.

Staking-based proof – all provers have to put up a stake. A selected algorithm determines who will generate the next proof (e.g., pseudorandom selection with the probability to be chosen proportional to the stake size). As an example of a pseudorandom mechanism, the input from the previous block can be used as a randomness to choose a prover for the next block.

The problem with this approach is that as provers don't have a huge incentive to compete with each other, it might result in less efficient proving. Another issue is that requiring a high stake might be a barrier for some potential operators to join that will limit the decentralization.

Check this article by Nick (aka Trace) for a more detailed overview.

In a more general context, a *p*roof market is any economic mechanism incentivising proof generation.

The proof market design is defined by several criteria:

- What restrictions are put on provers. For example, a fixed window to confirm a proof or a minimal level of stake is required to join as a prover. These restrictions might decrease decentralisation. For example, back in 2019 - 2021, there was a case when to submit proof directly to Filecoin protocol, provers were required to buy or lend a huge chunk of Filecoin tokens. It prevents from participating those who don't want to invest in large token amounts either because of the stake absolute value or the token volatility. The last issue can be fixed through staking, for example, in dollar-based or ETH-based assets.

The specific mechanism depends on the rollup application and strategy:

- In some cases, slot-based proof generation works better. For example, if proof should be verified before it is put in a rollup.
- In some cases, auctions are more reasonable. The fastest prover wins is, in fact, an auction.

Part 3: provers business and economics

- Generally, if it is economically profitable to contribute computational power – that is a sufficient criterion for a prover to jump into;
- How profitable proving is for the prover depends on each specific case;
- The proof generation market is pretty early. But as long as there is some free computational power in the world, operators are interested in participating in different projects, playing and experimenting with them even though their economics are not as mature and predictable (as, for example, for Bitcoin). Today it's not about business, it's more about experiments and exploration.
- The economics of proving is less obvious than what we are used to with mining. Furthermore, the economics might be very different for different types of proving systems (e.g., stake-based and proof mining). In some cases, it might be more profitable to focus on one particular network and optimize the software and hardware to be the most efficient in that domain. For other cases, it might be more profitable to provide some sort of service, that is, generating proofs for a specific company or app by agreement (e.g., this might be the case for ZKML).
- It is challenging to find a good trade-off between minimizing proof cost (that affects user transaction fees) and decentralization though both are crucial.

Part 4: secondary proof markets and delegation

- Secondary proof markets seem to be a lucrative idea both in terms of contribution to decentralization, low cost for users, and efficient hardware usage for provers. The reasoning for efficiency is that some good matching mechanism can match particular hardware with the most appropriate circuit.
- However, a list of limitations exists today to make secondary proof markets real. Firstly, some rollups use very different proof systems and schemes to construct proof.
- If we want to break down proof generation into parts (e.g., one computes MSM, and another computes FFTs), we need some sort of standardization. One part that needs standardization is the verify support for particular proof systems (such as Halo2). Another part is circuit description "standardization" that should be machine-readable. That is, it can be explicitly defined from the circuit by a machine what type of proving system is used there. For today, there exist not too many proving systems, so the task is pretty doable (=nil' Foundation is currently working on that).