Tl;dr: we can abstract execution tickets even further by expecting something more general than a signature. This could smoothly allow PEPC, distributed-block-builders and other weird things.

I would consider the original execution tickets post to have accomplished two things:

- proposed a concrete mechanism that takes the “PBS” separation to its natural conclusion
- formed the concept of “proposing rights” and made it clear that this is something which can be allocated in arbitrary ways.

In particular, the execution ticket proposal has gotten people to discuss the allocation of proposal rights more generally. The questions (which we are slowly working on answering) that came up along these lines are:

- what properties do/can we want from the proposing right market in terms of concentration, contestability, geo distribution etc?
- how far in advance should proposing rights be allocated?
- to whom should rights be allocated? Randomly to stakers? To whomever buys it?
- etc.

However, I think we haven’t really gone the full way of generalising what an execution ticket is. A reasonable model for capturing the popular understanding of execution tickets is a data structure with three fields. Here I am using an execution ticket as a means to represent proposing rights, not necessarily within the context of the lottery that was originally proposed:

- owning_account
- this eth account has the ability to change proposing and owning ticket fields (i.e. change the owner and proposing key).
- slot_no
- at some point the protocol sets this value according to some unspecified rule, likely with only one ticket per slot (but who knows)
- proposing_key
- when the current slot is equal slot_no, the chain accepts as valid only blocks that are signed by the proposing key
- as we have today, the protocol probably specifies some slashing conditions according to which funds associated with a ticket are subject slashing. For instance, this would prevent one ticket being used to produce two valid blocks (equivocating).

The core point I want to make is that we can generalise the proposing key to just be a proposing check. Instead of validators expecting a signature from a specific key, they could expect some input which, when fed into to the proposing_check script, evaluates to true.

The only constraint on the proposing script is that it shouldn’t be too expensive to execute to prevent validators from being DoS’ed. We could take a hint from the work on SNARK accounts on Celestia and verify proofs using this field. I suspect there’s a lot of wild stuff we can do here that I haven’t thought of, but two interesting directions:

- constrain block validity: Especially if there was a way to lock the proposing_check so that it couldn’t be changed, this would be a way to commit the block to conforming to a broad class of rules (similar to what was suggested in PEPC) in addition to the protocol’s block validity rules. This includes inclusion lists or certain ordering rules.
- light clients: For example, you could require that an L1 block is actually a concatenation of many smaller blocks from a different chain. This is interesting for decentralised block building, but probably for some other things.

Both of these framings can probably lead to some really interesting constructions. I haven’t thought about this very deeply yet so I assume there are other interesting directions to take this as well.

Outside of the constructions we would practically want to build, it is also just an interesting reframing of execution tickets (at least to me). One interesting intersection to explore here is the overlap between this idea and “the ethereum as a rollup to itself” framing. My aspiration with this post is to stimulate another philosophical tractus from @barnabemonnot.