What is PEPC?

PEPC (pronounced "pepsi") stands for Protocol-Enforced Proposer Commitments. PEPC was originally proposed in the post " Unbundling PBS", published in Oct. 2022 before Devcon 6. It is intended as an enshrined protocol gadget allowing validators ("proposers") to enter into binding commitments over the blocks they produce.

Unbundling PBS: Towards protocol-enforced proposer commitments (PEPC)Many thanks to @fradamt @casparschwa @vbuterin @ralexstokes @nikete for discussions and comments related to the following post. Personal opinions are expressed in the first-person-singular. Protecting the proposer and ensuring liveness of the chain are a big part of why PBS is considered to be moved into the Ethereum protocol. Ideally, when the proposer utilises the services of a builder, there is a contract between parties for the delivery of some goods (valuable blockspace), and the contract ...https://ethresear.ch/t/unbundling-pbs-towards-protocol-enforced-proposer-commitments-pepc/13879?u=barnabe

The simplest mental model for PEPC is as a generalisation of PBS. Today, validators overwhelmingly delegate the construction of their block to specialised parties known as "builders". This delegation is operated out-of-protocol by the MEV-Boost marketplace, where trusted relays guarantee the fair exchange of builder items (full blocks) against proposer signatures, without which the builder items remain invalid.

ePBS intends to enshrine some model of proposer/builder separation (PBS) in the Ethereum protocol, by enshrining the fair exchange in the rules of the protocol. Ethereum validators will be responsible for ensuring atomicity of builder block delivery against the proposer signature. This atomicity is critical to protect proposers and builders alike.

ePBS envisions enforcing a single type of exchange between builders and proposers, for instance, a full block containing a payment to the proposer, in exchange for a signature from the proposer to make the block valid. Many types of exchanges are possible, but ePBS would ultimately enshrine only one of them as a global setting for all proposers. Meanwhile, PEPC asks: what if proposers were able to enter into any programmable contract with builders? This includes a host of block-building primitives such as partial blocks, commitments to orderings and many other ways to segment block building, detailed below in the PEPC use cases.

Why is this useful?

Enshrining mechanisms in-protocol is desirable to boost the mechanism's credibility beyond that of third-party actors such as relays. By enshrining, the Ethereum community implicitly commits to defending the mechanism, via the control it exercises over validators who execute the protocol. Enshrining also provides additional introspection of the protocol, who becomes aware of certain actions executed in its environment.

Seeing like a protocolWhere does protocol credibility come from?https://barnabe.substack.com/p/seeing-like-a-protocol

The trade-off: enshrining anything locks in certain outcomes, which may reveal to be suboptimal from various perspectives

(political, economic , technical…) To explore this trade-off, we propose a different perspective on PBS, splitting PBS in two layers:

The market structure: Validators enter into agreements with outside parties (”builders”), and there is a legal system (consensus rules) which guarantees the fair exchange between them.

The allocation mechanism: The specific shape of the agreements entered into by validators and builders.

Devcon 6 presentation on the two layers of PBS

Today, neither layer is enshrined in the protocol. TheBuilder API, a module in all consensus-layer clients, allows proposers to offer their signature to builders, but this module is not part of the Ethereum consensus rules, nor is the abstraction of a builder ever instantiated there. On the opposite side, ePBS enshrines both the market structure, instantiating the “Proposer” and “Builder” abstractions in the consensus rules of Ethereum, as well as a specific allocation mechanism, for instance, a full block in exchange for a signature.

PEPC sits in the middle, recognising that thefundamental asymmetry in complexity between block construction and block validationmeans that the market structure is justified, but stopping short of enshrining a specific allocation mechanism regulating the contracts between proposers and builders.

What is the status of PEPC?

PEPC is very much still in the research phase. Since the publication of the “Unbundling PBS” post, PEPC has remained a subject of discussion with collaborators. In my opinion, it has prompted useful discussions around some risks of ePBS and the larger design space, but it has also failed to materialise a concrete design. Current work aims at bridging this gap, but doubts remain over the feasibility and our willingness to use PEPC in its full generality.

On the feasibility, it had become clear that PEPC would require the same research as ePBS to create a safe mechanism from a consensus perspective, i.e., a reorg- and griefing-resilient mechanism, hence resources were spent to think about PBS designs. This is all the more critical since PEPC would likely need to support multiple deliveries, e.g., two partial builders delivering two partial blocks (see below).

On the willingness, while tail risks from unlimited commitment power may exist , it is also highly possible to deploy PEPC incrementally, starting from “a PEPCified implementation of ePBS” and adding more capabilities as needed over time.

Practically, this would look like including a generic “hook” for proposers to include additional block validity conditions, but initially restricting the hook to allow only for commitments to builders making the whole block. This differs from the ePBS approach of enshrining the commitment to full-block building directly into the consensus logic.

Functional diagram of a proposer hook

Overall, all claims around PEPC’s functionalities should be implicitly appended with “assuming this is useful and works at all”. That it reappears periodically in the conversation without much news about its progress signals at least some inclination towards reassessing the role of the proposer in the process of block construction, so while it might not ever be deployed in its full generality in the protocol, it still makes sense to see where that thread leads. We give in the next question more details on PEPC’s architecture, which hopefully makes a convincing case that it can be made to work.

H ow would it work?

This is not a precise spec, but the best mental model is along the lines of thetwo-slot design of PBS:

Commit phase: A proposer creates a commit-block containing

Consensus data: Attestations and other beacon block attributes.

Proposer commitments (PCs): A representation of commitments made by the block proposer. How they are written remains a big topic of discussion (see question below).

Payload template: The shape of the execution payload, with placeholders for expected builder items.

Reveal phase: Builder items are revealed, and mapped to placeholders in the block, if they satisfy validity of the PCs .

Taking full-block PBS as an example, builders communicate to the proposer the availability of some execution payload with a specific hash, revealing only the hash and some bid. The proposer makes a commitment to using some execution payload with a specific hash. During the reveal phase, the builder who previously communicated the availability of such a payload reveals the entire payload, filling the placeholder left by the proposer in the commit phase.

While full-block PBS is one example, a proposer could decide to segment their block between a top-of-block auction and a rest-of-block auction, potentially alleviating centralisation concernsfrom the presence of CEX/DEX arbitrage and OFAs in the same block (also discussed in Can we do “out-of-protocol PEPC”? (PEPC-Boost)). This is shown in the following example:

Example of “two-slot PEPC” where the proposer commits to using partial blocks from two builders A & B.

What are PEPC’s use cases?

We collect here a few use cases which may be offered by PEPC:

Full-block auction: This is equivalent to the current design of ePBS.

Partial block auction: The proposer could make part of the block while the builder could make another part, using any of the designs in Vitalik’s ethresear.ch post.

Parallel block auctions: The proposer could segment blockspace, auctioning off individual parts to individual builders. Note that this construction does not specify which type of auction must be run, i.e., the proposer could choose to run N N N individual auctions for each of the N N N partial blocks they are offering, or run a combinatorial auction where builders may bid on bundles of partial blocks. The proposer remains free to choose their desired allocation mechanism.

Slot vs block auctions: In all these designs, the proposer could commit ex ante to some specific builder block content, by committing to use only the builder (partial) block which hashes to some specific value, stating “I expect a (partial) block from builder B B B with hash h h h ”. The proposer could also choose to commit to a builder, without committing to the contents of the builder block, stating “I expect a (partial) block from builder B B B ”. For more background, see also Julian’s post on block vs slot auctions.

Future slot auctions: This is often a topic of controversy, so it remains to be seen if the proposer will always be known in advance, but today, a validator can look up the schedule of proposers for the current epoch and the next. If a proposer was able to register ahead of time their commitment to use a (partial) block from a specific builder, they could sell this right ahead of time too, akin to a blockspace future.

Inclusion lists: Inclusion listsdo not require PEPC to be enshrined, but the proposer hook could offer the ability to make a commitment equivalent to that of inclusion lists. Forward inclusion lists, which bind the next proposer instead of the current one, would need a special protocol allowance, giving to the current proposer the ability to make a commitment binding the next proposer.

Future protocol upgrades: It is clear that more and more duties will be expected to be fulfilled by builders, from Danksharding data to witnesses in Verkle triesto validity proofs of an enshrined zkEVM. Figuring out a future-proof generic scheme for the provision of consensus-critical items from third-parties may be valuable.

How does PEPC relate to Eigenlayer?

In the initial post, “Unbundling PBS”, PEPC was presented iteratively, starting with the problem of reflecting Eigenlayercommitments to the Ethereum consensus layer. More precisely, when validators enter commitments into Eigenlayer, Eigenlayer tracks a “virtual balance” for the validator, initially equal to the total stake that the validator has deposited in Ethereum Proof-of-Stake. When the validator fails to perform services adequately on Eigenlayer, their virtual balance is decreased (”slashed”), but the Ethereum PoS system still believes that the full amount deposited is currently at stake. This misalignment is cause for concern for multiple reasons, including the wrong perception that the PoS system is more secure than it is, and the potential for virtually slashed validators to still earn full rewards on their stake. Most of these concerns are addressed by EIP-7002, which allows smart contracts to perform consensus layer updates, for instance communicating to the consensus layer that the validator was virtually slashed on Eigenlayer, and thus re-aligning their virtual balance with their true consensus layer balance.

The larger concern expressed by the "Unbundling PBS" post is the economic bound to security that Eigenlayer offers: If the value at stake for a restaked validator is X, then the system is not economically secure if there exists an opportunity to deviate netting more than X to the validator. While there is no free lunch and the validator stake cannot simply be duplicated "ad infinitum" to secure unbounded amounts of value, " Unbundling PBS" concluded that for specific validator commitments, one could do strictly better than Eigenlayer.

To understand which proposer commitments can be enhanced by PEPC, it is useful to consider the use cases of Eigenlayer, as expressed by Sreeram Kannan (Eigenlayer founder) in a recent podcast episode of Bell Curve.

Bell Curve Podcast: The Intersection of Restaking and Liquid Staking | Sreeram Kannan - BlockworksSeason 5 | Episode 4In this episode of Bell Curve, Mike and Myles are joined by Eigen Layer Founder Sreeram Kannan. They dive into Eigen layer's supply & demand sides, its interactions with Ethereum restaking, and the team's vision for the end game of Ethereum LSTs and restaking.- -Timestamps:(00:00) Intro: Pre-Interview(01:33) Interview Start: Sreeram(02:19) High Level Eigen Layer Overview(13:59) Supply & Demand Sides of Eigen Layer(14:23) "Permissionless Ad"(15:18) Supply & Demand Sides of Eigen Layer(26:06) Eigen Layer & Ethereum Restaking(37:24) Eigen Layer Demand Side(45:00) Vote Extensions & Proposal Commitments(49:15) Ethereum Restaking Prediction(55:20) Role of Eigen Layer Governance in Restaking(1:12:53) Eigen Layer & Liquid Staking(1:22:03) Future of Validator Services(1:31:05) Recap: Post-Interview- -Follow Sreeram: https://twitter.com/sreeramkannanFollow Myles: https://twitter.com/MylesOneilFollow Mike: https://twitter.com/MikeIppolito_Subscribe on YouTube: https://bit.ly/3R1D1D9Subscribe on Apple: https://apple.co/3pQTfmDSubscribe on Spotify: https://spoti.fi/3cpKZXH- -Use code BELLCURVE30 to get 30% off Permissionless 2023 in Austin: https://blockworks.co/event/permissionless-2023- -Resources:Eigen Layerhttps://www.eigenlayer.xyz/https://twitter.com/eigenlayerhttps://forum.eigenlayer.xyz/Witness Chain (Proof of Decentralization)https://witnesschain.com/Vitalik Account Abstraction Talk EthCChttps://www.youtube.com/watch?v=iLf8qpOmxQc - -Disclaimer: Nothing said on Bell Curve is a recommendation to buy or sell securities or tokens. This podcast is for informational purposes only, and any views expressed by anyone on the show are solely our opinions, not financial advice. Mike, Jason, Michael, Vance and our guests may hold positions in the companies, funds, or projects discussed.https://blockworks.co/podcast/bellcurve/3c07eaba-2aca-11ee-a06f-8be44501cf72

According to Sreeram, there exists three types of use cases for Eigenlayer (timestamp 26:25):

Economic use cases: The users of the Actively Validated Service (AVS, a service provided by validators via Eigenlayer) care that there is some amount at stake, which can be slashed if the commitments are reneged upon.

Decentralisation use cases: The users of the AVS care that many independent parties are engaged in the provision of the service. A typical example is any multiparty computation scheme, where collusion between parties defeats the guarantees of correct execution, e.g., yielding them the power to decrypt inputs in the case of threshold encryption.

Block production use cases (timestamp 40:10) : Validators acting as block producers can make credible commitments to the contents of their blocks.

Clearly, PEPC is particularly suited for the third use case, moving the failure mode of proposer commitments from an optimistic setting (proposer can renege on their commitments, but will be financially penalised for it) to a pessimistic setting (the protocol will not let a proposer renege on their commitment, invalidating the block if it does not satisfy the commitments). It remains an open question whether PEPC can support or improve "economic" or "decentralisation" use cases.

PEPC generalises some of the PBS constructions, and provides more security to specific Eigenlayer use cases.

P.S: Mike and Myles open the discussion on PEPC near the end of the episode (around 1:44:10).

How does PEPC relate to SUAVE?

SUAVEis a project led by Flashbots, best understood in this context as a decentralised builder. SUAVE takes in user preferences, and outputs blocks to be proposed on various domains not limited to Ethereum. Between the user input and the block output, various parties engage in partial construction using programmable privacy (among other methods) to maximise user welfare and limit negative extraction by operators.

PEPC complements SUAVE by offering the possibility to make (more) credible commitments before block construction, i.e.,

committing to block specifications in an inviolable manner before the block is made. While SUAVE is expected to run with faster block times than Ethereum, allowing the emergence of fast games(from the perspective of Ethereum) played on SUAVE, it may be desirable to trade-off some commitment speed against commitment credibility. See also [SUAVE+PEPC] Reply to Xynfor more discussions around these ideas.

Another way to view this is from the perspective of the validator-as-market-maker between the protocol-as-gas-supplier and the users-as-demand-for-gas. With PEPC, the validator itself is able to enrich the supply of gas by segmenting the use of blockspace, before selling blockspace back to the users, which is made more valuable by the previous enrichment.

How does PEPC relate to ABCI++?

Cosmos researchers have long discussed the interplay between the application logic of a chain and the underlying consensus mechanism. With ABCI++, originally designed by Dev Ohja and Sunny Aggarwal, it is possible to add supplementary steps to the construction of a block, which require to be fulfilled before the block is considered valid, with inputs from the application logic. For instance, Vote Extensions enable validators of a chain to input their view of some object (e.g., bids of an auction, partial blocks, oracle messages, mempool contents) before the construction of the next block. The next block may then consume the signals reported by the voting validators to condition its construction.

x/builder, a proposal by Skip, takes advantage of ABCI++ to develop infrastructure for applications to express opinionated preferences with respect to block construction. The builder encodes these preferences and must satisfy them to provide a valid block, e.g., it must run an auction for the allocation of the top-of-block transaction before building the block.

While x/builder is deployed as a module of the chain it builds for, globally setting block construction preferences for all blocks of the chain, PEPC is a system of local decisions made by block proposers, i.e., a single block proposer is free to decide its own commitments for its own blocks. The un-opinionated nature of many Ethereum components may prevent or decrease the appeal of global settings such as x/builder's. For instance, in Ethereum, it is possible to hide many transactions as one single "top-of-block" (meta-)transaction, whereas the application logic of some given Cosmos appchain would limit the top-of-block allocation to a single transaction of some chain-specific transaction type (see more discussion in Can we do "out-of-protocol PEPC"? (PEPC-Boost)).

Is this ABCI++ for Eth?

How does PEPC relate to mev-capture? (mev-burn or mev-smoothing)

Understanding PEPC in the larger context of ePBS research is valuable. In particular, we must ask how would PEPC mesh with other upgrades related to ePBS, such as mev-capture.

Mapping ePBS components, by Mike Neuder at pbs.day

We offer here a possible scheme to make mev-capture work with PEPC. What is needed for mev-capture is aconsensus bid, an objective value of the execution payload's revenue to the proposer, obtained from a permissionless auction. But how should one run this auction when PEPC instantiates multiple parallel auctions for blockspace allocation? Should each auction signal a partial consensus bid, with the final value arrived at by summing all the partial consensus bids together? This seems unwieldy and not particularly generalisable.

The trick is to use Domothy's parallel proposers idea. Each slot, 16 proposers are chosen to make a commit block. Along with their commit blocks, proposers specify a value that they are willing to forward to the protocol. Attesters select the highest proposer bid, and this proposer's commit block is enshrined by the fork choice, after which the committed builders proceed with the reveal phase (there may be several builders if the proposer is expecting multiple partial blocks). This trick neatly avoids the protocol receiving an individual signal of value for each of the commitments entered into by the proposer, and instead tasks the proposer with surfacing credibly the total value they expect to receive from their block.

Note that to avoid proposers forwarding their consensus rewards to the protocol too in a race to the bottom to win the right to make the block, we should split the duty of making a beacon block with consensus data from the duty of making the execution payload, i.e., move towards slots built in three stages: beacon block, commit block, reveal phase.

Can we do "out-of-protocol PEPC"? (PEPC-Boost)

Are there ways to try PEPC-type allocation mechanisms without enshrining PEPC, e.g., PEPC-Boost? Could relays

synthesise a version of PEPC, or would there need to be an entirely different architecture than MEV-Boost?

We propose here a draft specification of PEPC-Boost, requiring no changes to MEV-Boost. This implementation instantiates a parallel block auction splitting the block between a top-of-block slot and a rest-of-block slot, akin to the example pictured above in the How would it work? section. The intent is to separate CEX/DEX arbitrages from the rest of the block, and increase competitivity/reduce centralisation forces of the resulting system by unbundling the whole block auction into two separate “lanes”.

Here is a simple spec for the system:

A new relay known as PEPC-Boost opens. PEPC-Boost has two builder endpoints:

TOB endpoint (top-of-block): Accepts a single transaction , along with a bid

ROB endpoint (rest-of-block): Accepts a list of transactions, along with a bid

The PEPC-Boost relay constructs a block with the best bid received from the TOB endpoint and the best bid received from the ROB endpoint. The block is signed by the relay (now acting as a builder) and the header of the block is sent to the proposer along with a bid equal to TOB bid + ROB bid .

Obviously, this is too unspecified to be efficient. Bundles placed on top of the ROB portion may have conflicting state accesses with the TOB transaction, which would not have happened if the whole block had been built by a single logical entity (a builder). Additionally, the semantics of “a single transaction” received by the TOB may be gamed by searchers who pack multiple intents in a single meta-transaction, inflating the value of their TOB bid.

To start with, the PEPC-Boost relay could specify that it only accepts one single swap for the ETH/USDC pool on Univ3 (reputed to be the pool with the most arbitrage caused by external price movements, i.e., the most LVR). Then, ROB builders can build knowing that they should not include transactions from arbitrageurs who would typically do the CEX/DEX trade at the top of the block. Additionally, the TOB transaction slot cannot be gamed by meta-transactions.

It is then possible to extend the TOB slot with more transaction slots, e.g., the TOB slot has three virtual slots, one for each of the three pools with the most arbitrage, and the PEPC-Boost relay fills each slot with the best bid it has received for each pool. The relay could expose an endpoint which returns which slots exist for which pools, and update this endpoint dynamically.

What is left to work on?

Work on ROP-4: “Rewrite” vanilla IP-PBS in PEPC framework is under way, with contributor Diego exploring the design space and specifications.

Additional research on the use cases of PEPC and their representation as commitments would be valuable. Do commitments “live” in a different state than the Ethereum execution layer? Can they be represented with EVM instructions? etc.

An implementation of PEPC-Boost, thinking through the option of using optimistic relaying.

Further use cases, e.g., on proposers committing to specific orderings.

Many thanks to Diego Estevez, Thomas Thiery, Terence Tsao, Julian Ma, Davide Crapis, Mike Ippolito, Alex Stokes, Mike Neuder, Quintus Kilbourn, Sam Hart, Xyn, Sreeram Kannan, Hasu and Anders Elowsson for comments and discussion.