https://bittokoin.substack.com/p/block-builder-decentralization-is?r=tnhy7&utm_campaign=post&utm_medium=web

Introduction

In early 2022, there were discussions about the potential centralization of block building and the implications of MEV and block sequencing post-PBS (Proposer Builder Separation). I personally resonated with this concern and published an article on the risk of centralization. At the time, the community had mixed views on the issue, with some accepting the idea that Ethereum could have a decentralized validator network with a more centralized block builder, as outlined in the "Endgame Paper" by Vitalik Buterin. However, these concerns were eventually pushed to the back burner and faded from public discussion. The last SBC conference, though, brought these concerns back to the forefront, as Vitalik presented ways to decentralize block building (Jon Charbonneau wrote a great report on the pitch here). Since then, I've been considering potential designs for decentralized block builders and wanted to share my thoughts and the challenges that come with decentralizing block building.

How centralized is block building currently and why do we need to decentralize it?

Ethereum strives to be the most censorship-resistant, permissionless, and trustless blockchain and any sort of centralization factor could potentially harm those properties. I will lay out major concerns and will also explain why some matter and others don't.

Exclusive Orderflow: There have been active discussions by Flashbotsand the communityabout how exclusive orderflow could potentially lead to a single block builder monopolizing winning bids and orderflow. The assumption is that a builder will provide incentives, such as MEV sharing, private transactions, and MEV backrun guarantees, to gather orderflow that others do not have access to, thereby producing higher-value blocks that consistently win the auction. However, I believe it would be too broad to categorize all types of incentives that result in exclusivity under one category of concerns, so I have split them into the following sub-categories: Exclusive Orderflow via MEV redistribution and via MEV backrun guarantee & private transactions. It's important to note that these sub-categories are not mutually exclusive and have different implications.

Via MEV redistribution: This concern will likely become less of an issue as more orderflow marketplaces launch in the coming years. As competition increases among marketplaces, they will need to offer more MEV rebates through orderflow auctions to attract users and their transactions. This competition will ultimately influence where users choose to submit their transactions.

Via private transaction & MEV backrun guarantee: These are optional features that may be preferred by some individuals. However, the use of this type of exclusive orderflow could lead to orderflow centralization if it were to grow on a larger scale. To mitigate this, a decentralized block builder that consistently proposes higher-value blocks would be beneficial.

Censorship: Insufficient decentralization of the block builder could result in a subset of builders colluding and censoring transactions. While solutions like crList, Geth's native block building option, andencrypted mempoolaim to mitigate this risk, they also present challenges such as the partial enshrining of account abstraction, the need for infrastructure to enable state access by stateless validators, and security concerns around existing privacy-enabled technologies. To gain a deeper understanding of the current block building landscape, here are some stats on block production. Based on the block success rate of all builders in December 2022, the top 5 builders are the following:

Table of average success rate of top five block builders

The top three block builders collectively controlled 60% of the total number of successfully committed blocks in a given month, representing a significant amount of trust placed in their ability to not collude or censor transactions. To counteract this, implementing a framework to decentralize the block building process would raise the cost and barrier for collusion and censorship, resulting in a more trustless block production system.

Centralization due to network effect: In imperfect market competition, it may be suggested that one builder would naturally outcompete others by constructing higher-value blocks and attract more orders without the need for direct incentives, resulting in centralization. However, this is not always the case in reality.

Graph of block builder success rate

This graph illustrates the block success rate over a period, both before and after The Merge. Notably, it can be seen that Flashbots (represented in dark blue) has decreased and reached an equilibrium point similar to that of other major block builders, rather than completely dominating block production. This contradicts the proposed centralization due to the network effect, and there may be several explanations for this observation.

Competitors' exclusive orderflow: Other competing builders may have exclusive orderflow (private transactions etc.) that allows the production of higher-value blocks.

Orderflow congestion: Too many orders wanting to be included in the block via Flashbots results in latency competition. This may force users to put transactions across multiple builders to increase the chance of getting committed in a block, especially in times of high traffic.

OFAC compliance: Flashbots are OFAC compliant while some others aren't. Users may decide to pick builders based on some regulatory concerns.

Flashbots open-sourced the builder and relay code to invite more competition.

For the reasons above, I believe the builder centralisation by network effect was mitigated to a certain extent. However it still remains an issue as shown in the previous table, and I believe it is a problem that needs more attention and awareness in the community.

What do we decentralize?

The crucial aspect that requires decentralization is the authority over block production. Instead of entrusting a single entity with complete control over the construction of blocks, there should be a framework that allows for the participation of multiple entities or individuals in the process. Currently, there are two proposed approaches to achieve this, and it should be noted that these approaches are not mutually exclusive.

Proposer participation in block building

Decentralize block builder via new architectures and algorithms that allow coopetitive block production (massive design space)

There are still ongoing discussions around technical feasibility and challenges around the proposed approaches, and I would like to discuss and share my thoughts here.

Approach 1: Proposer participation in block production

There are currently several ways to enable proposer participation in block production as outlinedhere, but I believe that the Eigenlayer approach offers a cleaner solution. Before delving into the specifics of Eigenlayer, here is a brief overview: it is a set of smart contracts on the Ethereum blockchain that allows ETH staked validators to opt-in to providing security for new services by imposing additional slashing conditions on their staked ETH. For a more comprehensive understanding, you can read here.

In terms of how a builder currently interacts with a proposer during block production, the Proposer Builder Separation (PBS) system was implemented. In this system, a builder can submit a block to relayers who will then verify the block's correctness and relay it to a proposer. This system aims to prevent censorship by the proposer, as the content of the block is only revealed to the proposer once the highest bidding block is selected and the signed header of the corresponding block is relayed back to the relayer. The escrow, who is responsible for ensuring the data availability of the committed block, will then reveal the data (transactions) to the proposer, who then sends the block to the rest of the network. The following diagram illustrates the current process.

Diagram of current block building process by Eigenlayer

The Proposer Builder Separation (PBS) system increased the censorship-resistance of Ethereum by limiting the authority of validators to the proposal of higher-value blocks. Eigenlayer, however, proposes an MEV management framework called MEV+ that allows block proposers to include additional transactions on top of existing blocks. This is achieved through the imposition of additional slashing conditions.

Currently, block proposers are subject to only one slashing condition which prohibits the simultaneous proposal of two blocks by the same proposer. With the MEV-Boost+ framework, proposers will also be subject to slashing if they modify or fail to include the committed "builder_part" of the block when proposing a new block with added "proposer_part" transactions. This effectively prevents proposers from attempting to modify or remove the block builder's transactions. The following diagram illustrates the MEV-Boost+.

Diagram of proposer participation in block production by Eigenlayer

This mechanism is also effective when used in conjunction withcrList, which is a list of transactions that proposers must include in the block. This helps to reduce the potential for censorship of transactions at the builder level. While other solutions, such as pre-commitment to Merkle root or KZG commitment, have been proposed to allow proposer participation in block production, EigenLayer offers a simpler alternative that does not require additional computational resources from proposers and enables participation in block production simultaneously.

Overall, this approach is simple and enables further decentralization of block production by introducing proposers into the process, hence sharing the authority over the block with more entities. However, this approach is limited in impact as it fundamentally still relies on a centralized block builder and does not fully prevent censorship that arises from builder centralization. Hence we have the second approach.

Approach 2: Decentralizing Block Builder

Designing a decentralized block builder is a fascinating area of exploration and teams like Flashbots are already experimenting with different designs. When considering how to decentralize the builder, it is important to be aware of the following challenges:

MEV-stealing: Builders may access the information from bundles submitted by searchers and steal searchers’ MEV. To prevent this, the privacy of the submitted bundles and transactions should be protected in design.

Sub-optimal MEV: Decentralizing the builder should ideally be achieved while maintaining an optimal MEV. Some designs may introduce inefficiencies in block building due to decentralization and result in less MEV from the block, which in turn leads to less competitive blocks. But one can also argue that sub-optimal block production is acceptable as long as the decentralized builder attracts more orderflow than the rest.

Competition against centralized builders: Decentralized builders need to be competitive against centralized builders in producing MEV. And decentralized builders' goal would be to aggregate as much orderflow as possible to compete with the concentration of orderflow among centralized builders.

Latency: Block production is time sensitive, and depending on the design, there could be a significant latency issue.

Jurisdictional coverage (censorship-resistance): Decentralized builders should be distributed across many jurisdictions so that they can resist jurisdictional censorship (like OFAC). Regulation is still a grey zone across most countries and would not want to risk the builder network being taken down by a single regulatory entity.

And many more concerns that are design-specific.

With those challenges in mind, I would like to discuss proposed decentralized builder designs across the research community and dive into some potential issues around them. Currently, there are two main types of decentralized builders:

Searcher-Aggregator model: Searchers submit bundles and aggregators will construct the block from them without knowing much/any information about the submitted bundles.

Slot auction model: Block space auction is conducted sequentially and the block is constructed progressively by multiple builders, hence no aggregator.

The following proposed designs are variations of either of the two types.

Design 1: Searcher-Aggregator model with TEE/TPM

This approach requires an aggregator to use a Trusted Execution Environment (TEE) or Trusted Platform Module (TPM) to ensure the privacy of the received bundles and thus preventing MEV stealing. For more information on the differences between TEE and TPM, check here. Following is an illustration of the design (TPM and TEE would be interchangeable in this context):

Diagram of TEE based block builder by Vitalik

Flashbots have released a progress report on their experience and learning from running Geth in SGX, a type of TEE developed by Intel. While there were numerous technical challenges, they have successfully run Geth using encrypted swap space in SGX with Gramine, one of the library OS designed for SGX, with 500GBs RAM, 1TB of SSD swap, and 64GBs of protected memory. Here are some key takeaways from this experimentation:

Running Geth within SGX is possible but is resource-intensive and time-consuming, requiring a large amount of memory and 3 hours of startup time for storing and encrypting the on-chain data.

Encrypted swap space provides good performance, but may still be vulnerable to information leakage via side-channel attacks, covert-channel attacks, and other programming errors.

Needs to make a tradeoff between performance and resources. For example, a less resource-intensive approach will have poorer performance and potentially have a worse information leakage problem.

Additionally, here are other aspects to consider with the usage of SGX:

There are ways to mitigate the information leakage problem but some approaches may result in reduced performance. It definitely requires more trials and errors to hone out the best-performing framework to run SGX without requiring too much resource.

Complex setup + scarcity of SGX-compatible chips lead to a high barrier of entry. Cloud providers may provide accessibility to SGX and can be a temporary solution, but cloud operators would be a significant centralization factor over long term.

Information leakage via SGX is still an issue. In case people discover additional exploitable vulnerabilities in SGX, aggregators should immediately perform their own TCB recovery (a process designed to re-instantiate the SGX environment) instead of waiting on response by Intel, which would likely take a long time.

Decentralized builders, in principle, should welcome and incentivize the aggregation of order flow, unlike centralized builders. The searcher-aggregator model, assuming zero information leakage, should have trust-minimized security, trusting that encryption and SGX work perfectly. However, it could have a heavily trusted liveness where there is only a small number of entities running the aggregator, which is possible given that bundles will naturally aggregate to the most successful aggregators. In this case, order flow/bundle concentration will not lead to a censorship issue, but possibly a liveness issue.

If the number of entities running the aggregators is low, there could be insufficient geographical distribution leading to a skewed jurisdictional coverage, which will cause the network to be more vulnerable to the regulatory censorship.

While a TEE-based solution like SGX seems to be the more practical approach to builder decentralization right now, it still faces with many technical challenges that need to be overcome.

Design 2: Searcher-Aggregator model with Threshold Encryption and ZK-SNARK

This is another Searcher-Aggregator model where encryption is applied on the bundles instead of the aggregator. Here is a rough illustration:

Diagram of TE and ZKP-based block builder

This design has the potential for collusion between aggregators and proposers. The threshold encryption ensures the privacy of the bundle only up to the point before computing the state root, and the aggregator, who computes the state root, requires access to transaction information or state updates. This access could enable MEV stealing by tracking down the corresponding transactions. The design eliminates the need for TEE/TPM, but it fails to prevent such collusion without adding additional complexity, such as requiring the proposer to commit to the block before decryption that allows computing the state root. Here are some concerns with such a design:

Early commitment to the block by proposers could be achieved via re-staking infrastructure, like Eigenlayer, but would incur extra operational cost since the ETH stakers need to be sufficiently incentivized to opt-in to adding relevant slashing conditions.

Proposers may miss out on higher-value blocks by committing to a block early.

Added computational overhead and latency from generating ZK-SNARK and threshold encryption, making this model potentially unfeasible in practice.

A question on who would be the qualified keyholders for the threshold encrypted bundles. If searchers hold the keys, attackers who pretend to be searchers can block the decryption and delay the block production. Conversely, aggregators cannot hold the keys as each of them is competing for block production and may be incentivized to obstruct other aggregators. This may require a third party that is neither a searcher nor an aggregator and introduces additional trust assumptions that may discount the trustless/trust-minimized property of being a decentralized builder.

Same point on jurisdictional coverage as mentioned in the first design.

Design 3: Slot Auction model via chaotic iterative search

This design allows multiple block builders to participate in block production without needing an aggregator. Here is a rough illustration (excuse my drawing):

Diagram of slot auction based block builder

The proposed design aims to allow multiple block builders to participate in constructing a single block by partitioning the maximum gas of the block into n slots given x number of builders, where n = f(x) and n<MAX_THRESHOLD. Builders in the network start bidding for the partitioned block space, with the highest bidder filling that portion of the block. The first winner of the bid also becomes the generator of the block. The bidding continues for the remaining slots, with new bidders receiving the work-in-progress block and building upon it. The bidding cycle ends when there are no more bidders or the block is full. The builder who holds the block at the end commits it to the proposers. This design divides the traditional task of a single builder into smaller pieces by performing a MEV Auction (MEVA) at each slot and applying threshold encryption for privacy. The main advantage of this design is that it eliminates the need for a centralized aggregator and allows builders to be distributed globally, providing distributed jurisdictional coverage. However, here are potential challenges with this design:

When to reveal the encrypted slot would determine how efficient the auction will be (closer the bidding price to the corresponding MEV, the more efficient the auction is, since the MEV would be the “real” price of the slot). If the encrypted slot N-1 is revealed before the auction of slot N, then the auction will be efficient because builders would be able to bid based on their expected MEV. However, this could have latency issues in practice where strictly sequential auction and decryption of the slots may take some time. Therefore, the other option is to let builders bid for the rights to build the block for all the slots beforehand, and decrypt the block sequentially as builders fill the slot. Since builders do not know the MEV for the slot prior to bidding (need to know the transactions in slot N-1 to determine the MEV for slot N), the auction will likely be inefficient, a builder may even accidentally overbid for the slot due to lack of realized MEV. Though, this problem could be mitigated via statistical analysis of the historical bidding price of the slots at the given position in the block.

To prevent builders from modifying the transactions in previous slots, other builders have to act as witnesses and provide some form of fraud-proof.

Centralized builders could steal some MEV as the slots are decrypted sequentially. If users submit orderflow only through this slot-auction-based builder network, MEV stealing would be less concerning, but if orderflow is submitted across both the centralized builder and this builder network, then MEV stealing could be problematic.

Malicious attackers, like a competing centralized builder, could disrupt the network by winning the bid and not building the block, hence temporarily halting the block production. And with enough such attackers, this builder network could become dysfunctional (unable to produce any block). There needs to be a mechanism to ban malicious builders from the network and have a backup plan on the unfilled slot.

Design 4: Sequential Slot Auction with proposer commitment

This is another approach that uses the slot auction model similar to the third design that does not have to rely on an aggregator. The main difference here is the enforcement of proposer commitment via Eigenlayer for each filled slot. Instead of committing the entire block to the proposer at the end, each slot will be committed by the proposer sequentially, and Eigenlayer could be used to avoid protocol-level changes to Ethereum. Since the proposer commitment comes right after the slot is revealed, there is less concern about the builder modifying transactions in the previous slots. However, most other issues mentioned for the third design would be applicable to this design as well, with the exception that the latency could be worse due to the additional sequential proposer commitment in the process.

Comments on the future of block building

Decentralization is a crucial aspect of blockchain technology, and it is important to ensure that block builders are also decentralized. In this article, we have discussed a few possible approaches for decentralizing block builders. However, this is still an open design space for the research community to explore and collaborate on.

Designing a decentralized block builder that is also competitive would require a lot of experimentation and testing. And it is already work-in-progress as teams like Flashbots explore the limit of SGX and other privacy-enabled technologies. However, realizing the full potential of these technologies would require even more innovation and research.

Overall, decentralizing block builders is an ongoing challenge that requires the collaboration and expertise of the wider research community beyond crypto. Through experimentation, testing, and innovation, we can work towards creating a decentralized and competitive block builder that enhances the censorship-resistance of the network.

Special thanks to Hasuand Justin Drakefor their review and feedback!