Special thanks to @JustinDrake and @pseudotheos for all the discussion and theory-crafting surrounding this topic. Personal opinions are expressed in the first person singular.

# Introduction

This document aims to explore the idea of letting multiple proposers compete every slot, with the winner being the one committing to burning the most ETH. A possible implementation is presented, and the incentives are played out to argue this would result in most (if not all) of the MEV being burned.

The core idea of this proposal is to auction off the “right to build a block” on the consensus layer through an in-protocol burn auction. Once a winner is selected, the execution block they propose will be one that provably burns at least as much ETH as their bid. The idea is that the highest bid will naturally approach the maximal extractable value (MEV) in the block, therefore most of the MEV should be directly burned.

# Advantages

- Reducing incentives for proposer centralization - proposing more blocks simply translates into burning MEV more often, with limited personal gains
- Making validator rewards much smoother, very close to the issuance guaranteed by the consensus layer.
- This means the blockchain’s security budget will be paid by stable intrinsic rewards rather than uncontrollable extrinsic rewards, which mitigates many incentive-related problems.
- This means the blockchain’s security budget will be paid by stable intrinsic rewards rather than uncontrollable extrinsic rewards, which mitigates many incentive-related problems.
- This proposal is essentially a simpler form of MEV smoothing, where the MEV is “smoothed” across all

ETH holders, whether or not their ETH is staked.

- It lets Ether capture the value that is otherwise extracted from the economic activity happening on-chain.
- MEV burning enhances the economic attributes of Ether as an asset:
- It makes ETH the

currency of the block building market, similar to how EIP-1559 made ETH the

currency for the gas market, where out-of-protocol gas markets in other currencies become largely unviable.

- It also enhances and protects ETH’s monetary premium: any

opportunity that extracts value would result in the burn of an amount of ETH of equivalent value, even if the MEV opportunity doesn’t involve ETH at all

(e.g. if a given token is paired with USDC and there is an arbitrage opportunity for 100 USDC, then 100 USDC’s worth of ETH will have to be burned for this arbitrage to be exploited).

- It makes ETH the

currency of the block building market, similar to how EIP-1559 made ETH the

currency for the gas market, where out-of-protocol gas markets in other currencies become largely unviable.

- It also enhances and protects ETH’s monetary premium: any

opportunity that extracts value would result in the burn of an amount of ETH of equivalent value, even if the MEV opportunity doesn’t involve ETH at all

(e.g. if a given token is paired with USDC and there is an arbitrage opportunity for 100 USDC, then 100 USDC’s worth of ETH will have to be burned for this arbitrage to be exploited).

# High-level overview

Every slot, each validator has a small probability of being an eligible proposer

. The probability is chosen such that there is on average a target number of eligible proposers per slot who are allowed to broadcast a bid for the right to propose the next block.

The bids made by eligible proposers on the consensus layer constitute the baseline for how much ETH should be burned by whichever execution block ends up getting included on chain.

As is the case with the competition between block builders, each proposer should be willing to bid up to the amount of MEV for the right to propose the next block, meaning a majority of the MEV will end up getting burned. The protocol itself makes no attempt to quantify MEV in any way, other than by letting this burn auction take place every slot.

Concretely, this proposal includes increasing the slot time to 16 seconds, with the extra 4 seconds being a “bidding period” at the start of the slot. During the bidding period, eligible proposers broadcast their bids, committing to a specific hash of an execution block that will have to provably burn an amount of ETH that’s least equivalent to the proposer’s bid.

After the bidding period ends, the rest of the slot proceeds as usual, and the highest bidder is expected to reveal their block. Other bidders can

reveal their blocks as well, but they will be quickly ignored as soon as a higher bidder reveals their block.

## Execution layer changes:

- Naming convention change:

fee_recipient

header is renamed to builder

- New block headers:
- builder_fee

: How much the builder is willing to pay

- builder_signature

: transactions_root

signed with the builder

's private key

- builder_fee

: How much the builder is willing to pay

- builder_signature

: transactions_root

signed with the builder

's private key

-
    - New block validity checks:
- Verify that builder_signature

is valid

- Verify that builder's balance >= builder_fee
- Verify that builder_signature

is valid

- Verify that builder's balance >= builder_fee
- New block validity checks:
- Verify that builder_signature

is valid

- Verify that builder's balance >= builder_fee

- Verify that builder_signature

is valid

- Verify that builder's balance >= builder_fee
- New state transition:

After the block’s execution, subtract builder_fee

from the builder’s balance

- (Optional) New EVM opcode

: BUILDERFEE

, similar to EIP-3198’s BASEFEE

opcode

## Consensus layer changes

Every slot, we want a given validator to have a probability \frac{B}{N}

of being an eligible proposer, where B

is the target number of eligible proposers, and N

is the number of active validators.

As outlined by Vitalik here, a simple implementation for such an eligibility criterion could involve checking that the hash of the RanDAO reveal is less than \dfrac{2^{256}×B}{N}

for a given proposer to be easily verified as being eligible.

Slot times are increased to 16 seconds, with the first 4 seconds being the “bidding period”. During this period, nodes will receive bids from eligible proposers, which are tuples of the following elements: (proposer_index, randao_reveal, block_hash, amount, signature)

Nodes can easily verify the validity of a given bid:

- Check that the bidding proposer has the right to bid for that slot, i.e. the hash of randao_reveal

is lower than \frac{2^{256}×B}{N}

- Check the randao_reveal

against the proposer’s public key

- Verify the bid’s signature to confirm it originates from the proposer
- Verify that no different bid has already been received from the same proposer

At t=0, nodes will be actively listening for valid bids and will be relaying them to each other, keeping track of every valid bid that they heard.

At t=4, nodes stop listening to bids, and the rest of the slot proceeds normally, with the exception that a block can only be proposed by an eligible proposer who has previously made a valid bid. In other words, even if a proposer is eligible, if they never submitted a bid, any block they propose will be ignored.

From t=4 to t=8, multiple blocks may be revealed at the same time by eligible proposers who previously made a bid, keeping track of the best block (best_block

). For every new block

heard, nodes perform these quick checks in order:

1. Make sure that a valid bid was heard from the proposer during the bidding period
2. If a best_block

is known: 1. Check that the block’s builder_fee

is greater than or equal to the best_block's

1. If they are equal, check that the hash of the randao reveal is lower than best_block's
2. Check that the block's builder_fee

is greater than or equal to the best_block's

1. If they are equal, check that the hash of the randao reveal is lower than best_block's
2. Check that the block's builder_fee

is greater than or equal to the amount that was bid by the proposer

1. Check that the block's hash matches the one that the proposer committed to with their bid

If any of these checks fails, the block is ignored. Otherwise, the rest of the usual block validation rules apply. Assuming the block is valid, it becomes the new best_block

against which other incoming blocks are compared. Naturally, nodes will only relay the block they know as best_block

to each other.

From t = 8 to t = 12, the slot's validator committee will attest to what they know to be best_block

, and from t = 12 to t = 16 is the usual attestation aggregation period.

# Rationale

## Execution layer changes

Renaming fee_recipient

to builder

is a matter of reflecting reality more accurately, since that's the address that will have to pay the ETH that gets burned, and this naturally gravitates towards the role of a block builder. Everything else otherwise remains the same with respect to block.coinbase

, where priority fees go, etc.

Forcing the builder

to sign the block's transactions_root

header is to prove ownership over the address that will pay the builder_fee

. Such a proof of ownership is not currently necessary, as the fee recipient's balance can only ever increase. Signing transactions_root

is the natural choice, as this header is fully binding to the transactions included in the block, which will naturally be tailored to favor the block's builder in terms of MEV extraction.

Another subtle design choice is whether to validate that the builder's balance is higher than the builder fee before or after transactions in the block have been executed. If we check it before transactions are executed, we are forcing builders to have enough ETH ahead of time, letting them get compensated with the MEV they earn from the block. Such a scheme works fine, but it has the downside of disfavoring smaller builders by pricing them out of high MEV opportunities when they could otherwise compete.

If we instead check the balance after the block is executed, the builder's balance can start with 0 ETH, extract any amount of MEV, and then use it to pay the fee. The downside is forcing nodes to execute an entire block before

## Forcing one bid per eligible proposer

In a simple scheme where a proposer can drop their previous bid and bid with a higher amount, the process would start with low initial bids (from proposers hoping to collect most if not all of the possible MEV) and quickly turn into a series of escalating bids of slightly higher increments, until finally the highest bid is the one that's closest to the possible MEV. While this process would still converge close to the MEV available, it results in much higher bandwidth requirements than necessary.

Instead, if a single bid is allowed per eligible proposer, the best strategy to maximize the probability of being the one

collecting the proposer reward is to immediately bid as close to the MEV as possible, knowing that other proposers will do the same. In other words, making bids final renders bidding much lower than the MEV a risky strategy. The reality of PBS means that the bids from builders translate one-to-one with the bids made from proposers

## Colluding strategies

A form of collusion could consist of a scheme where eligible proposers all signal to each other that they will bid 0 ETH and share the MEV amongst each other rather than burn it. It only takes a single defector to defeat this scheme: Bidding 1 gwei would allow the defector to collect all the MEV for themselves (minus 1 gwei). The presence of a second defector would be enough to ensure the desired level of competition that drives the highest bid to be close to MEV.

In fact, the single defector has no way to know if a second defector exists until they make their bid. This means that the 1 gwei bid is far from guaranteed of winning all the MEV. Given this, the defector's strategy is still

to bid close enough to the MEV to outbid other potential defectors, otherwise they get nothing.

A second, stronger strategy would be if a single entity controls enough of the validator set to control all B

eligible proposers in the slot. In this case, defecting doesn't happen as eligible proposers are all controlled by the same entity seeking to pocket all the MEV.

But again, not only is this scenario extremely unlikely (eligible proposers are essentially a random sample of the whole validator set), it is also impossible to know for sure if you do

control all eligible validators. B

is merely the average number of eligible proposers per slot, there could easily be a $(B+1)$th eligible proposer making an honest bid and ruining the whole plan.

## Forcing eligible proposers to commit to a block hash

This has two benefits: preventing MEV theft, and nullifying time buying attacks from late reveals. If eligible proposers were only bound by the amount of ETH they're willing to burn, a dishonest strategy could consist of constantly outbidding the highest bidder by 1 gwei, waiting for their block to be revealed and stealing the MEV. An additional time-buying attack could happen by getting up to 4 seconds of extra MEV for the additional cost of 1 gwei.

Forcing validators to commit to a block hash nullify both concerns, as the best strategy becomes the one employed by honest and lazy proposers:

1. Commit to one and only one block, bidding as high as possible to the block's builder_fee

, naturally the block you commit to will itself have to be one that has a builder_fee

as high as possible.

1. Reveal the block immediately at t=4. Waiting becomes a risk (not being seen on time) with no added benefit.
2. If a block with a higher bid is revealed, there is no point revealing the block at all, given that nodes will immediately dismiss it.

Note that time-buying attacks are not entirely mitigated: There are still incentives to wait, they're merely relocated to the bidding period. A bid at t=2 has 2 extra seconds of MEV compared to a bid made at t=0. However, this is still preferable to the outcomes of the time-buying strategies that occurs today:

- In the status quo: when a time-buying block is proposed too late, it risks being missed by validators, which is bad for the network's liveness and user experience when slots are missed
- Under this proposal: time-buying during the bidding period is a risk for the bidder only: Waiting too long can result in the bid not being on heard on time, rather than the block, thus favoring timely bids. If a time-buying bid is not heard on time, there will still a block being proposed (expected at exactly t=4), just from a different bidder.
- If waiting for more MEV before bidding is profitable, more eligible proposers will do it, simply resulting in more MEV being burned rather than extracted.

## Allowing multiple block reveals

In previous iterations of this proposal, the highest bidder was elected as the single leader of the slot, meaning blocks proposed by other bidders after t=4 were categorically ignored. This had the advantage of keeping the bandwidth requirements equal to the ones we have today: A single block would be revealed per slot to the rest of the network.

Unfortunately it also paved the way to a fatal liveness attack: Any eligible proposer could easily commit to a very large bid,

e.g. 100 ETH higher than the highest honest bid, with no intention of actually revealing any block, resulting in a missed slot. Making this a slashable offense is not reasonable, as we might be punishing a proposer who had a legitimate issue when the time came to reveal the block.

The extent of such a liveness attack is amplified by the exponential nature of the probabilities at play: An entity controlling a faction p

of the validator set has a probability of 1 - (1-\frac{B}{N})^{pN}

of having at least one of their validator be an eligible proposer every slot.

With an otherwise reasonable value like B=64

, controlling 1% of a validator set of 450,000 validators means there is a probability of 47% of having at least one eligible proposer every single slot

. With 2% of the validator set, the probability already jumps to 72%. At 5%, it's already game over with a probability of 96%, and the attacker can force nearly every slot to be missed. Given that there is essentially no recourse against this attack, the idea of the highest bidder becoming the single leader of the slot is a non-starter.

Instead, the block reveal scheme proposed above mitigates the attack entirely: If an eligible proposer makes a dishonest outsized bid, the next highest bidder will notice that no such block is being proposed (e.g. at t=5) and have an incentive to reveal their own block, rendering this form of liveness attack ineffective. This turns the requirement of controlling at least one eligible proposer per slot to requiring the control of all

of them, which is extremely unlikely as argued previously.

As far as bandwidth/execution load on nodes are concerned, the worst possible case is having nodes process all revealed blocks, i.e. if the node receives them in ascending order of their bid. In reality, the average case would be a much lower value:

- Due to the nature of gossiping only the block with the highest bid known, I believe nodes would quickly converge to the best block being relayed and every other block being ignored
- The preliminary checks are very quick, consisting only of checking the block's header to see if it can safely be ignored
- Eligible proposers can clearly see when they've been outbid, saving them the trouble of revealing their block in the first place (except, as mentioned, when the highest bidder doesn't reveal their block)

# Considerations

## Eligible proposer secrecy

In the proposed scheme, every validator knows whether or not they are an eligible proposer at the start of the slot, but no one else does. This is very similar to Vitalik's Secret Non-Single Leader Election proposal, except that eligible proposers reveal themselves when they broadcast their bid, rather than when they reveal their block.

This means in theory, the highest bidder could be the target of a DoS attack, leaving the second highest bidder be the one proposing the slot. In practice however, the profitability of such an attack is heavily mitigated by the fact that most of the MEV is burned rather than handed to the block proposer. This second highest bidder still had to outbid everyone else after they took down the highest bidder, otherwise another block would have been proposed. Meaning even if their DoS attack works, they still had to make their own bid that was reasonably close to the MEV opportunities of the block.

Nevertheless, full secrecy of bids could potentially be enforced at the cost of additional complexity by using zero-knowledge proofs.

## Proposer-Builder Separation

While this proposal looks a lot like it, it's not quite PBS. The way to view this proposal is to consider the existing competition between block builders (each one fighting for their block to be the one getting settled), and noticing that the network could easily benefit from an additional layer of competition, this time between proposers, each one fighting for the reward for proposing a block. Simply put, this proposal removes the monopoly that a single validator has over the right to propose a block.

It is still interesting to explore the implications of out-of-protocol PBS such as the existing MEV-Boost infrastructure under MEV burn: Builders still ultimately rely on proposers, so the amount they're willing to pay for inclusion translates one-to-one into the amount proposers are willing to bid, which will be burned when a winning block is included. Let's see why:

#### Proposer strategy

With MEV burn, proposers lose their monopoly over block proposal, so their goal is no longer about maximizing MEV bribes, it's now about maximizing their chances to get the proposer reward.

Proposers connected to MEV-boost relays will see a list of block hashes and corresponding builder_fee

, and their best strategy is very simple: pick whichever one has the highest builder_fee

, because that's the absolute highest they can bid:

- Any higher bid will make nodes reject their block at the time of reveal
- Any lower bid hurts their chances of winning the auction, given that other eligible proposers may be seeing the same block with the same builder_fee

.

Notably, even if they do manage to win the auction with a lower bid than builder_fee

, the difference does not constitute any kind of financial gain for them.

**Builder strategy**

The strategy for builders remains largely unchanged from today: They still compete with each other, the only meaningful change is that instead of bribing proposers, they bribe the network itself. It is of no difference to them that the bribe is burned rather than given to a proposer.

Knowing the simple strategy employed by eligible proposers, the only thing builders have to worry about is A) having block with a higher builder_fee

than their competition, and B) making their block known by as many eligible proposers as possible.

Exclusive deals between staking pools and private builders are rendered moot. No matter how large a staking pool is, there will always

be on average B

eligible proposers willing to propose their blocks as long as the builder_fee

is higher than other possible blocks.

Similarly, competition between builders means there is no way around the burn: A builder can

become a proposer to try proposing his own blocks, hoping to minimize MEV burn and thus maximize net gains from value extraction. But he still has to win the proposer auction, meaning his bid will still be competing with every other builder's builder_fee

, leaving him with roughly the same marginal gains when most of the MEV is burn even when he does win the auction.

**Proposer/Builder trust**

Of course, due to the untrusted nature of the relationship between builders and unknown eligible proposers, MEV theft is a concern. Consider the presence of a reputable/trusted relay between builders and eligible proposers every time I mention builders broadcasting blocks to proposers.

Solving this aspect of trust is out of the scope of this proposal, but something like Barnabé's PEPC proposal could work nicely here, and we could achieve the goals of fully trustless PBS even without having it explicitly enshrined in the protocol.

## Censorship concerns

This proposal's first and foremost goal is mitigating proposer centralization. MEV burning by itself doesn't attempt to mitigate builder centralization nor its ensuing censorship concerns.

That said, MEV burning does solve the incentive-alignment concerns related to inclusion lists and other builder constraining schemes: An honest proposer will now have no problem adding transactions to the inclusion list, as their income doesn't depend on bribes from builders who could refuse to build blocks while the inclusion list is not empty. The only meaningful end result is that less MEV gets burned for the time it takes for the inclusion list to be emptied by non-censoring blocks builders.

Likewise, builders can no longer "punish proposers retroactively by refusing to build for them in the future", as builders can't know who the eligible proposers are until they reveal themselves.

If the censorship instead happens at the proposer level, it is even more a non-issue: refusing to reveal a block containing censored transactions simply means some other proposer will gladly propose it and reap the proposer rewards. In the same vein, a proposer can still decide to connect exclusively to “regulation-compliant relays”, but if non-compliant blocks have a higher builder_fee

, the compliant proposers will simply lose the auction - their personal preferences regarding regulations have a much lower impact than they do today.

### Sophisticated proposers

Consider the following simplified situation:

- Proposing a block has a reward of 0.05 ETH on the consensus layer
- One of the eligible proposers is a sophisticated validator: He is able to detect and extract his own MEV directly
- Every other eligible proposer is connected to an external builder through MEV boost
- There is an MEV opportunity of 1 ETH in the mempool

MEV-Boost proposers will merely see a block hash from a builder, along with the builder_fee

of 1 ETH. They will thus blindly relay that 1 ETH as their own bid, in the hopes of being the one to get the 0.05 ETH proposer reward.

The sophisticated proposer however, is clearly able to bid 1.01 ETH to win the auction easily: he has full control over his custom MEV-extracting block.

This would effectively mean that the 0.05 ETH reward on the consensus layer could (very loosely speaking) also count as MEV, and we’d have the same concern of proposer centralization, since sophisticated validators have an unfair advantage over simple validators and can easily win every auction.

Thankfully, it is easy to see why this is not the case by comparing the gains made by an MEV-Boost proposer with those of the sophisticated builder who outbids them:

- MEV-Boost proposer: +0 ETH MEV - 0 ETH burned + 0.05 ETH reward = net gain of 0.05 ETH
- (The MEV extracted and burned belonged to the builder, not them)
- (The MEV extracted and burned belonged to the builder, not them)
- Sophisticated proposer: +1 ETH MEV - 1.01 ETH burned + 0.05 ETH reward = net gain of 0.04 ETH

This result is obvious: the sophisticated proposer was indeed able to easily win the auction by bidding an extra 0.01 ETH, but winning the auction this way involved sacrificing gains by burning more ETH than everyone else, which is not a desirable outcome in terms of opportunity costs.

# Conclusion

We have seen how such an auction could plausibly work, resulting in the MEV bribes that currently go to block proposers being burned instead. The idea of such a burn auction is not new, but I believe there is a lot of merit to applying it with a focus on MEV.