Leaderless Auctions

Leaderless Auctions are decentralized auctions with no auctioneer. They address the “last look” problem that can emerge when one participant is allowed to act after all others.

In a leaderless auction, all participants must commit to their bids at the same time. Any violation of this commitment leads to an attributable fault. Bids are threshold encrypted to avoid information leakage.

Auctions can be finalized by submitting the results to a blockchain, where they can be efficiently verified via signature aggregation. For the purposes of this paper, we will use Ethereum.

The protocol, similarly to Byzantine-fault-tolerant consensus protocols, assumes a predetermined participant set of which more than two thirds are honest. It also requires that participants can reliably get messages to one another within a fixed period of time (say, two seconds). If this latter assumption is violated, for example by a network partition, some honest parties may receive a fault. However, assuming such violations are rare, we can design fault penalties so that the impact of this issue is minimal.

There are several issues this design does not address. In particular, participants can still submit their bids later than others to the extent that they have lower latency. Additionally, in this version of the protocol, bids from the public are unencrypted for simplicity. This means a dishonest auction participant has a last look relative to members of the public who trust them.

Motivation

Auctions are everywhere in crypto. Timing games, last looks, and short-term censorship are starting to show up, and this protocol is an attempt to resist at least some of them.

We’re exploring how this could be incorporated into Angstrom’s top-of-block and batch auction, which uses a separate consensus mechanism for MEV protection by executing orders at a common price.

Variants of this auction may be applicable in other settings, such as:

- Enshrining a PBS auction in an L1 like Ethereum, as proposed by Mike Neuder and Justin Drake
- Enforcing fairness of an orderflow auction like UniswapX or MEV-share
- Hardening an onchain batch auction, like the one discussed by dYdX
- Decentralizing the sequencer for an L2 like Optimism

Prior Work

There has been much excellent recent work on crypto auctions. See, for example:

- Censorship Resistance in On-Chain Auctions (2023) by Elijah Fox, Mallesh M. Pai, and Max Resnick
- Credible, Optimal Auctions via Blockchains (2023) by Tarun Chitra, Matheus V. X. Ferreira and Kshitij Kulkarni
- A New Architecture to Mitigate MEV (2023) by dYdX
- Multiplicity (2023) by Duality
- Cicada (2024) by Noemi Glaeser, István András Seres, Eötvös Loránd, Michael Zhu, and Joseph Bonneau

While much of this work addresses the problem of censorship

—the ability of a leader to exclude others' transactions or bids—we are not aware of any work that directly addresses the problem of the last look

—the ability of the proposer to insert or cancel their own bid a significant amount of time later than any other participant.

Problem

Context

Zed wants to auction off one ETH every twelve seconds. He sets up a committee of his friends, Alice, Bob, Charlie, and Dan to run the auction. Each of these participants

will collect bids from the public, and then, together, they will decide on the highest bid any of them has seen. Zed trusts the group as a whole, but thinks at most one of them might be dishonest.

First Attempts

One possibility would be to use a single-block auction in a decentralized consensus protocol like Tendermint. Each participant would collect bids from the public and submit the highest one for inclusion in the block. However, Tendermint features a block proposer who has total control over the contents of the block. This means a dishonest proposer could simply censor everyone else's bids and win the auction with a bid of 0.

To address this issue, we could change the duration of each auction from one block to two. Two different proposers would then propose a block containing the highest bid they had seen, one after the other. The overall winner of the auction would be the high bidder across those two blocks. Because at most one proposer is dishonest, at least one of those blocks would contain an honest bid. This is a big improvement!

The Last Look Problem

However, imagine that the dishonest proposer, say, Bob, is the one who gets to propose the second block.

The first proposer, Alice, submits a bid of $10,000 for the Ether in her first block. Bob now has a few seconds to decide what he wants to do. He checks the price of Ether and sees it has just spiked to $11,000. He places a bid of his own for $10,001 and wins the auction, netting $999 in profit.

As a dishonest participant, Bob is going to do this whenever it's profitable – that is, any time the price of Ether is greater than the current winning bid in Alice's block. That means anyone sending their bids to Alice will only get them filled if the current price of Ether is less than or equal to their bid price – in other words, if the trade is unprofitable for them. Even if Bob wasn't able to see other bidders' bids, he still would have an informational advantage thanks to his ability to submit his bid a few seconds after anyone else.

If they are rational and know what is going on, bidders other than Bob will stop bidding when he is the second proposer. On the margin this will depress bids overall and result in less money for Zed.

This is known as the "last look" problem, and it's a form of MEV. Because it's quite similar to other currently tolerated forms of MEV, even "honest" participants like Alice, Charlie, and Dan might be tempted enough to engage in it as well, putting Zed and any honest bidders remaining at a disadvantage.

We'd like to prevent this from happening.

Solution

Problem Summary

This protocol is addressed specifically at solving the last look problem in decentralized auctions – the situation where one

participant gets to place or cancel their bid meaningfully later than others at no cost to themselves, giving them an unfair advantage.

In particular, we want to ensure that all bids were submitted by a given wall-clock time (say, 12:00 PM) regardless of who submitted them.

Most blockchain consensus protocols feature a leader

or proposer

who suggests a block for other participants to agree on. Because all other participants must send their bids to the leader in time for the leader to include them, the leader gets longer to decide on their own bid, giving them a free last look.

Protocol Overview

A leaderless auction arrives at auction results in three rounds, and creates an easily-validated aggregate signature verifying them in a fourth. Anybody can submit these signed results to Ethereum to finalize the auction.

The protocol also specifies a set of easily provable fault conditions. Anyone can submit proof of a fault at any point, penalizing the participant responsible.

Properties

As we will prove below, this protocol solves the last look problem by satisfying the following two properties:

1. No latecomers:

No participant can construct a bid after the wall clock time marking the end of round 1.

1. No free option:

Any participant who preserves the optionality to cancel after round 1 has to pay for that option ahead of time. Effective penalties should make this economically unviable.

It's worth noting that some participants may have lower latency than others, and will therefore be able to act later than others in round 1.

Assumptions

We assume that out of 3f+1 participants, 2f+1 are "honest" (the same threshold required by Byzantine-fault-tolerant consensus protocols).

Our synchrony assumption is that all honest participants are able to send messages to one another that are guaranteed to arrive within some fixed amount of time, say, 1 second. This assumption doesn't need to hold for overall safety or liveness (ensuring we don't produce conflicting auction results and that auctions eventually happen), because we're relying on Ethereum for those properties. We discuss what happens when this assumption is violated below.

We also assume participants have synchronized clocks, as in Ethereum.

Auction

Round 1 - Sending Bids

Each participant sends one signed bid to each other participant.

These bids are threshold encrypted

, meaning they are encrypted using a public key that can be decrypted by any collection of f+1 or more participants (which will happen in round 3). This means that no participant is able to see the value of any other participant's bid before submitting their own, and therefore cannot exploit knowledge of that bid. There are several ways this encryption could be implemented, but the details are outside the scope of this paper – see for example vetKeys.

Honest participants are required to submit the highest bid they have received from a member of the public. However, because bids from the public are not encrypted in this design, a dishonest participant could opt to outbid the bids they received from the public by a small amount. Members of the public could address this by sending their bids only to one participant they trust. Cryptographic solutions are possible but would add complexity.

Round 2 - Sending Bid Sets

Each participant gossips the signed set of all the encrypted bids (a "bid set") they received in round 1, including their own, to all other participants. All participants now have a "bid view", or set of bid sets received from other participants.

An honest participant will only include another participant's bid in their bid set if that bid arrived before the end of round 1, as determined by the participants' synchronized clocks. So, if a bid is included in an honest participant's bid set, it must have been sent by the end of round 1. Since there are at most f dishonest participants, if a bid appears in f+1 or more bid sets, at least one of them was from an honest participant, and that bid must have been sent by the end of round 1.

Any honest participant could use this information to resolve the auction. But, the protocol has no way of knowing which participants are honest. So, we need another round.

Round 3 - Sending Bid Views and Threshold Decryption Information

Each participant gossips all bid sets they received in round 2 (their "bid view"), including their own, to all other participants.

Together with their bid view, they also send out their share of the decryption information used in the threshold encryption scheme discussed above. Any collection of f+1 of these is sufficient to decrypt all of the encrypted bids.

This is the most expensive step in terms of bandwidth: each participant must send O(n) bid sets of O(n) bids to O(n) participants, for a total of O(n^3) bandwidth in the worst case. However, everyone should have the same actual bids from each other participant (in the absence of a costly equivocation fault, discussed below), and the presence or absence or a bid in a given set is just a single bit. Furthermore, if everyone is honest and the network is functioning properly, all participants should have every bid in their bid set, and every participant should have a bid set from every other participant. So, if participants only send messages indicating which bids or bid sets they are missing

, in ideal conditions they have to send only a single bit ("everything was here") or else just a few ("set 3 was missing bid 13, everything else fine").

At this point, every participant should have the 2f+1 bid views from the honest participants and be able to decrypt them. With f+1 of these, they can prove they have at least one bid view from an honest participant, enough to resolve the auction.

The protocol could in theory end here, with any participant able to submit f+1 bid views to the blockchain for finalization. However, due to the space complexity and cost of verification, it is impractical to submit 2f+1 bid views to Ethereum and verify them (perhaps it would be more feasible on a custom chain).

Accordingly, we need one more round so that participants can certify auction results in a way that can be easily verified on-chain.

Round 4 - Verifying Bid Views

In order to guarantee our desired properties, we want to easily verify that the winning bid of the auction was the winning bid of at least one valid bid view, and that it was greater than or equal to the winning bid of at least one honest participant's view. Round 4 is designed to enable this.

Each participant examines all of their decrypted bid views. A bid is _valid _in a bid view if it is present in at least f+1 of the bid sets that make it up. The winning bid of a view is the highest valid bid in that view.

A participant nominates

the winning bid of a given bid view if it is greater than or equal to the winning bid in that participant's own bid view.

Each participant constructs a threshold signature for each bid they are nominating, and gossips the set of all of these bids and signatures to each other participant. Note this is just O(n) in space complexity (one signature per view at most) and O(n^2) in bandwidth complexity. In fact, if all participants are honest and there are no network difficulties, the same bid will win each bid view, and so each participant will nominate only that single bid.

Settlement

Any bid that has been signed in this way by at least f+1 participants is confirmed

. Note that in some cases, two or more different bids may be confirmed (though this means at least one participant must have faulted in round one, and therefore did not receive a free option).

Any confirmed bid may be submitted to the Ethereum blockchain, where it can be verified by a simple signature verification. The submitter may receive a small payment from the protocol to reimburse them for gas. Only one confirmed bid will be accepted on Ethereum, allowing all participants to come to consensus on the winning bid in case multiple confirmed bids are present.

It is possible that the Ethereum block proposer may censor the auction and not include it in the block. This may even happen several blocks in a row. In this case, the participant who submits the results for the current block's auction must also submit the results for all of the missing blocks as well. We discuss how this changes fault penalties below.

Faults

Basics

Anyone can submit fault proofs asynchronously as soon as they have the evidence to do so, most likely resulting in full or partial slashing of the offending participant's stake. participants reporting fault proofs may receive some part of the slashed stake as a bounty for reporting.

An equivocation fault

occurs when a participant has conflicting bids present in different bid sets or different bid sets present in different bid views. Because this kind of equivocation is either intentional or at the least due to client-side issues as opposed to network problems, penalties for it can be quite severe, including full slashing.

An absence fault

occurs when a participant's bid is absent in some collection of f+1 bid sets. This could either mean that the participant was dishonest and was attempting to back out of a bid they sent in round 1, or, hopefully rarely, that the participant was honest and the network was partitioned.

The penalty for an absence fault shouldbe set to be higher than the option value of being able to cancel a bid. This needs to be balanced against the frequency of genuine network failures, and is ultimately a quantitative problem. In the worst case, having a system of escalating penalties for repeat offenses can limit the frequency with which taking this option is profitable and keep the activity to a minimum. Any system using this protocol could monitor how often these types of faults are occurring relative to verified network failures and adjust penalties accordingly.

Faults for Censored Auctions

As we will discuss further below, auction participants can preserve the optionality to cancel their bids in a later round by committing a fault in round 1.

Any participant who does this can create even more optionality for themselves by bribing the Ethereum block proposer to censor the auction results of this auction from the Ethereum block, potentially several times in a row, before finally being submitted as discussed above under "settlement." To preserve the "no free option" property, we increase the fault penalty for a given auction linearly each time it is censored, so that the original faulter must pay for the option value of each missed block.

If an honest participant unintentionally faults due to network issues, somebody could use these increasing penalties to grief them, but the griefer would have to pay to censor the blocks.

Mitigating Unintentional Faults

If any participant has f+1 or more bids missing from their bid set, we know by assumption that they are missing the bid of at least one honest participant, since at most f participants are dishonest. So, we can ignore this participant's bid set for the purposes of computing absence faults, since either they are dishonest, or there is a network partition occurring. This should reduce the number of faults generated by major network partitions.

Proofs

i. No Latecomers: Any participant must send their bid to at least one honest participant in round 1 in order for it to be valid in any bid view and therefore have a chance of winning

Assume some participant does not send their bid to any honest participants in round 1. Because there are 2f+1 honest participants, this bid will appear in at most (3f+1)-(2f+1)=f bid sets in round 2, which is not enough for it to be valid in any bid view.

If it is not valid in any bid views, it cannot be the winning bid in any bid views, and no honest participant will nominate it. Since f+1 participants must nominate a bid for it to win the auction, at least one of those participants must be honest, and this bid cannot win.

ii. In the absence of network partitions, a participant who sends their bid to all participants in round 1 will not generate a fault

Let's say our honest participant is Alice.

She sends her bid to all 3f+1 participants in round 1. Because 2f+1 of those are honest, every honest participant will send her bid in their bid set to all other honest participants. That means every honest participant will see her bid in at least 2f+1 of the bid sets they receive (including their own).

Because there are at most 3f+1 bid sets, her bid will be absent in at most (3f+1)-(2f+1)=f bid sets, so it will not generate a fault.

iii. In the absence of network partitions, a participant who sends their bid to f+1 or more honest participants in round 1 will win the auction if it is the highest bid submitted to any honest participant in round 1

Assume Alice sends her bid to f+1 honest participants in round 1. Each of these honest participants will then include her bid in their bid set, which they send to every other participant. This means that each of the 2f+1 honest participants sees Alice's bid in f+1 bid sets, and therefore considers her bid to be valid in their bid view.

By assumption, Alice's bid is the highest bid submitted to any honest participant in round 1. So her bid is the winning bid in every honest participant's bid view, and every honest participant will nominate it. Furthermore, by proof (i) above, no higher bid can be valid in any bid view, so no honest participant will nominate it. This means Alice's bid is the only one with at least f+1 nominations, and must win the auction.

Note there is an edge case if two such bidders are tied with the highest bid. We can adjudicate this with randomness from the threshold decryption.

iv. Any participant must send their bid to at least f+1 honest participants in round 1 to avoid generating a fault

Assume the participant sends their bid to k<=f of the honest participants in round 1 (potentially including themselves).

This bid will then appear in k of the bid sets sent by honest participants.

Every honest participant will receive 2f+1 bid sets from the other honest participants (including themselves), and only k of those contain the bid in question, so that 2f+1-k>=2f+1-f=f+1 of those will not contain the bid in question, which is enough to generate a fault.

Note that even if the network is partitioned, if honest participants ensure that receipt of their bid sets is acknowledged by all other participants, and re-send the sets until acknowledgement is received, a fault can be generated once the network is restored.

v. No free option

By (iii), any participant who sends their bid to f+1 or more honest participants in round 1 will win the auction if it is the highest bid sent to any honest participant in round 1, no matter what they do later. By (iv), any participant who sends their bid to fewer than f+1 participants in round 1 will receive a fault (and therefore be penalized for it). Accordingly, anyone wishing to preserve the optionality to cancel will have to pay for it.

Conclusion

Decentralized auctions have only recently begun to see widespread use, introducing many novel problems. This paper is aimed at an issue we have not seen addressed elsewhere. We hope and expect that many better solutions will emerge – ideally ones that are simpler, with fewer rounds, or have fewer assumptions around things like synchrony or number of honest participants.

Still, while this design doesn't solve every issue, it does address some of the common "last look" issues inherent in decentralized auctions. We hope it can be of some use in and of itself, help others working on similar problems, or even be improved or remixed into something better.

Acknowledgements

Frankie, Joachim Neu, Achal Srinivasan, Georgios Konstantopoulos, Ciamac Moallemi, Max Resnick, Lefteris Kokoris-Kogias