Circumventing the Impossibility of Order Policy Enforcement

Introduction

Order manipulation attacks such as frontrunning and sandwiching have become an increasing concern in blockchain applications. Enforcing strict transaction ordering in a block is one of the many proposals for reducing MEV in blockchains. The paper *Breaking the Chains of Rationality: Understanding the Limitations to and Obtaining Order Policy Enforcement* explores the impossibility conditions for Order Policy Enforcement(OPE) under the assumption of rational actors. This article introduces a novel protocol design integrating a DAG-based consensus mechanism and Merkle Root commitments to circumvent this challenge.

The Impossibility of OPEs

The paper suggests that rational actors in a decentralized system can manipulate transaction order to their advantage. This typically manifests as front-running, where an actor places their transaction ahead of a seen pending transaction to profit from market impact.

Implementation of OPE protocols like Themis works under the strong requirements of byzantine actors($f < n/4$ for Themis) which is rarely the case in the real world. The paper argues that an attacking set of rational nodes A can collude and manipulate the transaction order such that they can extract maximal value in each block. The argument is based on the fact that you can't design a protocol that incentivizes actors in A to incriminate colluding actors and subject them to slashing. Using a TEE ensures all proof of collusion is available only to nodes that are part of A.

Proposed Protocol Design

The following protocol incorporates several elements to mitigate the fair ordering challenge:

Narwhal Bullshark Consensus

At the core of the protocol is Narwhal Bullshark, a DAG-based consensus mechanism. DAG-based consensus algorithms allow nodes to continuously create a DAG and reach a consensus about ordering with zero communication overhead. DAGs provide causal ordering, that is, with a DAG you can run a topological sort algorithm to order the vertices in the DAG but does not ensure total ordering. Bullshark utilizes a deterministic algorithm ζ to totally order the vertices in the DAG. ζ depends on the vertices itself to provide total ordering.

Merkle Root Commitment

Each transaction within the vertex of the DAG is ordered using Merkle root commitments and the client submitting the transaction receives the signed merkle root from the validator node. If a node that produces a vertex within the DAG is found to have misrepresented the transaction order, it is subject to slashing penalties by the client(as the client recieves a commitment to the order). This ensures a verifiable transaction ordering in every vertex.

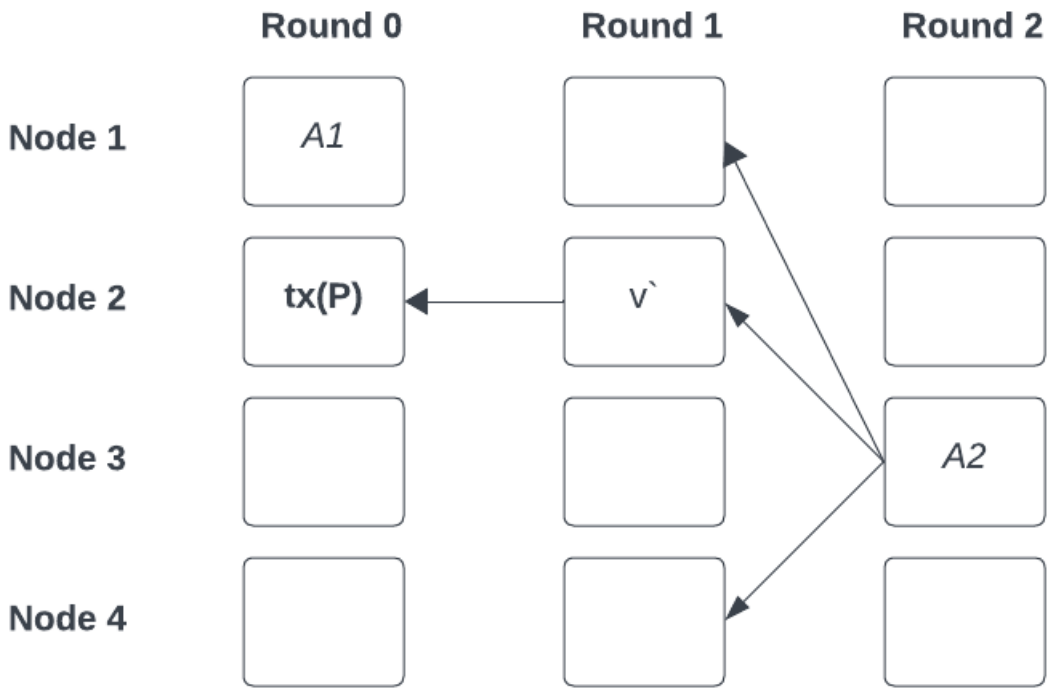
Merkle root commitments alone does not prevent frontrunning/sandwiching as nodes can always re-order the transactions as soon as they see the transaction and provide the client with the Merkle root commitment. It ensures that the node has to front-run it immediately and can't front-run it at a later point by replaying the transaction order.

Incentivization Mechanism

Clients can pay a priority fee for transactions(similar to EIP-1559) and this fee gets split between all vertices ahead of the node which includes the transaction when ordered with a topological sort all the way till the anchor vertex of a block(2 nodes for Bullshark).

Priority fee is paid to all nodes that include the transaction in their vertex(rpc call has to be updated such that it includes addresses of validators that get paid if they include it in their vertex).

In the diagram below, if ***Node 2*** includes a transaction $tx(P)$ with a priority fee of P in ***Round 0*** and votes for itself in ***Round 1*** and the anchor node $A2$ which is ***Node 3*** includes it in its block, the priority fee is split between ***Node 2*** and ***Node 3***.



Analising Merkle Root Commitments + DAG

A key feature to mitigate manipulation risks is allowing clients to send their transactions to multiple nodes. This approach introduces uncertainty regarding which node's vertex will be ordered first with ζ.

Let us consider the case of Alice sending a transaction tx to two nodes $n1$ and $n2$ with a priority fee of p. The transaction received by both the nodes is identical which ensures only one instance of tx can be included when ordered with ζ.

If $n1$ or $n2$ want to front-run/sandwich tx, they have to do it at the time of inclusion of the transaction in their vertices $v1$ and $v2$ respectively(Merkle Root Commitment). It is fair to assume if $n2$ front runs tx with a transaction tx' it incurs a fee of f. The MEV is profitable if the gain g because of front running is greater than f. If both $n1$ and $n2$ include the transaction tx and $v1$ gets ordered before $v2$ with ζ, $n2$ incurs a loss of f.

Collusion is not a stable Nash equilibrium for nodes as they are incentivized by p to not collude.

| ($n1$, $n2$) | Attack | Defer |

| ---------- |:--------- |:-------- |

| **Collude** | (k, k) | (0, 0) |

| **Include** | (p/2, -f) | (p/2, 0) |

Suppose the extractable MEV is m and the attacking set is A where $|A| > 2n/3$, and the shared MEV $k = 3m/2n$.

$n1$ has an incentive to deviate from collusion as long as $p > 2k$.

As long as the client is ready to pay a priority fee $p > 3m/n$, rational actors will not indulge in frontrunning.

When the nodes in the network increase, $p << m$.

DAG vs Leader based consensus

In contrast to the DAG-based approach proposed in our protocol, leader-based consensus algorithms' reliance on a single leader for transaction proposal presents inherent vulnerabilities to front-running and MEV exploitation. In leader-based consensus algorithms, a leader, potentially utilizing a Trusted Execution Environment (TEE) along with an attacking set of nodes A, can observe and manipulate transaction order, enabling them to front-run before providing a Merkle Root Commitment. This risk can't be mitigated through a priority fee mechanism as adding a priority fee will just end up in the leader receiving both the split MEV and priority fee. Unlike leader-based consensus algorithms, the DAG approach, with its distributed transaction inclusion and the introduction of uncertainty in the total order with ζ, offers a significant

advantage. It mitigates the risk of order manipulation by creating competition between the non-leader nodes to win a part of the priority fee.

How do you guarentee negative utility

The total ordering of transactions of the DAG is completely random(for all transactions of a round while the causal order is still maintained). This disincentivizes validators from front running/ sandwiching as thier transactions as only the relative order of transactions of a vertex is maintained and other transactions might be ordered before the front running transaction/ between sandwiching transactions.

The leader has a view of the existing DAG and will include all the good MEV(such as cross DEX arbs) that has not been captured in the previous round in his vertex, and because his block has the highest priority in the causal order, his vertex has a deterministic position. The protocol can enforce leaders to only include transactions signed by the leaders address.

Conclusion

The combination of these elements disrupts the Nash equilibrium for rational actors seeking to exploit transaction ordering. The uncertainty introduced by multi-node submission, coupled with financial incentives for fair ordering and penalties for dishonest behavior, aligns the interests of network participants with the goal of equitable transaction processing. This I believe circumvents the impossibility theorem.

Future Work

I want to work on modeling the game based on epochs and multi-epoch games for nodes like my previous on the *Flipper Game*.