Time Restricted Transaction Execution

Add a new field minStartTime

to the transaction signature that replicates the behavior of a nonce

. When this value is set, it creates a time delay for the transaction in which the block producer cannot include the transaction.

minStartTime

is a uint256

epoch timestamp at which the transaction will become valid

Signature layout:

[nonce, minStartTime, gasPrice, gasLimit, to, value, data]

Example logic for transaction exclusion:

if(msg.minStartTime < block.timestamp)

A block observed that includes a msg.minStartTime

lower than block.timestamp

of the block the transaction is included in should be rejected by implementing nodes.

Based on some early feedback, I am modifying this proposal with rules for mempool inclusion.

Let’s deem a minStartTime

of 0 to be a special case that implies the transaction signer does not want to set a future minStartTime

and can be included in a block immediately.

Using standard inclusion rules for mempool, transactions that have a nonce

lower than the account’s current nonce will be dropped from mempool. Meaning the user can submit multiple transactions with the first included transaction invalidating the other transactions in mempool.

Mempool inclusion should be restricted to a single transaction with a msg.minStartTime > 0

. This will prevent having multiple transactions in mempool at the same time. If two transactions arrive with msg.minStartTime > 0

, the node should select the transaction with the lowest msg.minStartTime

.

Using heuristics within mempool will be key to prevent transaction with msg.minStartTime > 0

where the msg.minStartTime

is set arbitrarily high ex: 10 years. Each transaction should be restricted to block.timestamp + 1 days

.