Time-dependent logic is ubiquitous in decentralized applications. This thread aims to clarify how time constraints can be realized in resource logics.

A simple example would be a coupon resource

, i.e., a resource that you can redeem for something but that becomes invalid on a certain date, e.g., January 1st 2025 00:00. Another example would be a voting application, where voting is possible only within a voting period.

The coupon resource logic could look like this

if (currentTime >= expirationTime) return false; else return true;

where currentTime

and expirationTime

could be unix timestamps.

expirationTime

would simply be a constant, hardcoded value stored in the resource plaintext.

In contrast, currentTime

must be the timestamp of the block which this transaction is part of in the moment it gets executed.

Two sets of questions are coming to mind:

1. Who provides this currentTime

timestamp? The controller of this resource? An oracle?

1. How can it be accessed in the resource logic and ensured to be up-to-date in the moment of execution? Must there be a built-in in the RM or can the timestamp be provided simply though a resource being part of the transaction object?

Providing time-information sounds like a natural responsibility of the controller.

In Ethereum smart contracts, for example, block.number

and block.timestamp

are special variables existing in the global namespace. The latter is calculated from the current block number and the genesis timestamp (since blocks/slots take exactly 12 seconds in PoS Ethereum).

Could this work similarly in the case of the ARM? Could we provide this information as special variables built into the RM or would this be non-linear resources

data provided by the controller?

Moreover, this would require resource logics proofs of time-dependent resources to be computed at execution time (which requires knowledge about resource time dependencies of the block proposer). This sounds undesirable because you effectively have two resource types and this leaks information in the shielded case. Wdyt @vveiln

Furthermore, it is unclear how this can be generalized for resources moving between different controllers. If a resource moves between controllers, block.timestamp

will still be approximiately

the same value, whereas block.number

will certainly be different. Any thoughts on this@isheff?

In contrast to a controller, an oracle is an application providing off-chain data. This application would provide time information in the form of resources being added to the transaction.

There are several problems with this approach.

The oracle application cannot know the time in which the transaction will get executed and would require a mechanism to add the resource right before transaction execution. This sounds impossible since there can be latency issues (e.g., because of packet routing, proof computation times, etc.).

Therefore, time provided by an oracle could just be seen as a lower bound for the actual time, which is insufficient in many cases. Similar questions arise for price-oracles that also must be up-to-date during execution time.

Moreover, trust assumptions can differ from the controller and oracle manipulation attacks could happen. Overall, attempting to provide time through oracle services/applications doesn't sound like a good approache to me.

I would be happy to hear your thoughts on this topic @cwgoes @degregat @isheff @vveiln.