Per discussions at the hacker house (as I recall, at least), with a diagram and some slightly revised names (proposals also welcome).

[

engine-composition-structure

1136×651 84.6 KB

](https://europe1.discourse-cdn.com/standard20/uploads/anoma1/original/1X/a07deae553ee9e29896724d7575278b2bf7009aa.png)

This segmentation proposal would split all of the engines into five groups (called “machines” here):

- The message machine (corresponding to the current P2P engine / P2P layer work)
- The ordering machine (corresponding to the current mempool, consensus, and execution engine work ~ Typhon)
- The strategy machine, including
- Solver engine (makes decisions about how and when to do solving)
- Automation engine (automatically makes decisions about what consensi to run, who to store data or do computation for, intents to periodically send, etc. based on user preferences)
- Interaction engine (handles I/O with the user)
- Solver engine (makes decisions about how and when to do solving)
- Automation engine (automatically makes decisions about what consensi to run, who to store data or do computation for, intents to periodically send, etc. based on user preferences)
- Interaction engine (handles I/O with the user)
- The identity machine, which stores all

private key material in this system, including: * Decryption engine (decrypts messages addressed to this agent)

- Commitment engine (makes commitments ~ signatures from this agent)
- Decryption engine (decrypts messages addressed to this agent)
- Commitment engine (makes commitments ~ signatures from this agent)
- The hardware abstraction machine (HAM), which abstracts the expensive hardware operations:
- Computational search (compute engine)
- Durable storage (storage engine)
- Computational search (compute engine)
- Durable storage (storage engine)

Rough functionality for these engines is as described inthe specs. It needs to be explicated in much more detail, but in case you were wondering about what a particular engine does, you might find the associated page there helpful.

Outstanding questions / notes:

- I’m not sure about the use of the word “machine”. Alternatively, we could call everything here an engine, and some engines are just composed of other engines (which is fine). Thoughts?
- Sometimes the private key material will lie out of control of the Anoma implementation, we’ll need to think about how to abstract this properly (but I don’t think it’s super difficult).
- Did I miss any engines?
- Is everyone OK with using “Typhon” and “Taiga” as the names for the implementations

? (and keeping all of the abstractions as “XX machine” or “XX engine”)

Does this all make sense? Any alternative proposals or concerns?

cc @isheff @nzarin @tg-x @graphomath @vveiln @degregat for feedback (and in case I missed anything from the

discussions)