Another interesting problem we've been thinking about relates to handling hard-forks in Secret Network.

Recall that nodes in our network share secrets in their enclaves This happens when a new nodes registers their enclave/node in the network, and their enclave in turn is provisioned with the network's secret keys.

Secure enclaves have no persisted memory, so instead, SGX has an API (called sealing) for persisting data from the enclave. This essentially encrypts the data inside of the enclave (using an encryption key that is hard-wired to the specific machine's TEE), and persists it on disk. The enclave (and only the enclave) can then 'unseal' the data, which is essentially loading it from disk into the enclave's memory and decrypting the information.

Sealing API provides two types of sealing policies: MRENCLAVE and MRSIGNER:

1. MRENCLAVE: Only allows the same 'enclave' (i.e., the same code binary) to unseal previously sealed data, even from the same machine.
2. MRSIGNER: Allows the entity who signed the enclave's code binary to unseal the data. This could allow Enigma, or whomever distributes the latest network code (specifically, the enclave part) to decrypt all of the network's data.

So clearly, we want to work with option #1

. This does however create a challenge. #1

implies that whenever there's a hard fork (which also means that the code changes), then no node in the network would be able to recover its secret, which would render the entire blockchain state unreadable. Not great…

To overcome this, we expect hard forks would require a 'hand-over' stage, in which a node running the new version joins in, registers in very much the same way, and gets the shared secrets. From that point and on, all other nodes can upgrade as well and join the network. Essentially, this means all nodes need to re-register when a hard-fork happens.

To be clear, it won't matter who is the first node to do so, so there's no risk of centralization. It's just more challenging to coordinate a hard fork, which is already a challenging process even without this added complexity.

We're still trying to come up with better solutions, if any exist.