This post is a follow-up to the previouspost about MRENCLAVE sealing, and describes an improvement in the way we store the current validator set.

Intro

A Secret node logically consists of 2 parts: the code that runs in the SGX enclave, and the so-called “untrusted” code.

The code within the SGX is responsible for things that should be kept secret, in particular it’s responsible for executing smart contracts and encryption/decryption of the contract state variables.

To prevent replay and fuzzing attacks, there are restrictions on when and who is authorized to call various contract methods.

Before a contract can be executed, the Enclave ensures that the specific transaction is part of a block that was signed by the majority of validators.

Keeping Current Validator Set

For this reason the enclave keeps the current validator set, and follows this logic during block execution:

1. A Signed block is exposed to the enclave. This includes the following:
2. Block header. Among other things it contains a hash of the next validator set.
3. Block contents (list of transactions)
4. Block signatures

The enclave verifies that the block was signed correctly according to the current validator set. Validator signatures are verified, total voting power is checked, and the minimum of validators from the whitelist is satisfied.It saves the block contents, and the hash of the next validator set.

1. The block is executed. Transactions not related to contracts or new node registration are executed entirely in the untrusted code (enclave doesn’t get involved).
2. Eventually, when there is a contract transaction, and the enclave gets called to execute it. This is where the enclave checks that this specific transaction was a part of the submitted block.
3. After the block was executed - the new (updated) validator set is submitted to the enclave.

In the current implementation, the updated validator set is not verified, and the safeguard against potential attack is the demand for the presence of a minimum number of validators from the allow list.

In the context of switching to MRENCLAVE sealing, we decided to improve the verification of the updated validators set. The above scheme is modified this way:

1. When the signed block is exposed to the enclave, in addition to what was done earlier, the enclave returns an evidence to the next validator set.

It’s calculated as follows:

evidence

= Hash(network_seed | blockchain_height | next_validatorset_hash)

This evidence

is returned to the untrusted code, and is stored in the blockchain state storage.

1. After the block is executed, the validator set is submitted to the enclave together with the evidence. The enclave calculates the evidence to the submitted validator set, and ensures it matches what it should be.

This scheme works both for sequential (normal) block processing, as well as for statesync, since the most recent validator set evidence is downloaded automatically with the whole blockchain state.

Rollout plan

There’re only 2 cases where the above won’t work:

1. For initial validator set, when the network is bootstrapped.
2. The very first block after the version upgrade from the current version (that doesn’t store the validator set evidence).

To overcome this, this is the rollout plan:

In the first release of this feature (likely in upgrade 1.16), the enclave would allow submitting the very first validator set without

the evidence. This covers the following cases:

1. Very first block after the upgrade from the previous version.
2. Bootstrapping a new network
3. Executing the very first block of the new (bootstrapped) network.

Then, during further upgrades, the option to set the very first validator set without evidence will be removed at all for production build.

Code

The code was implemented in mrenclave3 branch. The most relevant changes were made in the following 3 commits:

- ab500f33ff24a33478f9771d2783bdd4cc2d0dc7
- 3a9d44972557a515b3ccbfcdd7613d91ceecdc27
- F07f10a33c1f5e21b5a4451eb3681c9c177af0ad

As usual, we welcome questions and comments from the community.