Hi all

I am glad to announce that we have completed the development of MRECLAVE migration code, and will be continuing with testing and rollout as we move forward

Background re-cap

MRENCLAVE

The MRENCLAVE (Measurement of Enclave) in SGX is calculated using a SHA-256 hash of the enclave unsigned binary. The enclave unsigned binary is the compiled code of the enclave, which is the trusted part of an SGX application. Essentially, MRENCLAVE identifies the software that is running inside the enclave.

MRSIGNER

The MRSIGNER (Measurement of the Signing Key) in SGX is calculated using a SHA-256 hash of the enclave's developer public key.

To calculate MRSIGNER, the enclave's developer public key is first converted to a byte string. The byte string is then hashed using SHA-256. The hash is the final MRSIGNER.

MRSIGNER identifies the developer who developed and released the enclave

Secret Network

In 2020, SCRT Labs decided to use MRSIGNER to seal the consensus seed. This means that for every version of Secret Network, SCRT Labs has signed the enclave with its private key.

The main reason for choosing MRSIGNER is that it made upgrading the chain easy. If SCRT Labs had opted to use MRENCLAVE, we would have also needed to implement a handover mechanism to send the consensus seed from the old enclave to the new enclave every time the chain is upgraded. The development is quite involved, and bugs in the code introduce significant risks for the network upgrade process,

In the past several months, we have been working on implementing the MRENCLAVE migration code, and the main highlights are outlined below.

Current Implementation

The implementation follows the framework proposed by Assaf here, with some notable changes and improvements:

1. Simpler upgrade authorization verification.

In the original proposal the idea was to verify the system state in the enclave using Merkle proofs. It's a feasible option, but pretty cumbersome to implement. The difficulty comes from the fact that the system state, and specifically the accepted upgrade proposal state, is stored in a non-trivial way in the storage, and is subject to changes with newer cosmos SDK versions.

We overcome this by introducing an additional message for upgrade confirmation that should be sent to the network (in the form of a transaction) after the upgrade proposal is accepted.

Upon interpretation of this message in a block signed by the validators the enclave will update its internal variable (next_mrenclave).

Nodes will filter-out this transaction if it's malicious (i.e. no such an upgrade was actually accepted) in a custom Ante-handler developed for this kind of transactions

Hence, if the block containing this message was indeed signed by the majority of the validators - this would mean that the new MRENCLAVE was approved by the majority of the voting power

1. Mitigation of "mixing" attacks.

An attacker can run an additional network (e.g. a bootstrap network) using the official Secret version, where the attacker has the majority of the voting power, a.k.a. scamnet, and accept malicious version upgrades. It's important to make sure that this kind of attack is limited to this local network, and that the attacker won't be able to copy sealed files from scamnet to mainnet and vice versa, and by such manipulations export sensitive data to a malicious enclave.

For this we took the following steps:

- All the data is kept in a single sealed file (instead of several individual files, which can be interchanged).

- During the migration process, neither old nor new enclave exposes its sealing key (that'd be equal for both scamnet and production networks).
- The sealed data migration procedure is implemented in terms of files, rather than communication via sockets.
- Instead of "Secret labs final vote" we simply continue to rely on the current MRSIGNER. That means that in addition to the network consensus, only the builds signed by the Secret Labs would be eligible for upgrade.
- The emergency migration procedure, mentioned in the original proposal, is implemented in terms of "offline" migration (as opposed to the "online", where the upgrade proposal is accepted on-chain in a usual manner).

To perform the offline migration, validators should sign the agreed MRENCLAVE value with their account secret key, then all those signatures are aggregated into a single json file.

The accept/reject criteria is identical to the one we use for block validation: at least 66% of the total voting power, plus at least some number of whitelisted validators.

In other words, we stick to the same trust model as with online upgrade proposals.

The code can be found in a separate branch here.

Rollout Plan

The rollout of the migration will be done in stages in 2025.

Stage 1: Upgrade the network to a version that supports migration to MRENCLAVE (sealing will be moved to MRENCLAVE immediately)

Stage 2: Perform a network upgrade using MRECLAVE. All the sealed data will be migrated from the 'old' MRENCLAVE to the new one.

Stage 3: Perform network seed rotation, completing the process

Risks

In case of a software defect in a new enclave code, the network may fail to start post-upgrade. With MRENCLAVE sealing, the faulty code cannot be replaced with a fixed version, because the new version would have a different MRENCLAVE value and thus won't be able to unseal the network key and other key parameters.

To mitigate this risk, we introduce the offline emergency upgrade procedure (explained above) and allows the old enclave to export the key to the fixed enclave. Notably, testing the emergency upgrade procedure can be performed on mainnet without disrupting the normal network operation (it would require participation from multiple validators though).

Call for Comments

We are calling on the community to comment, review and criticize the architecture and the code.

We will also be performing further testing before the code is rolled out to Testnet and eventually Mainnet