

# The Initialization Of Secret Network

## Bootstrap Process

Secret Network is set up to perform computation while having encrypted state, input and output. The state is encrypted using private keys generated during the network bootstrapping from `aconsensus_seed`. The bootstrap node was the first full node on the network and generated the shared secrets similar to a universal trusted zero knowledge setup.

The bootstrap node first does a remote attestation to prove they are running a genuine SGX enclave with updated software. After registering the bootstrap node handled the initialization of following variables:

- `Consensus_seed`
- a true random 256 bit seed used as entropy for generating shareable keypairs between the nodes of the network.
- `consensus_seed_exchange_pubkey`
- `-consensus_seed_exchange_pubkey`
- an HDKF private key and a matching curve25519 public key for encryption of the random seed and sharing this with other full nodes in the network.
- `consensus_io_exchange_pubkey`
- `-consensus_io_exchange_pubkey`
- an HDKF private key and a matching curve25519 public key for encrypting transactions IO in the network. Also referenced as the "Secret Network keypair".
- `consensus_state_ikm`
- An input keyring material (IKM) for HKDF-SHA256 is used to derive encryption keys for contract state.
- `consensus_callback_secret`
- A curve25519 private key is used to create callback signatures when contracts call other contracts
- 

## Bootstrap process epilogue

### 1. Remote attestation

The bootstrap node proves to have a genuine enclave after which it can participate in the network. More information can be found on [this page](#).

### 1. generateconsensus\_seed

The bootstrap node is instructed when started to generate a true random 256 bit seed inside the enclave, the `consensus_seed`.

The `consensus_seed` is sealed with MRSIGNER to a local file `:HOME/.sgx_secrets/consensus_seed.sealed`

...

```
Copy // 256 bits consensus_seed=true_random({ bytes:32});
```

```
seal({ key:"MRSIGNER", data:consensus_seed, to_file:"HOME/.sgx_secrets/consensus_seed.sealed", });
```

...

### 1. Generate seed exchange keypair

For the network to start decentralizing the bootstrap node needs to be able to share the `consensus_seed`. The network can then use the seed to create shared secrets while in the enclave. To securely share the seed the Network uses a DH-key exchange.

Using HKDF-SHA256, `hkdf_salt` and `consensus_seed` a private key is derived. From `consensus_seed_exchange_privkey` calculate `consensus_seed_exchange_pubkey`

...

```
Copy consensus_seed_exchange_privkey=hkdf({ salt:hkdf_salt, ikm:consensus_seed.append(uint8(1)), }); // 256 bits
```

```
consensus_seed_exchange_pubkey=calculate_curve25519_pubkey( consensus_seed_exchange_privkey );
```

...

### 1. Generate IO encryption keypair

Using HKDF-SHA256, `hkdf_salt` and `consensus_seed` a private key is derived. - From `consensus_io_exchange_privkey` calculate `consensus_io_exchange_pubkey`

...

Copy consensus\_io\_exchange\_privkey = hkdf({ salt: hkdf\_salt, ikm: consensus\_seed.append(uint8(2)), }); // 256 bits

consensus\_io\_exchange\_pubkey = calculate\_curve25519\_pubkey( consensus\_io\_exchange\_privkey );

...

#### 1. Generate consensus\_state\_ikm

Using HKDF-SHA256, hkdf\_salt and consensus\_seed derive consensus\_state\_ikm

...

Copy consensus\_state\_ikm = hkdf({ salt: hkdf\_salt, ikm: consensus\_seed.append(uint8(3)), }); // 256 bits

...

#### 5. consensus\_callback\_secret

Using HKDF-SHA256, hkdf\_salt and consensus\_seed derive consensus\_callback\_secret

...

Copy consensus\_state\_ikm = hkdf({ salt: hkdf\_salt, ikm: consensus\_seed.append(uint8(4)), }); // 256 bits

...

#### 1. add info to Genesis state

Publish to genesis.json :

- The remote attestation proof that the Enclave is genuine
- consensus\_seed\_exchange\_pubkey
- consensus\_io\_exchange\_pubkey
- 

Last updated 1 year ago On this page \* [Bootstrap Process](#) \* [Bootstrap process epilogue](#)

Was this helpful? [Edit on GitHub](#) [Export as PDF](#)