Full Node Boostrap

Full Node Process Epilogue

When a full node resumes network participation, it readsconsensus_seed fromHOME/.sgx_secrets/consensus_seed.sealed , does the same key derivation steps as above in steps 2-5.

1. Remote Attestation

The new full node verifies the remote attestation proof of the bootstrap node fromgenesis.json and creates a remote attestation proof for their own machine to show to the network that the node's Enclave is genuine.

1. Generate Registration Keypair

Using HKDF-SHA256,hkdf_salt andnonce (a 256 bit true random) a private key is derived. Fromregistration_privkey calculateregistration_pubkey

```

Copy registration_privkey=hkdf({ salt:hkdf_salt, ikm:Nonce-256bit, });// 256 bits

registration_privkey=calculate_curve25519_pubkey( registration_pubkey );

```

1. Authorize Full Node

The node needs to send ansecretcli tx register auth transaction with the following inputs:

- The remote attestation proof the node's Enclave is genuine
- registration_pubkey
- 256 bits true randomnonce
-

1. Network Receives Auth

On the consensus layer, inside the enclave of every full node the auth transaction comes in.

The Network validator nodes:

- Receive thesecretcli tx register auth
- transaction
- Verify the remote attestation proof that the new node's Enclave is genuine
-

1. Generateseed_exchange_key

seed_exchange_key : AnAES-128-SIV encryption key is used to sendconsensus_seed to the new node

This key is derived in several steps:

- firstseed_exchange_ikm
- is derived usingECDH
- (x25519
-) withconsensus_seed_exchange_privkey
- (which exists in the enclave) andregistration_pubkey
- (which is a public identifier of the full node)
-

This IKM is never publicly available and protects the Secret network private entropy

```

Copy seed_exchange_ikm = ecdh({ privkey: consensus_seed_exchange_privkey, pubkey: registration_pubkey, }); // 256 bits

```

In the second stepseed_exchange_key is derived using HKDF-SHA256 fromseed_exchange_ikm andnonce . When sending theseed_exchange_key to new nodes the Nonce is added as plaintext, it just serves the function of making eachseed_exchange_key unique.

```

Copy seed_exchange_key = hkdf({ salt: hkdf_salt, ikm: concat(seed_exchange_ikm, nonce), }); // 256 bits

```

1. Sharingconsensus_seed with the new node

Theseed_exchange_key generated in step 5 is used to encrypt theconsensus_seed . TheAD for this encryption algorithm is the public key of the new node:new_node_public_key All this logic is done in side the Authorization transaction.secretcli tx register auth outputs theencrypted_consensus_seed

```

Copy encrypted_consensus_seed=aes_128_siv_encrypt({ key:seed_exchange_key, data:consensus_seed, ad:new_node_public_key, });

returnencrypted_consensus_seed;

```

The encrypted output is received by the new full node. The new node now has access to theencrypted_consensus_seed and will have to decrypt to plaintext to receive theconsensus_seed

1. New full node generates ownseed_exchange_key

The AES-128-SIV encryption keyseed_exchange_key is used to decryptconsensus_seed To derive this the reverse logic is followed highlighted in step 5.

First the sameseed_exchange_ikm is derived usingECDH (x25519) withconsensus_seed_exchange_pubkey (public ingenesis.json) andregistration_privkey (available only inside the new node's Enclave) This is the DH-key echange in action as this is the reverse public/private input of the IKM generation in step 5.

```

Copy seed_exchange_ikm = ecdh({ privkey: registration_privkey, pubkey: consensus_seed_exchange_pubkey, // from genesis.json }); // 256 bits

```

seed_exchange_key is derived using HKDF-SHA256 withseed_exchange_ikm andnonce

```

Copy seed_exchange_key = hkdf({ salt: hkdf_salt, ikm: concat(seed_exchange_ikm, nonce), }); // 256 bits

```

1. Decryptingencrypted_consensus_seed

encrypted_consensus_seed is encrypted with AES-128-SIV,seed_exchange_key as the encryption key and the public key of the registering node as thead as the decryption additional data The new node now has all of these parameters inside its Enclave, so it's able to decryptconsensus_seed fromencrypted_consensus_seed and then sealconsensus_seed to disk atHOME/.sgx_secrets/consensus_seed.sealed

```

Copy consensus_seed=aes_128_siv_decrypt({ key:seed_exchange_key, data:encrypted_consensus_seed, ad:new_node_public_key, });

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

```