Distributed Key Generation

Overview

Adistributed validator key is a group of BLS private keys that together operate as a threshold key for participating in proof-of-stake consensus.

To make a distributed validator with no fault-tolerance (i.e. all nodes need to be online to sign every message), due to the BLS signature scheme used by Proof of Stake Ethereum, each key share could be chosen by operators independently. However, to create a distributed validator that can stay online despite a subset of its nodes going offline, the key shares need to be generated together (4 randomly chosen points on a graph don't all necessarily sit on the same order three curve). To do this in a secure manner with no one party being trusted to distribute the keys requires what is known as adistributed key generation ceremony .

The charon client has the responsibility of securely completing a distributed key generation ceremony with its counterparty nodes. The ceremony configuration is outlined in acluster definition .

Actors Involved

A distributed key generation ceremony involvesOperators and theirCharon clients .

- AnOperator
- is identified by their Ethereum address. They will sign a message with this address to authorize their charon client to take part in the DKG ceremony.
- ACharon client
- is also identified by a public/private key pair, in this instance, the public key is represented as anEthereum Node Record
- (ENR). This is a standard identity format for both EL and CL clients. These ENRs are used by each charon node to identify its cluster peers over the internet, and to communicate with one another in anend to end encrypted manner
- . These keys need to be created (and backed up) by each operator before they can participate in a cluster creation.

Cluster Definition Creation

This cluster definition specifies the intended cluster configuration before keys have been created in a distributed key generation ceremony. Thecluster-definition.json file can be created with the help of theDistributed Validator Launchpad or via theCLI .

Carrying out the DKG ceremony

Once all participants have signed the cluster definition, they can load thecluster-definition file into their charon client, and the client will attempt to complete the DKG.

Charon will read the ENRs in the definition, confirm that its ENR is present, and then will reach out to relays that are deployed to find the other ENRs on the network. (Fresh ENRs just have a public key and an IP address of 0.0.0.0 until they are loaded into a live charon client, which will update the IP address and increment the ENRs nonce and resign with the clients private key. If an ENR with a higher nonce is seen by a charon client, they will update the IP address of that ENR in their address book.)

Once all clients in the cluster can establish a connection with one another and they each complete a handshake (confirm everyone has a matchingcluster_definition_hash), the ceremony begins.

No user input is required, charon does the work and outputs the following files to each machine and then exits.

Backing up the ceremony artifacts

At the end of a DKG ceremony, each operator will have a number of files outputted by their charon client based on how many distributed validators the group chose to generate together.

These files are:

- Validator keystore(s):
- These files will be loaded into the operator's validator client and each file represents one share of a Distributed Validator.
- A distributed validator cluster lock file:
- Thiscluster-lock.json
- file contains the configuration a distributed validator client like charon needs to join a cluster capable of operating a

number of distributed validators.
- Validator deposit data:
- This file is used to activate one or more distributed validators on the Ethereum network.

Once the ceremony is complete, all participants should take a backup of the created files. In future versions of charon, if a participant loses access to these key shares, it will be possible to use a key re-sharing protocol to swap the participants old keys out of a distributed validator in favor of new keys, allowing the rest of a cluster to recover from a set of lost key shares. However for now, without a backup, the safest thing to do would be to exit the validator.

DKG Verification

For many use cases of distributed validators, the funder/depositor of the validator may not be the same person as the key creators/node operators, as (outside of the base protocol) stake delegation is a common phenomenon. This handover of information introduces a point of trust. How does someone verify that a proposed validatordeposit data corresponds to a real, fair, DKG with participants the depositor expects?

There are a number of aspects to this trust surface that can be mitigated with a "Don't trust, verify" model. Verification for the time being is easier off chain, until things like aBLS precompile are brought into the EVM, along with cheap ZKP verification on chain. Some of the questions that can be asked of Distributed Validator Key Generation Ceremonies include:

- Do the public key shares combine together to form the group public key?* This can be checked on chain as it does not require a pairing operation
-
    - This can give confidence that a BLS pubkey represents a Distributed Validator, but does not say anything about the custody of the keys. (e.g. Was the ceremony sybil attacked, did they collude to reconstitute the group private key etc.)
- Do the created BLS public keys attest to theircluster_definition_hash
- ?* This is to create a backwards link between newly created BLS public keys and the operator's eth1 addresses that took part in their creation.
-
    - If a proposed distributed validator BLS group public key can produce a signature of thecluster_definition_hash
-
    - , it can be inferred that at least a threshold of the operators signed this data.
-
    - As thecluster_definition_hash
-
    - is the same for all distributed validators created in the ceremony, the signatures can be aggregated into a group signature that verifies all created group keys at once. This makes it cheaper to verify a number of validators at once on chain.
- Is there either a VSS or PVSS proof of a fair DKG ceremony?* VSS (Verifiable Secret Sharing) means only operators can verify fairness, as the proof requires knowledge of one of the secrets.
-
    - PVSS (Publicly Verifiable Secret Sharing) means anyone can verify fairness, as the proof is usually a Zero Knowledge Proof.
-
    - A PVSS of a fair DKG would make it more difficult for operators to collude and undermine the security of the Distributed Validator.
-
    - Zero Knowledge Proof verification on chain is currently expensive, but is becoming achievable through the hard work and research of the many ZK based teams in the industry.

Appendix

Sample Configuration and Lock Files

Refer to the detailshere . Edit this page Previous Introduction Next Cluster configuration