Overview

Encryption Protocols / Key Management

Secret Network uses both symmetric and asymmetric encryption protocols. Specifically, asymmetric cryptography is used for achieving consensus and sharing secrets between nodes and users, whereas symmetric cryptography is used for input/output encryption with users of Secret Contracts, as well as internal contract state encryption.

Secret Network Protocol uses the Elliptic-curve Diffie-Hellman (ECDH) key exchange mechanism between users and validators. This process involves the user, the Secret Blockchain, as well as the trusted component of the Secret Protocol. It is initiated any time a transaction is sent from the user to the Secret Contract.

The encryption and key derivation logic Secret Network uses are as follows:

- HKDF-SHA256
- function for deterministic key derivation;
- ECDH
- (x25519) function for generating public / private key pairs; and
- AES-128-SIV
- symmetric encryption scheme.
-

Symmetric And Asymmetric Encryption

A combination of the above symmetric and asymmetric encryption methods is used to create a safe process for the bootstrapping of the decentralized network, the addition of new SGX nodes, and the encryption of input, output, and state. The management of all the private and public keys may only be shared under specific conditions, and others never leave the SGX instance to ensure private data handling.

Consensus Seed

The Secret Network uses a random number called the “consensus seed” as a parameter to derive a public and private key set used by the Network for encryption purposes. The consensus seed is unknown to any party in the ecosystem and was created at the genesis of the network by the network itself. For deterministic key generation, the network uses a known “salt” which is chosen to be the hash of the Bitcoin halving block.

Transaction encryption

As a transaction is initiated by a user of the network they derive Input Key Material (IKM) using their own private key and the network-generated public key, the network derives the same IKM using the user's public key and the network-generated private key. An encryption key for the transaction is then derived inside the TEE from the IKM, the salt, and a random number generated with the consensus seed. The encryption key is used to encrypt the input specific to this transaction. Only the protocol and the user signing the transaction have access to the encryption key, and therefore access to the input, output, and state related to this specific smart contract computation.