Private transactions - A quick summary

How to reach consensus for a Secret transaction - The flow of a user's data

The following process is based on how Tendermint works to reach consensus in a standard app chain but with specificity related to the privacy features of Secret Network. Please note that Secret Network is permissionless, validators and nodes can join the network at any time.

1. Developers write and deploy Secret Contracts which are executed by the validators, the binary for these contracts is public.
2. Users submit transactions to the mempool — which can include encrypted data inputs
3. Validators receive encrypted data from users and execute the Secret Contract
4. During Secret Contract execution:
5.
   - Encrypted inputs are decrypted inside a Trusted Execution Environment
6.
   - Requested functions are executed inside a Trusted Execution Environment
7.
   - Read/write state from Tendermint can be performed (state is always encrypted when at rest, and is only decrypted within the Trusted Execution Environment)
8.
   - Outputs are encrypted
9. *
10. The block-proposing validator proposes a block containing the encrypted outputs and updated encrypted state
11. At least 2/3 of participating validators achieve consensus on the encrypted output and state following Tendermint BFT
12. The encrypted output and state is committed on-chain to the specific contract
13.

A private transaction on secret - detailed step by step

1. Using the Eliptic-Curve Diffie Hellman key exchange (ECDH) the user generates a shared secret from their private key and the Secret Network public key. The network can generate the same shared secret using the user public key and the Network private key (only available within the Trusted Execution Environment (TEE)).
2. The user then generates a sharedtx_encryption_key
3. using HKDF-SHA256 and the shared secret generated in step 1. The pseudo-random HDKF is used to ensure deterministic consensus across all nodes. The random component comes from a 256-bit nonce so that each transaction has its own encryption key, an AES-256-GCM encryption key is never used twice.
4. After initiating a transaction the user encrypts the input data with the shared transaction encryption key, using an AES-256-GCM authenticated encryption scheme.
5. The user then sends a single transaction with encrypted input data, gas fee in SCRT, and optionally a SCRT deposit (as required by the contract) to the contract address. The transaction adds a message with the user public key and the used Nonce so that the network can generate the sametx_encryption_key.
6. Validators receive the transaction in the shared mempool, after the Tendermintcheck_tx
7. is deemed successful, and take up the data to process the transaction.
8. Each validator runs on a machine with an Intel SGX compatible CPU. The enclave within this CPU is the trusted component, whereas the process interacting with the enclave locally from the outside is the untrusted component. The encrypted transaction input is passed from the untrusted to the trusted component.
9. The enclave uses ECDH to derive the same shared secret from the users public key and the networks private key. The network then derives thetx_encryption_key
10. from the public signed nonce and this shared secret using HDKF. Within the trusted component the transaction input is decrypted to plaintext.
11. The requested contract computation is executed on the plaintext input by the WASMI runtime, which is instantiated within the trusted enclave.
12. There are several possible outcomes from each computation, all of which occur on-chain simultaneously:
13.
    - The contract state is updated.
14.
    - The transaction output is encrypted for the transaction sender and committed on-chain
15.
    - If present, callbacks to other contracts.
16.
    - If present, send messages to other modules.
17. *
18. The output of the computation is encrypted using thetx_encryption_key
19. and an AES-256-GCM authenticated encryption scheme.
20. The encrypted output is then returned to the untrusted component of the validator node.
21. The validator that proposes the current block broadcasts the encrypted output.
22. Other validators compare their result to that of the block proposer. If more than two-thirds of the current voting power

agrees on the result, the proposed block (and all transactions within it) are included in the Secret Network.

23.

Want to learn about Secret transactions in a video format instead? Check out the last part of the Secret Network techstack video series.

Last updated11 days ago On this page * How to reach consensus for a Secret transaction - The flow of a user's data* A private transaction on secret - detailed step by step

Was this helpful? Edit on GitHub Export as PDF