Detailed Architecture

Here you'll find a more in-depth overview with a detailed description of what the two core primitives (Gateways & Relayers) do. First, let us give you the core idea on what SecretPath really hinges on.

The core idea of SecretPath is the following:

The "master private gateway" is a confidential smart contract living on Secret Network.

Since this contract is confidential by default , we canstore a (secret) private key, which only this smart contract can use tosign messages/packets and is unknown to the operators or to any contract administrators.

The only trust assumption that we have for this bridge model to work is therefore that the private key does not leak out the TEE (trusted execution environment)/confidential computation environment. In case the TEE breaks and that private signing keys could potentially leak, the encryption and verification keys can be rotated to new ones, keeping the bridge intact. Since we only rely on a smart contract to handle the core bridging functionality, we call thistrustless . Some may argue that since we are using TEE's, we are not completely trustless but insteadtrust-minimized .

Everything else around this "master private gateway" isuntrusted . We only use a lot of consistency checks and hashing to ensure that data weren't manipulated along any transmission path.

As outlined in the overview, the primary functions of the Gateway are:

1. Handle signature operations

For the "master public gateway" that is on a public blockchain, that means only verification of signatures. In our case, this public gateways verifies all secp256k1 signatures signed by the user or by the master private gateway.

The private master gateway does verification of the secp256k1 signed and encrypted user payloads. It also signs all outgoing messages using secp256k1 so that other user made private contracts as well as the public gateway contract can verify the validity of the message/packet.

1. Serialize/deserialize operations

Both gateways also handle serialize/deserialize operations. They convert contract inputs into correctly formatted packets and vice versa, while also doing the appropiate conversions between base64 (mainly used for secret) and hex (mainly used for EVMs) inputs.

1. Re-encrypt inputs

Third, gateways on privacy-preserving chains can re-encrypt their inputs under the key of the final compute contract. The master private gateway can handle encrypted or unencrypted payloads. In the case of encrypted payloads, the payload is encrypted using a shared key that is generated via ECDH from a users randomly generated private key and the gateways encryption public key. We use the well known Chacha20-Poly1305 to do the symmetric encryption using that shared key.

Last updated1 month ago On this page *1. Handle signature operations * 2. Serialize/deserialize operations * 3. Re-encrypt inputs

Was this helpful? Edit on GitHub Export as PDF