ZKL Architecture

This file describes the cryptographic activities that take place in zk-Lokomotive's platform and web applications.

Yigid BALABAN, fyb@fybx.dev

Revision 3
20/09/2024

2. Cryptographic Activities

System Elements

The system elements are/will be described and discussed in the ZKL System Architecture document. This section is to provide a reminder/reference.

1. Key Derivation Service

The Key Derivation Service (or KDS for short) provides

1. a deterministic secp256k1 keypair generator from BIP-39 mnemonics,
2. a pseudo-random BIP-39 mnemonic generator through web-bip-39 package.

2. Cross-chain Identity Registry

The Cross-chain Identity Registry (or CCIR for short) provides a method to look up identities and their corresponding public keys.

3. Encrypted File Storage

The Encrypted File Storage is a distributed storage solution that allows the recipient to retrieve payloads that were uploaded for them.

4. Client

The client (the sender) generates the encrypted payload to be sent to the reciever, whose public key is retrieved via the CCIR. The workflow of sending an encrypted file for a recipient is described in the next section.

Workflows

Definitions

Q_r : Recipient's public key on curve secp256k1
G : The generator point on curve secp256k1
Z : A symmetric key derived for the file to be sent, the shared secret
F_c : The file contents, in plaintext
F_m : File's metadata, name, etc
F : The intermediate file format, ready to be encrypted
F_c : The file contents, in ciphertext
IV : The initialization vector required for AES-GCM-256
P : The payload, what is sent to the recipient

The F, intermediate file format is as follows:

Bytes	**[0]..4**	**[4]..1024**	**[1024]...1024+len(F_c)**
Content	Length of F_m	F_m	F_c
Length in bytes	4 bytes	1020 (255 * 4 bytes)	variable

Workflow: Sending a file

1. The Q_r is retrieved from the CCIR.
2. An ephemeral keypair is generated, Q_e and d_e.
3. The shared secret which will be used in symmetric encryption is computed from $Z = d_e \times Q_r$.
4. File intermediate F is encrypted using AES-GCM-256 with encryption key Z, and a randomly generated initialization vector, IV.
5. The payload P is created by concatenating $Q_e,\ IV,\ \mathrm{MAC},\ F_e$.
6. The payload is uploaded to the EFS.

Reminder

The MAC in the payload P at step 7 is a result of using AES-GCM-256, which is selected in ECIES implementation in the Rust crate we consume through ecies/rs-wasm package.

Workflow: Receiving a file

TBE