ZKL Architecture

This file describes modules and submodules present in zk-Lokomotive's platform and web applications.

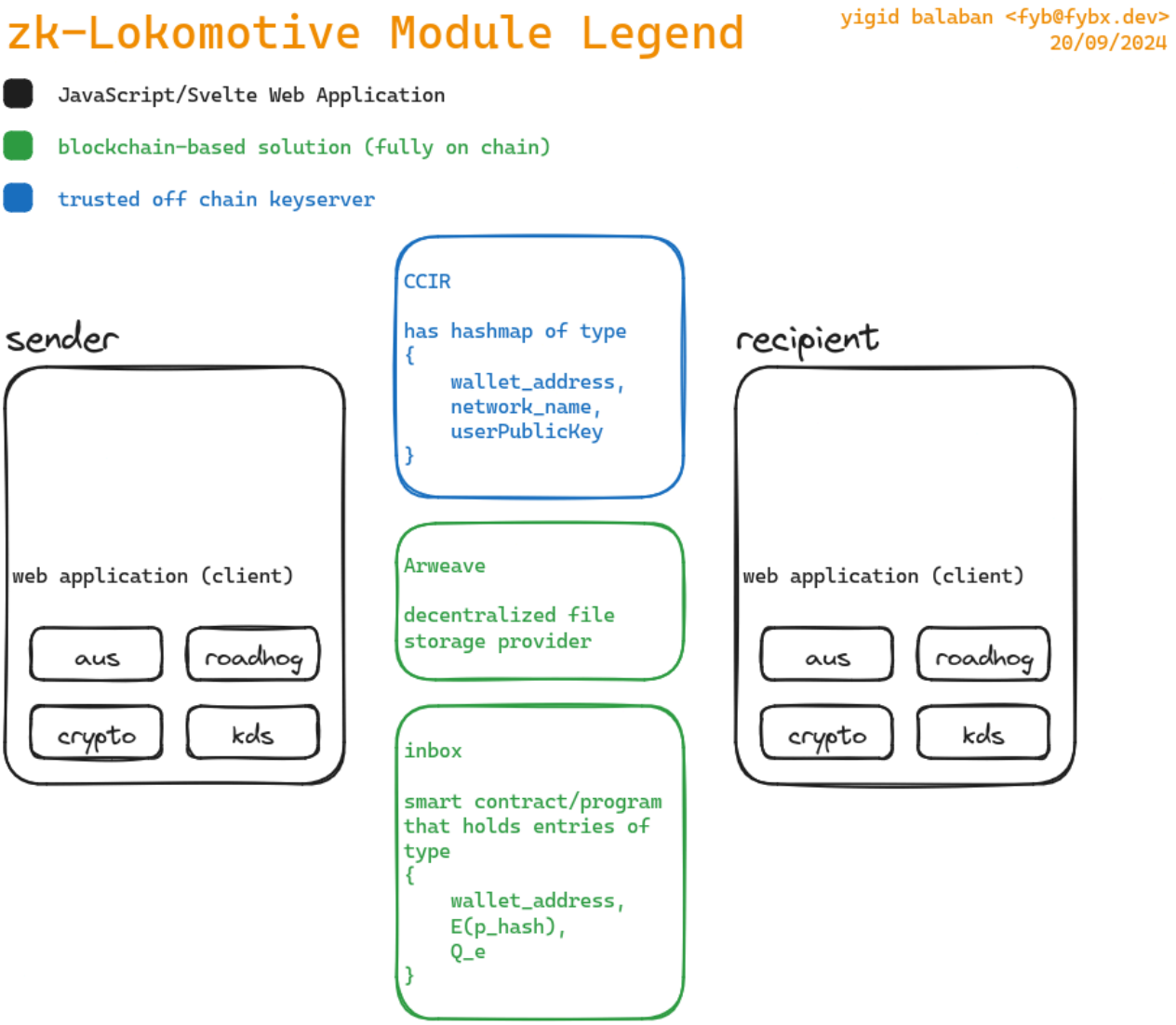
Yigid BALABAN, fyb@fybx.dev

Revision 1
20/09/2024

1. Elements

Our architecture consists of 3 layers, middle one being the platform and the edges being the users.

Figure: zk-Lokomotive Module Legend



Difference between installations

We offer enterprise-level solutions where the platform is full on-chain for an unmatched privacy potential. With the use of a decentralised "CCIR", we provide identity proving and account management in blockchain, removing any centralised layers where data can be tampered with.

See "Appendix A" for the enterprise module legend.

Modules

The infrastructure consists of 3 modules, with the JavaScript/Svelte web application consisting of 4 submodules. Here's the breakdown of each module:

JavaScript/Svelte web application (client)

The client is the user-facing module of zk-Lokomotive, allowing people to send and receive files using the accounts they "create" through linking their wallet accounts.

Submodule: aus

This submodule provides the functionality of communicating with distributed file chains and/or decentralised networks like IPFS. In our current implementation (see "crypto sysarch", revision 3) we are utilising Arweave.

Submodule: roadhog

This submodule interfaces with the several blockchain networks utilised in our platform to perform specific objectives by calling or querying smart contracts, Solana programs, etc.

Submodule: crypto

This submodule performs cryptographic applications. In our current implementation we support elliptic curve integrated encryption scheme through ecies/rs-wasm package which consumes the Rust crate eciesrs.

Submodule: kds

This submodule generates pseudo-random mnemonics using BIP-39 that are then derived to elliptic curve keypairs.

Platform-level: CCIR

Cross-Chain Identity Registry, or CCIR for short is the address book of zk-Lokomotive users, linking their wallet addresses to their elliptic public keys. On CCIR, the wallet user can only add or modify information about themselves after proving that they own the private key to that wallet account (simply, authenticating), but everyone is allowed to do queries from a public EC key to the wallet address, or from a wallet address to the public EC key.

The CCIR holds the following information:

```
{
        "wallet_addresses": [
                { "address": "0x123...", "network": "ethereum" }
        ],
        "userPublicKey": "0x123...",
        "main_wallet_address": "0x123..."
}
```

Platform-level: Arweave

Following our current (rev.3) architecture, we are using Arweave for a decentralised file storage solution.

Platform-level: Inbox

Inboxes are on-chain and network-specific implementations which the client calls using the submodule roadhog when querying for incoming files, or sending a file.

The inbox can be logically defined as:

```
type message = { q_e: string; e_plink: string; uPK: string; }
let inbox : message StringMap.t ref = ref StringMap.empty
```

Appendix A

Figure: zk-Lokomotive Enterprise Deployment Module Legend