Generating accounts

Representing accounts

Representing accounts using the Namada SDK is straightforward. An account on Namada is defined by its public key(s) and private key(s) (plural for multisignatures). The public key(s) is/are used to identify the account and the private key is used to sign transactions. In the below snippet, we represent the account using the public key and private key.

use namada_sdk :: types :: key :: common :: { PublicKey , SecretKey }; struct

SimpleAccount { public_key :

PublicKey , private_key :

SecretKey } For a multisignature account, we can represent this through a vector of keys.

use namada_sdk :: types :: key :: common :: { PublicKey , SecretKey }; struct

MultisigAccount { public_keys :

Vec < PublicKey

, private_keys :

Vec < SecretKey

} Multisignature accounts, because they are initialized by an on-chain transaction, will always have their public key revealed to the ledger. However, when keypairs are generated in an offline fashion, the user must submit a transaction in order to reveal their public key. Because of this, it is helpful to add the fieldrevealed to the account struct.

use namada_sdk :: types :: key :: common :: { PublicKey , SecretKey }; struct

Account { public_key :

PublicKey , private_key :

SecretKey , revealed :

bool }

Revealing the public key of an implicit account

In order to reveal the public key of an implicit account, the user must submit a transaction to the ledger.

use namada_sdk :: io :: NullIo ; use namada_sdk :: NamadaImpl ; use namada_sdk :: types :: chain :: ChainId ;

// Define the namada implementation (assuming we have a wallet, http_client, and shielded_ctx) let

mut namada =

NamadaImpl :: new (& http_client, &mut wallet, &mut shielded_ctx, & NullIo) .await . expect ("unable to construct Namada object") . chain_id (ChainId :: from_str ("shielded-expedition.88f17d1d14") . unwrap ());

// Generate an account (assuming sk is a SecretKey) let account =

Account { public_key : sk . to_public (), private_key : sk, revealed :

false , };

// Build the reveal pk transaction using the NamadaImpl object let reveal_tx_builder = namada . new_reveal_pk (account . public_key . clone ()) . signing_keys (vec! [account . private_key . clone ()]); let (mut reveal_tx, signing_data, _) = reveal_tx_builder . build (namada) .await . expect ("unable to build reveal pk tx"); // Sign the transaction namada . sign (&mut reveal_tx, & reveal_tx_builder . tx, signing_data) .await . expect ("unable to sign reveal pk tx"); // Submit the signed tx to the ledger for execution namada . submit (reveal_tx . clone (), reveal_tx_builder) Once the public key is revealed, the account can be used to sign transactions.

Setting up a wallet Constructing transfers