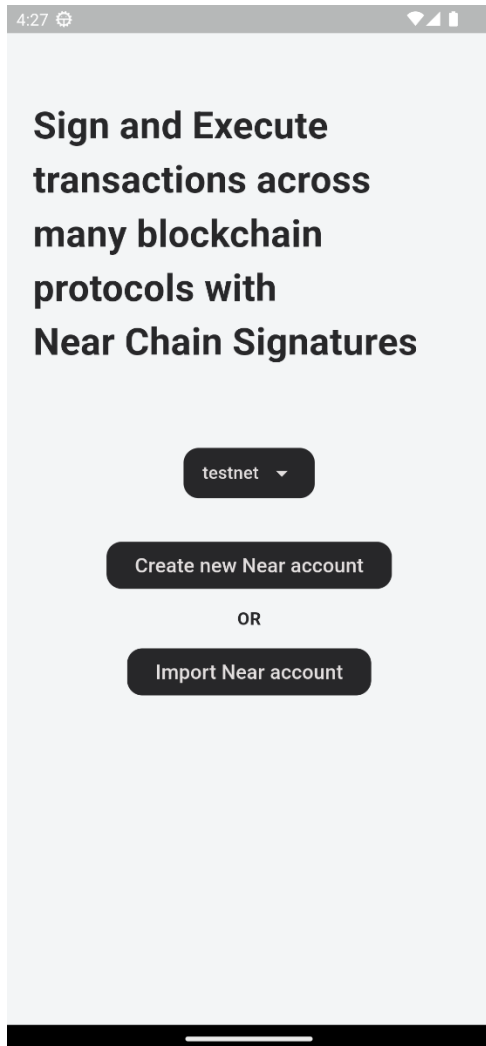


## Near Chain Signatures usage example:

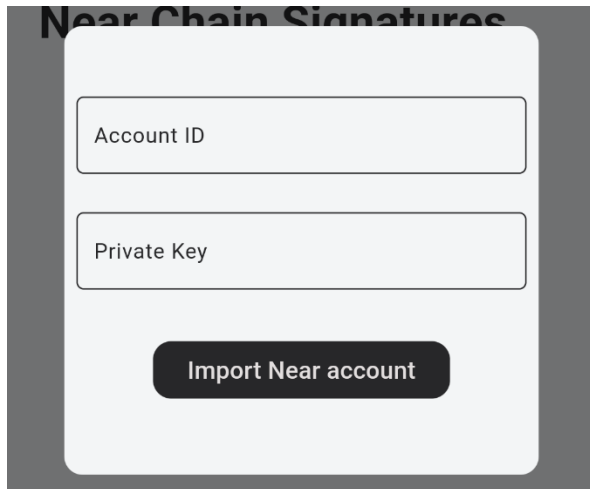
To begin working with Near Chain Signatures, you need to create or import a Near account.

Our application's initial screen looks like this:



Here, you can choose the network (testnet/mainnet) to work with. Currently, only the testnet is available. The "Create new Near account" button allows you to create a new, funded Near account from scratch. Alternatively, you can import your existing Near account by tapping on "Import Near Account" and entering your account credentials. The private key must be in the format "ed25519: ...".

The Import Dialog looks like this:



Near Chain Signatures

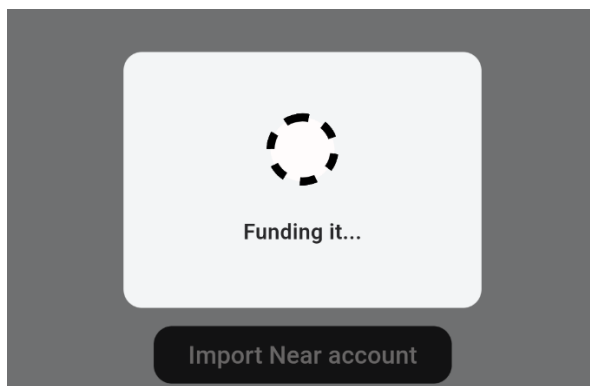
Account ID

Private Key

Import Near account

This image shows a dialog box titled "Near Chain Signatures". It contains two input fields: "Account ID" and "Private Key". Below these fields is a dark button labeled "Import Near account".

The Near account creation process looks like this:



After creating or importing a Near account, you'll see the main action screen:

The screenshot shows the 'Near Chain Signatures' app interface. At the top, the status bar shows the time 4:39 and various icons. The app header includes the 'Near Chain Signatures' logo and a menu icon. Below this, the 'Your Near Account' section displays the account address: 'a9c3e6cf4a7ae4849f6a2a3ba99239649778ddda4a80c672556424c38ad7c6bd' and the balance: '1.0 NEAR'. A 'Blockchain:' label is followed by a button labeled 'ETHEREUM'. Below this is a 'Derivation path settings' button with a dropdown arrow. The next section shows the Ethereum address: '0xbc85daf1eb76d04d313bfc446d6d9210cdfb2aa0' and the balance: '0.0 ETH'. An 'Action to call:' label is followed by a dropdown menu currently set to 'Transfer'. Below this are two input fields: 'Receiver address' and 'Amount'. At the bottom is a 'Sign and send transaction' button.

4:39

Near Chain Signatures

**Your Near Account**

Address: a9c3e6cf4a7ae4849f6a2a3ba99239649778ddda4a80c672556424c38ad7c6bd

Balance: 1.0 NEAR

Blockchain: ETHEREUM

Derivation path settings

Address: 0xbc85daf1eb76d04d313bfc446d6d9210cdfb2aa0

Balance

**0.0 ETH**

Action to call: Transfer

Receiver address

Amount

Sign and send transaction

At the top of the page, you'll see your Near account ID and its balance. Be cautious with your balance, as calling Near chain transactions requires gas to work.

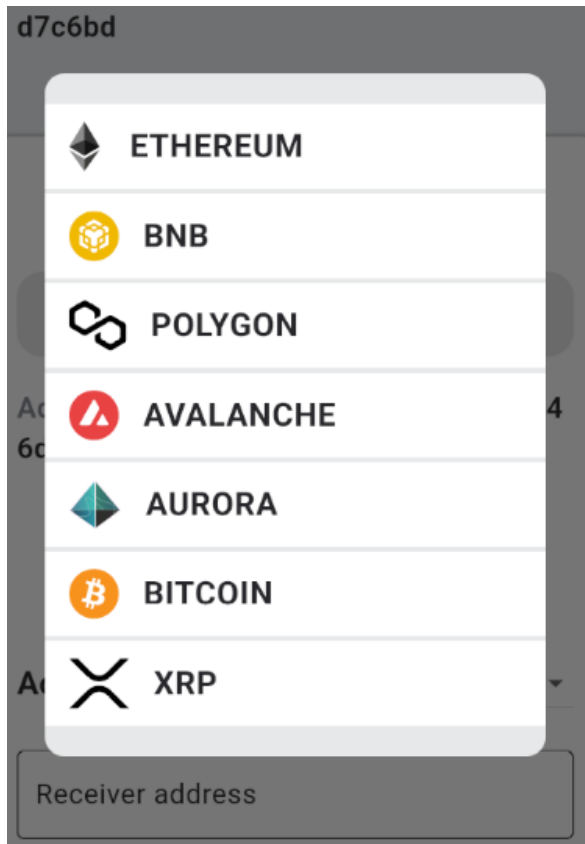
Below, you can see the blockchain information, including the address, balance, and actions that can be performed.

By tapping on the blockchain name, such as "Ethereum," here:

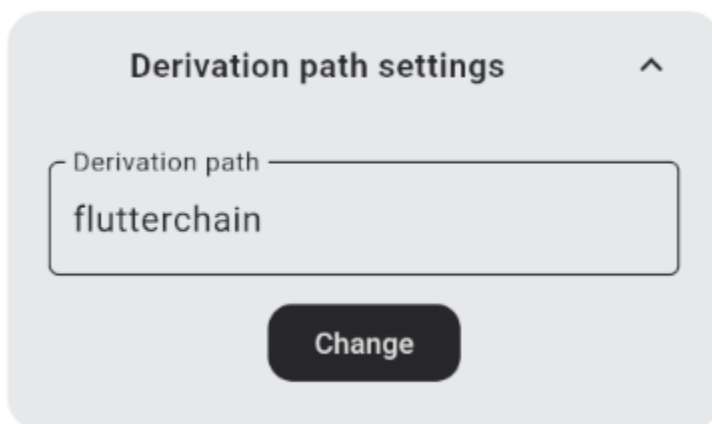
A close-up of the 'Blockchain: ETHEREUM' button, showing the text 'Blockchain:' followed by a button labeled 'ETHEREUM'.

Blockchain: ETHEREUM

you can change the blockchain you want to work with:



Additionally, you can generate another blockchain account by changing the "derivation path." By default, "flutterchain" is used, but you can specify your own.



We can perform two types of actions with the blockchain: "Transfer" and "Smart contract call" (the latter is currently only available for EVM blockchains). After entering the receiver's address and the amount of currency, tap on "Sign and send transaction".

Balance

**0.05 ETH** ↻

**Action to call:** Transfer ▼

Receiver address

0x37D7Ee5a200e8Dea2FFbefe30294c3F

Amount

0.03

**Sign and send transaction**

You'll be able to see the status of the executing operation below:

Amount

0.03



Signing transaction with Near MPC...

After successful execution, you'll see the transaction hash and a button to view your transaction result on a blockchain explorer in your browser.

Sign and send transaction

**Tx hash:** 0xed6776d6742a59bf46186646  
e0e3d86502c2e61827e1071e37ba26fe1  
5c3b5b2

View on explorer

To call a smart contract, we need to fill in the following fields:

**Action to call:** Smart Contract Call ▼

Receiver address

0xfF9976782d46CC05630D1f6eBAb18t

Amount

0.0

Smart Contract Function

transfer(address, uint256)

Format: function1(typeOfArgument1,  
typeOfArgument2)

Smart Contract Args

["0xDksdkd...", 1]

Format: [arg1, arg2]

Sign and send transaction

Smart Contract Function field requires the function name of the smart contract and the type of arguments that it takes. If there are no arguments, then we leave it in the format "function1()".

The Smart Contract Args field requires arguments for this function. If there are no arguments, we leave it in the format "[ ]".