Seed phrase and key derivation

There are two separate things we'll want to do:

1. Create a random seed phrase
2. for the user when they visit the crossword puzzle. This will be used if they win and don't have a NEAR account and wish to create one. They can then paste this seed phrase into NEAR Wallet afterward to import their account (which is basically like "logging in" and is currently possible athttps://testnet.mynearwallet.com/recover-seed-phrase
3.).
4. Turn the crossword solution into a key pair
5. , instead of just hashing it.

near-seed-phrase library

We can add thenear-seed-phrase package to our project with:

npm install near-seed-phrase --save

Code snippets for this chapter At this point in the tutorial, it's more difficult to share code snippets that are both meaningful and meant to be copy/pasted into a project.

The snippets provided might differ slightly from the implementation of thecompleted code for chapter 3 , which might be the best place to look for the functioning code.

Generate random seed phrase for new account creation (if the winner doesn't already have an account)

import

{ generateSeedPhrase }

from

'near-seed-phrase' ;

// Create a random key in here let seedPhrase =

generateSeedPhrase ( ) ;

// generateSeedPhrase() returns an object {seedPhrase, publicKey, secretKey} localStorage . setItem ('playerKeyPair' ,

JSON . stringify (seedPhrase)) ;

Parse solution as seed phrase

(This security measure prevents front-running.)

import

{ parseSeedPhrase }

from

'near-seed-phrase' ; // Get the seed phrase from the completed puzzle. // The original puzzle creator would have already called this same function with the same inputs and would have // already called AddKey on this contract to add the key related to this seed phrase. Here, using this deterministic // function, the front-end will automatically generate that same key based on the inputs from the winner. const seedPhrase =

parseSolutionSeedPhrase (data , gridData) ;

// returns a string of space-separated words // Get the public and private key derived from the seed phrase const

{ secretKey , publicKey }

=

parseSeedPhrase (seedPhrase) ;

// Set up the account and connection, acting on behalf of the crossword account const keyStore =

```
new

nearAPI . keyStores . InMemoryKeyStore ( ) ;

// Another type of key const keyPair = nearAPI . utils . key_pair . KeyPair . fromString ( secretKey ) ; await keyStore . setKey ( nearConfig . networkId , nearConfig . contractName , keyPair ) ; nearConfig . keyStore

= keyStore ; const near =

await nearAPI . connect ( nearConfig ) ; const crosswordAccount =

await near . account ( nearConfig . contractName ) ;

// Call the submit_solution method using the discovered function-call access key let transaction =
```

await crosswordAccount . functionCall (…) ; The last line should look familiar. We did something similar in the last chapter, except we used theWalletConnection 's account to do the function call.

This time we're using anInMemoryKeyStore instead of the browser, as you can see toward the middle of the snippet.

Key stores

We have now used almost all the key stores available innear-api-js :

1. UnencryptedFileSystemKeyStore
2. — early on, when we used the NEAR CLI commandnear login
3. , this created a file in our operating system's home directory containing a private, full-access key to our account.
4. BrowserLocalStorageKeyStore
5. — in the last chapter, when the user first logs in, the function-call access key is saved in the browser's local storage.
6. InMemoryKeyStore
7. — for this chapter, we'll simply use the computer's memory to store the private key derived from the crossword solution.

You can have multiple key stores Technically, there's another type of key store called theMergeKeyStore .

Say you want to look for private keys in various directories. You can essentially have a list ofUnencryptedFileSystemKeyStore key stores that look in different places.

Use theMergeKeyStore when you might want to look for a private key in more than one placeEdit this page Last updatedonDec 5, 2023 bygagdiez Was this page helpful? Yes No

Previous Solution as seed phrase Next Linkdrop contract