Viewing Keys

Introduction to Secret Network viewing keys with code examples

Viewing Keys Introduction

Viewing keys are passwords meant to validate users at times when the blockchain cannot. Specifically in queries, the query sender isn't authenticated and the contract doesn't know who is the querier. Therefore viewing keys were invented to provide a way of access control for users:

1. Alice sends a transactionset_viewing_key(password)
2. The contract stores(alice,password)
3. Later on, a query is sent to the contractquery("balance",alice,password)
4.
    - If(alice,password)
5.
    - matches what's in storage, the contract returns Alice's balance to the querier.
6. *
7.

To see an implementation of viewing keys in a Secret smart contract, check out theSecret Labs Viewing Keys example repository

Importing Secret Toolkit

Secret Network developed thesecret-toolkit viewing key package for creating and managing viewing keys in Secret smart contracts. The methods provided include:

- set_seed
- : Sets an initial pseudorandom number generator (PRNG) seed for the store.
- create
- : Creates a new viewing key, saves it to the storage, and returns it.
- set
- : Sets a new viewing key based on a predetermined value.
- check
- : Checks if a viewing key matches an account.
-

To make use of the secret-toolkit viewing keys package, import it into yourcargo.toml file:

```

Copy secret-toolkit={ version="0.9.0", default-features=false, features=[ "utils", "storage", "serialization", "viewing-key", "crypto", ] }

```

Viewing Keys Implementation

If you would like to see an example implementation of Secret Network viewing keys, see theSecret Labs examples repository here .

This contract is designed to create top secret messages that can only be decrypted if the user has the correct viewing key that is associated with thesecret_message struct. Let's review thetry_create_secret_message() function:

```

Copy pubfntry_create_secret_message( deps:DepsMut, env:Env, info:MessageInfo, secret_message:SecretMessage, index:u8, )->StdResult { letmy_account=info.sender.as_str(); letviewing_key=ViewingKey::create(deps.storage,&info,&env,&my_account,b"entropy");

SECRET_MESSAGE.insert(deps.storage,&viewing_key,&secret_message)?; VIEWING_KEY .add_suffix(info.sender.as_bytes()) .insert(deps.storage,&index,&viewing_key)?; Ok(Response::default()) }

```

This function stores a secret message at a specified index in the contract's storage, which is mapped to a viewing key. This ensures that the secret message can only be accessed by the correct viewing key, and that each secret message has its own unique viewing key.

Let's go over it in more detail:

1. A new viewing key is created by callingViewingKey::create
2. . The parameters passed include the mutable dependencies (which includes the storage), the environmentenv
3. , theMessageInfo
4. , the sender account, and a hard-coded entropy valueb"entropy"
5. .
6. The secret message is then stored in the contract's storage and is mapped to the viewing key. TheSECRET_MESSAGE.insert
7. line is performing this task.
8. Next, the viewing key itself is stored in the contract's storage, but this time indexed with a user-definedindex
9. parameter and prefixed by the sender's account address. This is done byVIEWING_KEY.add_suffix(info.sender.as_bytes()).insert(deps.storage, &index, &viewing_key)
10. .
11. Finally, the function returns a defaultResponse
12. instance indicating successful execution of the function.
13.

Additional Viewing Keys methods

check

```

Copy letaccount="user-1".to_string(); letmutdeps=mock_dependencies(); letenv=mock_env(); letinfo=mock_info(account.as_str(),&[]);

letresult=ViewingKey::check(&deps.storage,&account,"fake key"); assert_eq!(result,Err(StdError::generic_err("unauthorized")));

```

set

```

Copy letviewing_key="custom key"; ViewingKey::set(&mutdeps.storage,&account, viewing_key);

```

set_seed

```

Copy ViewingKey::set_seed(&mutdeps.storage,b"seed");

```

Summary

Viewing keys provide a mechanism for access control in blockchain applications when the blockchain itself can't authenticate the query sender.Secret-toolkit allows developers to set seeds, create and check viewing keys, and set a new viewing key based on a predetermined value

Should you have further questions, please reach out ondiscord and a Secret developer will get back to you shortly

Last updated7 months ago On this page *Viewing Keys Introduction * Importing Secret Toolkit * Viewing Keys Implementation * Additional Viewing Keys methods * Summary

Was this helpful? Edit on GitHub Export as PDF