Fortanix®

User Guide

FORTANIX - CUSTODIAL WARM WALLET SOLUTION

VERSION 1.0

Fortanix, Inc. | 800 West El Camino Real | Suite 180 | Mountain View, CA 94040 | United States of America | ☎ +1 (650) 943-2484 |
✉ info@fortanix.com | 🌐 www.fortanix.com

TABLE OF CONTENTS

1.0 INTRODUCTION

This document describes how the **Fortanix "Custodial Warm Wallet"** solution provides an additional layer of security to crypto-currency wallets by incorporating a second factor of authentication (2FA) for transaction signing using Time-based One-Time Passwords (TOTP). This solution comprises a plugin that is securely deployed inside Fortanix Data Security Manager (DSM) Software-as-a-Service (SaaS). This solution also comprises a `Node.js` SDK that makes it easy for wallet backends to interact with Fortanix DSM.

1.1 FORTANIX CUSTODIAL WARM WALLET PLUGIN

The Fortanix "Custodial Warm Wallet" solution implements a Warm Wallet as a Fortanix DSM plugin. The warm wallet supports secure-second factor authentication (2FA) using TOTP, secure key management, and secure transaction signing that enables B2C crypto-currency businesses to ensure that customers' assets are not transferred without their explicit consent.

The plugin is protected by a quorum policy that involves multiple admin users. Once deployed, the plugin code cannot be modified without explicit permissions from multiple administrators.

The plugin performs the following operations:

- Registers users for 2FA using TOTP
- Derives the public key
- Signs data or Ethereum transaction

NOTE:

- The name of the plugin used in the Fortanix "Custodial Warm Wallet" solution is "TOTP ETH Signer".
- The name of the `Node.js` SDK used in the Fortanix "Custodial Warm Wallet" solution is "fortanix-web3-eth-accounts".

2.0 DEFINITIONS

- **Fortanix Data Security Manager**

  Fortanix DSM is the cloud solution secured with Intel® SGX. With Fortanix DSM, you can securely generate, store, and use cryptographic keys and certificates, as well as secrets, such as passwords, API keys, tokens, or any blob of data.

- **Accounts**

  A Fortanix DSM account is the top-level container for security objects managed by the Fortanix DSM. An account is generally associated with an organization, rather than an individual. Security objects, groups, and applications belong to exactly one account. Different accounts are fully isolated from each other. *See support for more information.*

- **Fortanix Data Security Manager Security Objects**

  A security object is any datum stored in Fortanix DSM (for example a key, a certificate, a password, or other security objects). Each security object is assigned to exactly one group. Users and applications assigned to the group have permission to see the security object and to perform operations on it. *See support for more information.*

3.0 SETUP

3.1 CREATE A FORTANIX DSM GROUP

1. To use the Fortanix "TOTP ETH Signer" plugin in Fortanix DSM, you must first create a Fortanix DSM group, and add the Plugin to this group from the Fortanix DSM Plugin Library.

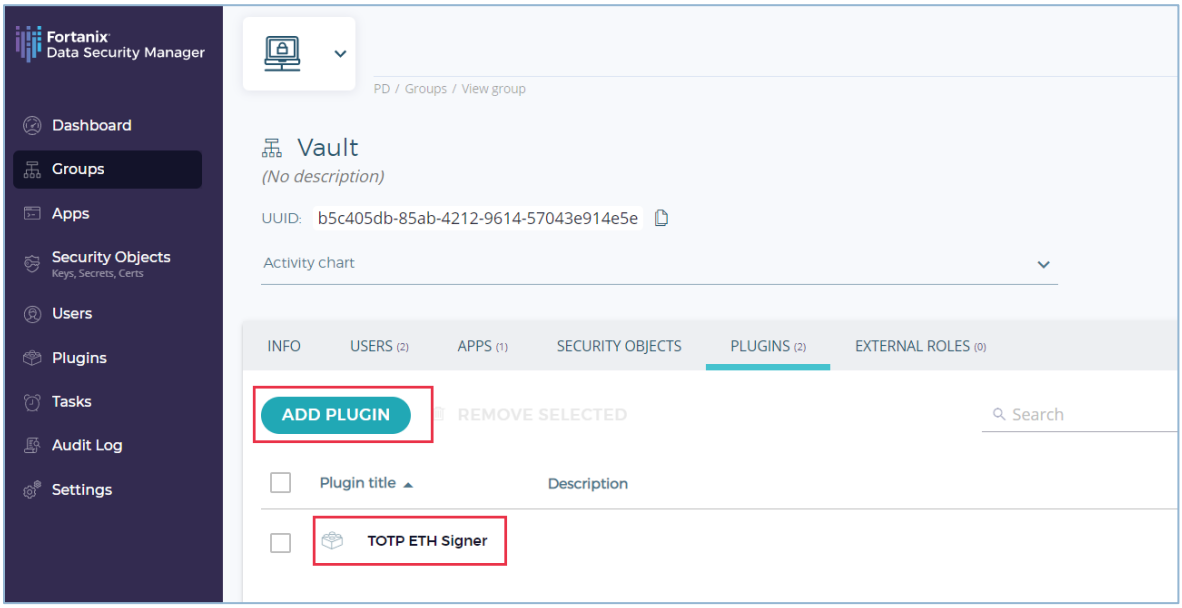

FIGURE 1: IMPORT PLUGIN

Refer to the User's Guide: Plugin Library for steps to access and install the plugin from the Fortanix DSM Plugin Library.

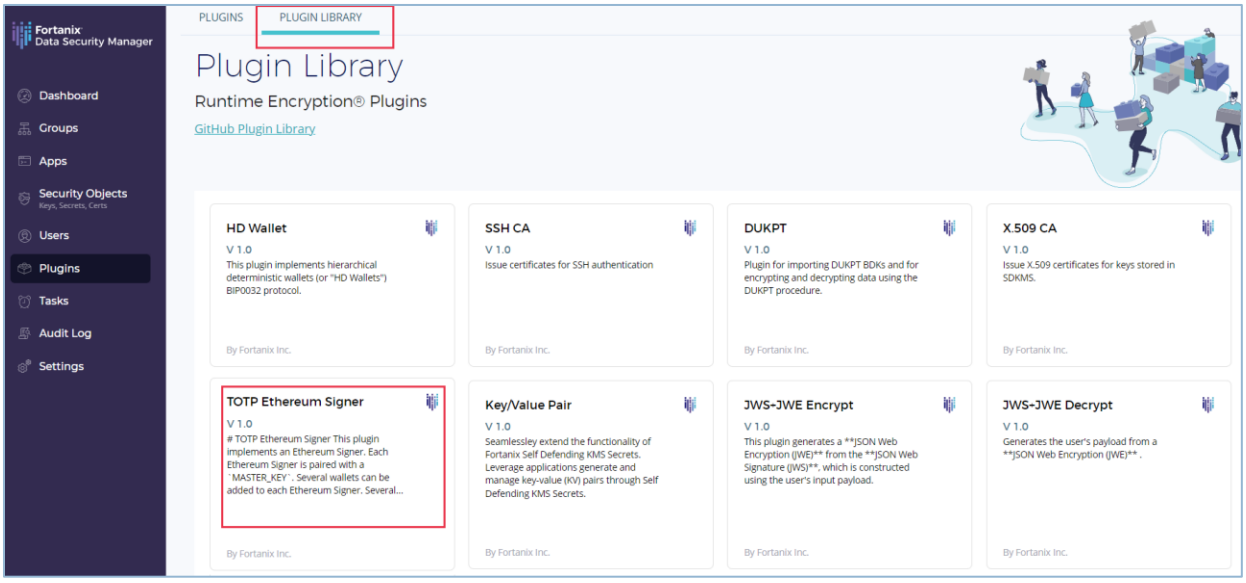

FIGURE 2: INSTALL PLUGIN FROM THE PLUGIN LIBRARY

2. Copy the UUID of the plugin. When using the "fortanix-web3-eth-accounts" SDK, the environment variable `signerId` is assigned this UUID.

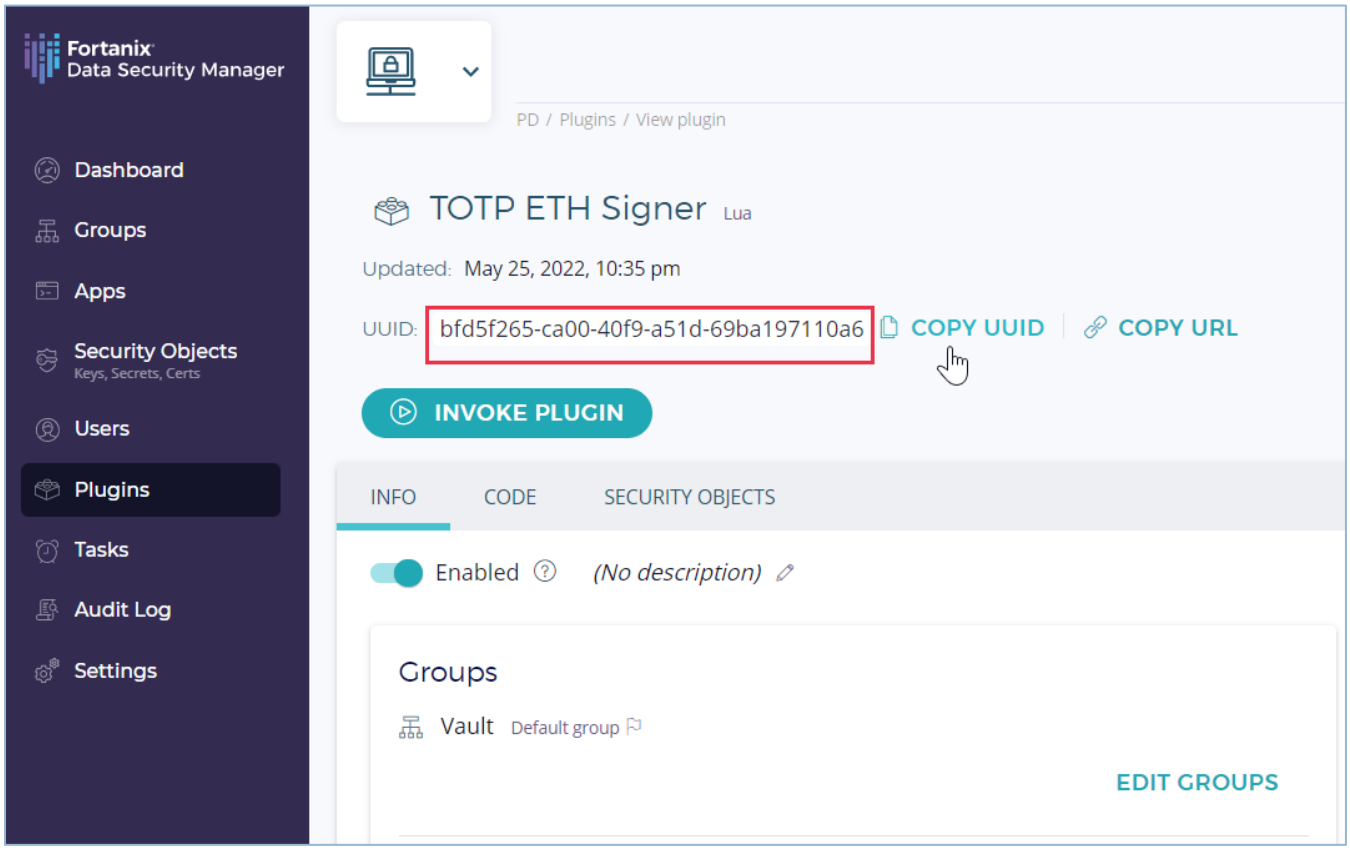

FIGURE 3: PLUGIN UUID

3.2 CONFIGURE A QUORUM POLICY FOR THE GROUP

After creating the Fortanix DSM group and adding the "TOTP ETH Signer" plugin to this group, configure a Quorum Policy for the group to protect the plugin. This will ensure that the plugin code cannot be modified without the approval of the Group Administrator.

1. Go to the detailed view of the group, and click the **INFO** tab.
2. In the **Quorum approval policy** section, click **ADD POLICY** to add a new quorum policy.
3. Configure the Quorum approval policy and click **SAVE POLICY**.

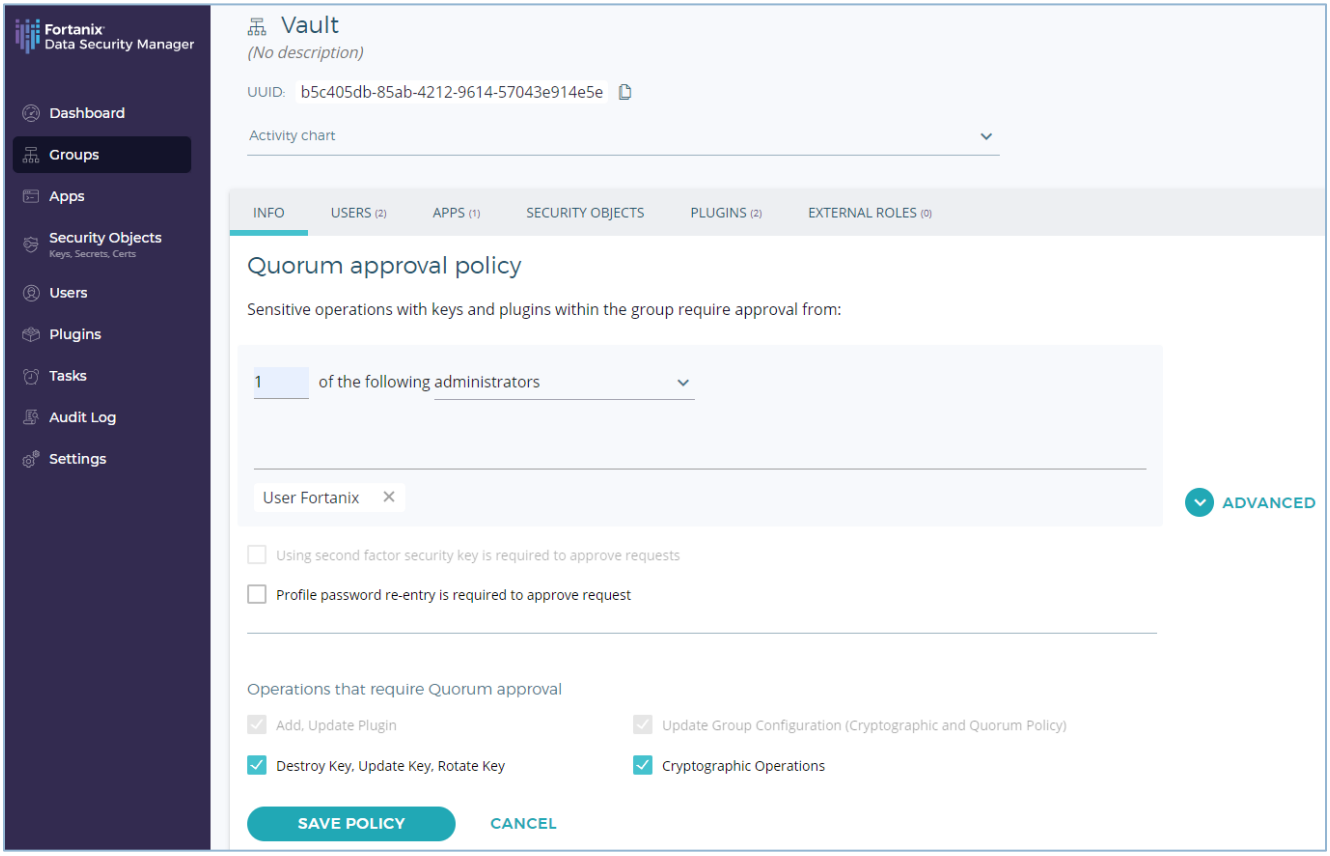

FIGURE 4: CONFIGURE QUORUM APPROVAL POLICY

3.3 CREATE AN APP IN FORTANIX DSM

Create an app in Fortanix DSM for the Ethereum Signer and copy the app's API KEY. When using "fortanix-web3-eth-accounts" SDK, the API Key of this application is used as the value of the environment variable `signerAccessToken` to interact with Fortanix DSM for signing the crypto transactions.

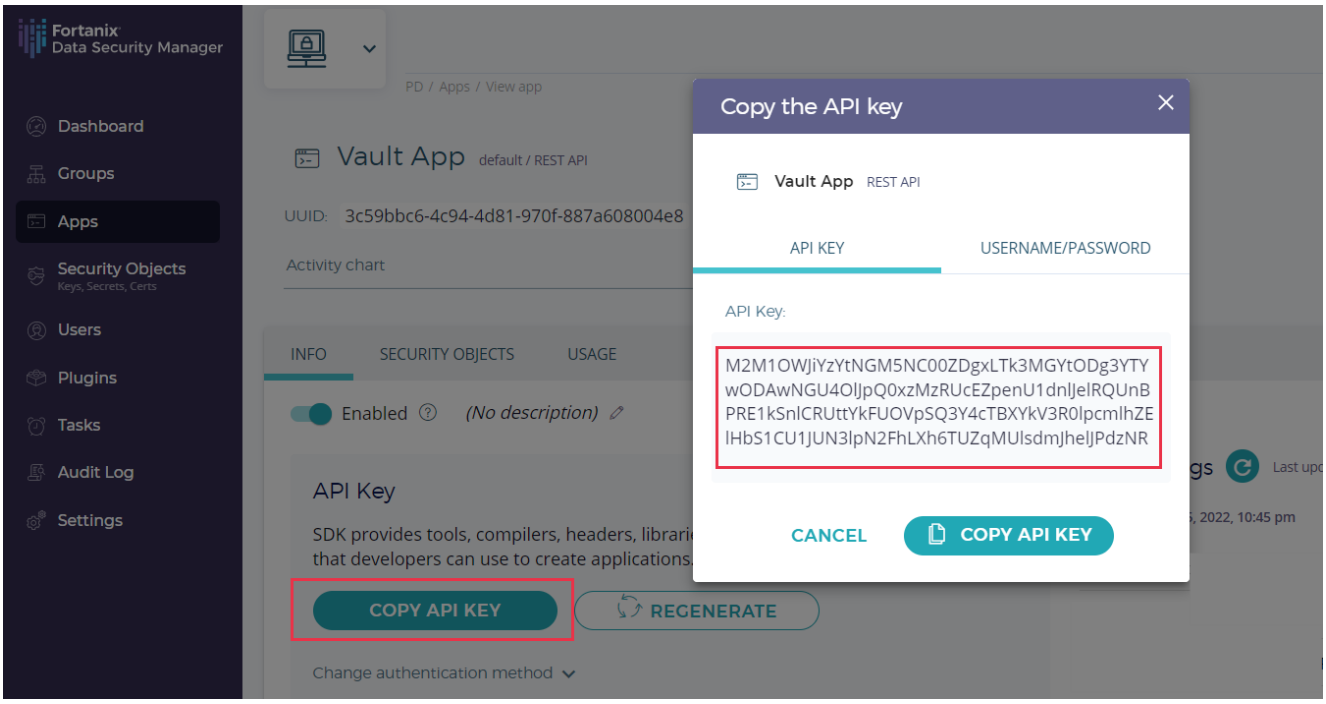

FIGURE 5: CREATE AN APP AND COPY THE API KEY

3.4 GENERATE A MASTER KEY

The "TOTP ETH Signer" plugin is paired with a `MASTER_KEY`. To use the plugin in Fortanix DSM, you must first manually generate this `MASTER_KEY` and initialize the plugin.

You may use the following JavaScript code snippet to generate a master key:

```
const bip39 = require('bip39')
const bitcore = require('bitcore-lib')
const bitcoin = require('bitcoinjs-lib')
const bip32utils = require('bip32-utils')

let mnemonic = bip39.generateMnemonic()
let seed = bip39.mnemonicToSeedSync(mnemonic)

var xprv = bitcore.HDPrivateKey.fromSeed(seed);
console.log("MASTER_KEY: " + xprv.xprivkey)
```

Here is a sample master key:

```
MASTER_KEY =
xprv9s21ZrQH143K31xYSDQpPDxsXRTUcvj2iNHm5NUtrGiGG5e2DtALGdso3pGz6ssrdK4PFmM8
NSpSBHNqPqm55Qn3LqFtT2emdEXVYsCzC2U
```

3.5 IMPORT THE MASTER KEY IN FORTANIX DSM

After manually generating the master key, import this key into Fortanix DSM SaaS as a Secret Raw key in the group created in *Section 3.1*.

NOTE: The `MASTER_KEY` file size must be 888 bits.

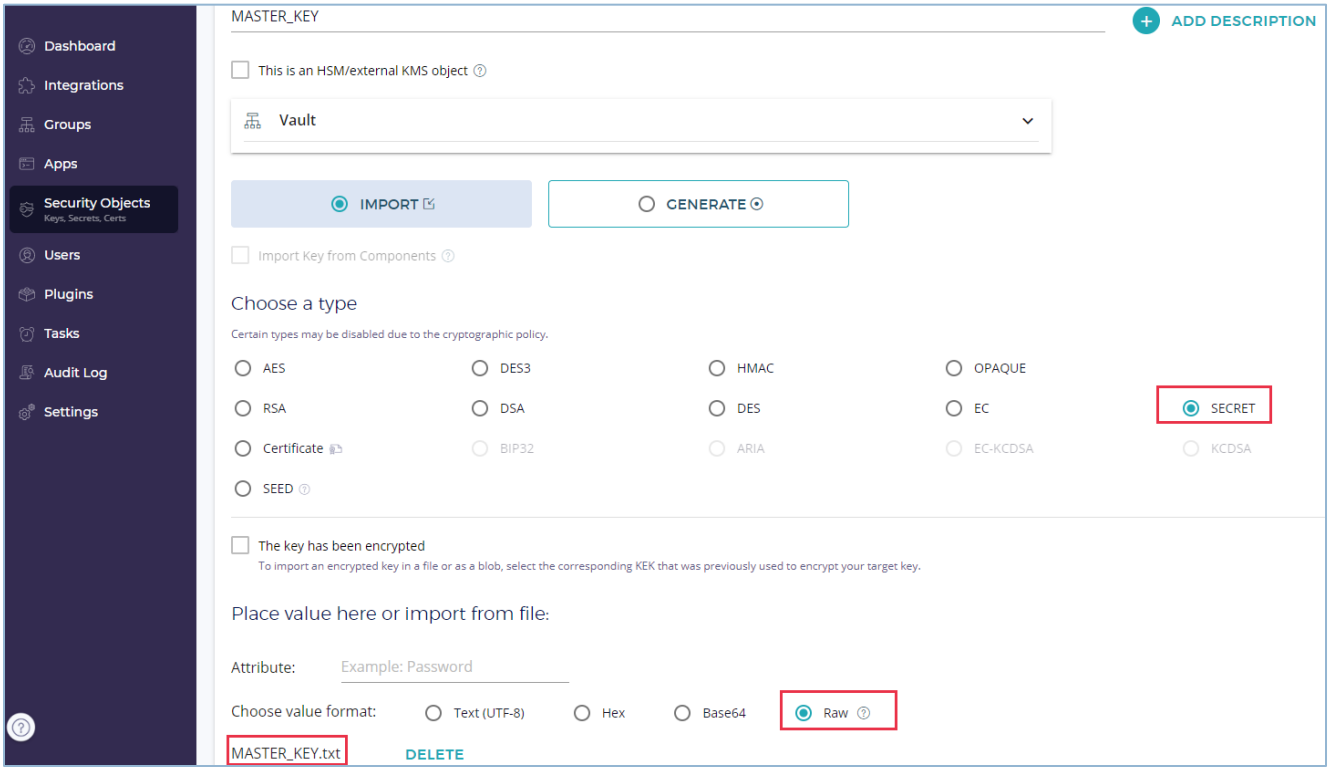

FIGURE 6: IMPORT MASTER KEY

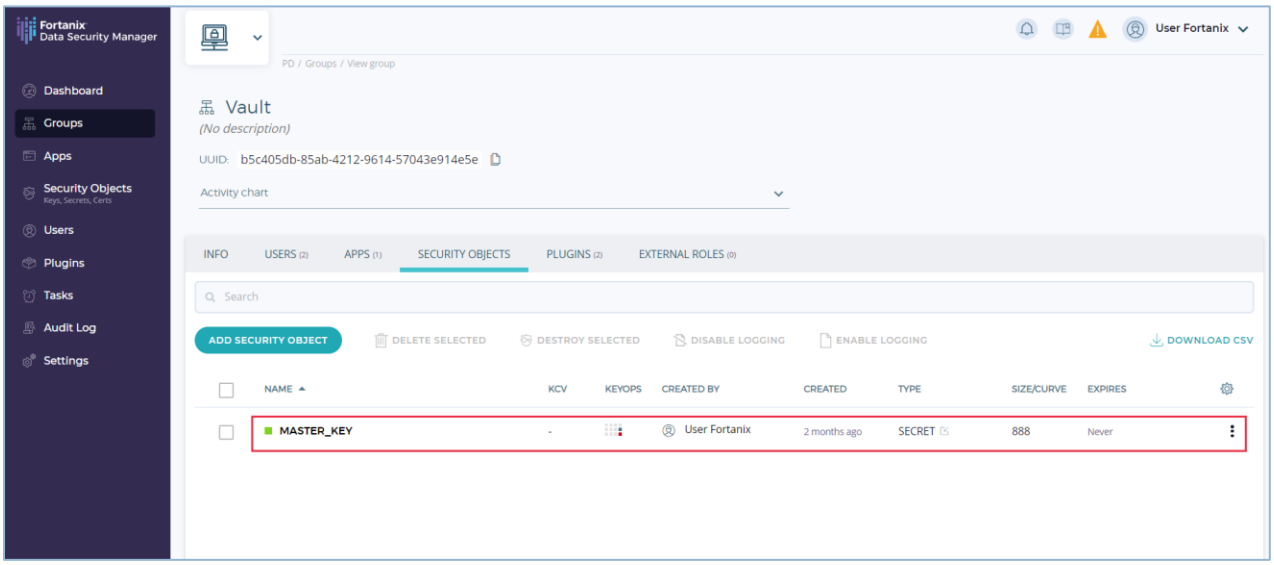

FIGURE 7: MASTER KEY IMPORTED

4.0 PLUGIN OPERATIONS

4.1 USER REGISTRATION

At user registration time, the TOTP authentication system generates a token.

The plugin must be invoked using `walletName` (Label) as the input for deriving the shared token during registration. For example: `walletName= 'alice@acme.com'`

After the registration is successful:

- A security object of type HMAC is created in Fortanix DSM for the wallet name provided. For example: "**totp/alice@acme.com**"

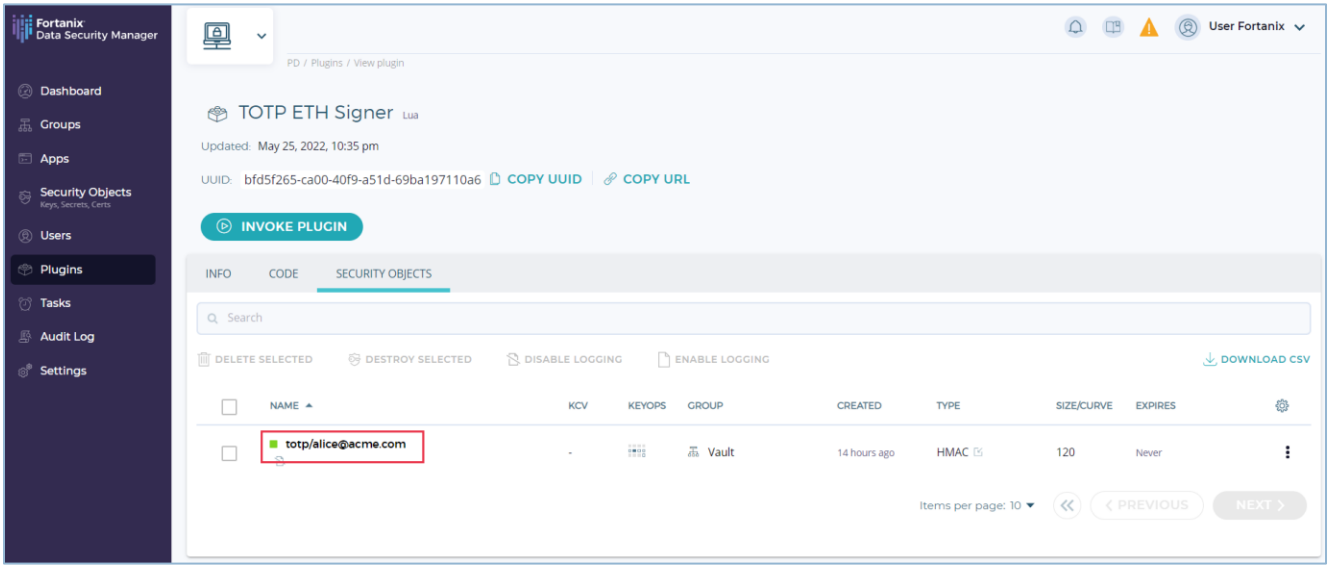

FIGURE 8: SECURITY OBJECT CREATED

- A TOTP path containing the Secret, Label, and Issuer is returned. For example:

`otpauth://totp/alice%40acme.com?secret=DZKFQ3J4DRUOL7MY2HR7BJ7M&issuer=Fortanix%20DSM`

where,

  - Secret: `DZKFQ3J4DRUOL7MY2HR7BJ7M`
  - Label: `alice@acme.com`
  - Issuer: `Fortanix DSM`

The Secret, Label, and Issuer information can be used by a QR code generator to generate a QR code. For a quick check, try generating a QR code with the QR code generator such as https://stefansundin.github.io/2fa-qr/.

4.2 DERIVE THE PUBLIC KEY

Given a `walletName` and a `keyIndex`, the plugin can be used to retrieve the corresponding public key. For example, `"walletName" : alice@acme.com, "keyIndex": "0"`. The retrieved public key is used by the "fortanix-web3-eth-accounts" SDK to retrieve the account address.

4.3 SIGN DATA OR ETHEREUM TRANSACTION

The plugin can be used to sign Ethereum transactions or arbitrary data. If a user has been registered for 2FA, a TOTP code must be provided along with the signing request. The transaction is signed using the following input payload:

```
{
     "operation": "sign",
     "walletName": "string",
     "keyIndex": "number as string",
     "msgHash": "<32-Byte-Message-Hash>"
     "code": "number as string" // code to be provided only if wallet is
      registered for 2FA.
}
```

Where, `msgHash` is a 32-byte message hash that is signed by the "TOTP ETH Signer" plugin. The message hash is generated by the `Node.js` blockchain SDK.

5.0 USING NODE.JS BLOCKCHAIN SDK

5.1 USER REGISTRATION

During user registration time, the TOTP authentication system generates a token.

The SDK must be invoked using `walletName` (Label) as the input for deriving the shared token during registration. For example: `walletName= 'alice@acme.com'`. See the example code here.

After the registration is successful:

- A TOTP path containing the Secret, Label, and Issuer is returned.

5.2 DERIVE THE ACCOUNT ADDRESS

Given a `walletName` and a `keyIndex`, the SDK can be used to retrieve the corresponding account address. See the example code here.

5.3 SIGN DATA OR ETHEREUM TRANSACTION

The SDK can be used to sign Ethereum transactions or arbitrary data. If a user has been registered for 2FA, a TOTP code has to be provided along with the signing request. See the example code here.

6.0 DOCUMENT INFORMATION

6.1 DOCUMENT LOCATION

The latest published version of this document is located at the URL:

https://support.fortanix.com/hc/en-us/articles/6677033204756-Using-Fortanix-Custodial-Warm-Wallet-Solution

6.2 DOCUMENT UPDATES

This document will typically be updated on a periodic review and update cycle.

For any urgent document updates, please send an email to: support@fortanix.com