Usage tKey Android SDK

Once you've installed and successfully instantiated tKey and initialized the service provider in your constructor, you can use it to authenticate your users and generate their tKey shares. Further, you can use various functions exposed by the tKey SDK and its modules to manage different aspects of your users' authentication needs.

ThresholdKey

â

Natively, the instance oftKey , (i.e.,ThresholdKey) returns many functions. However, we have documented a few relevant ones here.

Log Inâ

The login with the tKey SDK is a two-step process. First, you need to trigger the login process by calling thetriggerLogin() function of the CustomAuth SDK. Using the returned information, use theinitialize() function of the tKey to generate the Threshold Key corresponding to the user.

However, before starting this process, you must set up Custom Authentication on your Web3Auth Dashboard. You mustCreate a Verifier from theCustom Auth section of theWeb3Auth Developer Dashboard with your desired configuration.

tip For further information on how to set up and use a verifier, please visit theCustom Authentication Documentation .

Triggering Login and Initializing Service Providerâ

import

org . torusresearch . customauth . CustomAuth ;

private

CustomAuth torusSdk ; MainActivity activity =

((MainActivity)

requireActivity ( )) ;

CustomAuthArgs args =

new

CustomAuthArgs ("https://scripts.toruswallet.io/redirect.html" , TorusNetwork . TESTNET , "torusapp://org.torusresearch.customauthandroid/redirect") ;

this . torusSdk =

new

CustomAuth (args , activity) ;

selectedLoginVerifier

new

LoginVerifier (name :

"Google" , typeOfLogin :

LoginType . GOOGLE , clientId :

GOOGLE_CLIENT_ID , verifier :

GOOGLE_VERIFIER) ;

torusLoginResponseCf

torusSdk . triggerLogin (new

SubVerifierDetails (selectedLoginVerifier . getTypeOfLogin ( ) , selectedLoginVerifier . getVerifier ( ) , selectedLoginVerifier . getClientId ( )) . setPreferCustomTabs (true) . setAllowedBrowsers (allowedBrowsers)) ;

torusLoginResponseCf . whenCompleteAsync ((torusLoginResponse , error)

->

{ if

(error !=

null)

{ renderError (error) ; }

else

{ activity . runOnUiThread (( )

->

{ String publicAddress = torusLoginResponse . getPublicAddress ( ) ; activity . postboxKey = torusLoginResponse . getPrivateKey ( ) . toString (16) ; binding . resultView . append ("publicAddress: "

+ publicAddress) ; }) ; } }) ;

activity . tkeyStorage =

new

StorageLayer (enableLogging :

false , hostUrl :

"https://metadata.tor.us" , serverTimeOffset :

2) ;

activity . tkeyProvider =

new

ServiceProvider (enableLogging :

false , postboxKey : activity . postboxKey) ; Generating a private key is an essential step for the tKey to create its share. ThetriggerLogin() function of theCustomAuth is called to accomplish this.

SubVerifierDetails

â

Parameter Type Mandatory Description typeOfLogin LoginType Yes loginProvider to be used. [google ,facebook ,twitch ,reddit ,discord ,apple ,github ,linkedin ,kakao ,twitter ,weibo ,line ,wechat ,email_password , andjwt] verifier String Yes Web3Auth verifier name clientId String Yes login provider's client Id. jwtParams String No Additional JWT parameters to be passed. isNewActivity boolean No isNewActivity Boolean preferCustomTabs boolean No preferCustomTabs allowedBrowsers String[] No String[] array

CustomAuth

â

CustomAuth(CustomAuthArgs, activity)

CustomAuthArgs

Parameter Type Mandatory Description browserRedirectUri String Yes It refers to a page that the browser should use in the login flow, it should have a http or https scheme. e.g.https://scripts.toruswallet.io/redirect.html redirectUri String Yes It refers to a url for the login flow to redirect into your app, it should have a scheme that is registered by your app, for examplecom.mycompany.myapp://redirect network Network Yes Network to be used. [MAINNET ,TESTNET ,CYAN ,AQUA]

Instantiate tKeyâ

activity . appKey =

new

ThresholdKey (metadata :

null , shares :

null , storage : activity . tkeyStorage , provider : activity . tkeyProvider , transitions :

null , lastFetchedCloudMetadata :

null , enableLogging :

false , manualSync :

false) ;

Parametersâ

Parameter Type Description Mandatory metadata Metadata Metadata object containing the metadata details of tKey. No shares ShareStorePolyIdIndexMap Array of ShareStore with PolyId. No storage StorageLayer Takes in the Storage Provider Instance No provider ServiceProvider Takes in the Service Provider Instance No transitions LocalMetadataTransitions Local metadata transitions No lastFetchedCloudMetadata Metadata lastFetchedCloudMetadata No enableLogging boolean This option is used to specify whether to enable logging or not. No manualSync boolean manual sync provides atomicity to your tkey share. If manualSync is true, you should sync your local metadata transitions manually to your storageLayer, which means your storage layer doesnât know the local changes of your tkey unless you manually sync, gives atomicity. Otherwise, If manualSync is false, then your local metadata changes will be synced automatically to your storage layer. If manualSync = true and want to synchronize manually. No Usage activity . postboxKey = torusLoginResponse . getPrivateKey ( ) . toString (16) ;

activity . tkeyStorage =

new

StorageLayer (enableLogging :

false , hostUrl :

"https://metadata.tor.us" , serverTimeOffset :

2) ;

activity . tkeyProvider =

new

ServiceProvider (enableLogging :

false , postboxKey : activity . postboxKey) ;

activity . appKey =

new

ThresholdKey (metadata :

null , shares :

null , storage : activity . tkeyStorage , provider : activity . tkeyProvider , transitions :

null , lastFetchedCloudMetadata :

null , enableLogging :

false , manualSync :

false) ;

Initialize tKeyâ

initialize(params?)

â

Once you have triggered the login process, you're ready to initialize the tKey. This will generate a Threshold Key corresponding to your login provider.

Parametersâ

Parameter Type Description Mandatory importShare String An optional string representing the import share. No input ShareStore An optional ShareStore object representing the input. No neverInitializedNewKey Bool A boolean value indicating whether or not to initialize a new key. No includeLocalMetadataTransitions Bool A boolean value indicating whether or not to include local metadata transitions. No Usage activity . appKey . initialize (activity . postboxKey ,

null ,

false ,

false , result ->

{ }) ;

Get tKey Detailsâ

getKeyDetails()

ThegetKeyDetails() function provides information about the keys created for a particular user. It includes the user's public key X and Y, as well as the share descriptions, number of required shares, total shares, and threshold.

Usage Sampleâ

KeyDetails keyDetails = activity . appKey . getKeyDetails ( ) ;

// Returns a KeyDetails object.

//Required shares keyDetails . getRequiredShares ( )

// Threshold keyDetails . getThreshold ( )

// Total Shares keyDetails . getTotalShares ( )

// Share Descriptions keyDetails . getShareDescriptions ( )

// This is a json object in string format From here, you can know whether the user key can be reconstructed or not.

- If the value ofkeyDetails.getRequiredShares()
- is more than zero, it implies that the threshold hasn't been met yet, and as a result, the key
- can't be reconstructed since the user hasn't generated enough shares.
- When the value ofkeyDetails.getRequiredShares()
- is 0 or less, the user can reconstruct the key. They can then use the shares to generate their
- private key and carry out additional operations on the tKey to manage their keys.

Reconstruct Private Keyâ

The functionreconstruct() reconstructs the private key of the user from the shares. This function returns the private key of the user once the threshold has been met.

activity . appKey . reconstruct (result ->

{ //... KeyReconstructionDetails details =

((Result . Success < KeyReconstructionDetails

) result) . data ; })

// public final class KeyReconstructionDetails { // public String getKey() // public ArrayList getSeedPhrase() // public ArrayList getAllKeys() // } Example activity . appKey . reconstruct (result ->

{ // show loading state if

```
( result instanceof

Result . Error )

{ renderError ( ( ( Result . Error < KeyReconstructionDetails

        ) result ) . exception ) ; // hide loading state }

else

if

( result instanceof

Result . Success )

{ requireActivity ( ) . runOnUiThread ( ( )

->

{ try

{ KeyReconstructionDetails details =

( ( Result . Success < KeyReconstructionDetails

        ) result ) . data ; binding . generateNewShare . setEnabled ( true ) ; }

catch

( RuntimeError e )

{ renderError ( e ) ; }

finally

{ // hide loading state } } ) ; } } )
```

Generate a New Shareâ

The functiongenerateNewShare() generates a new share on the same threshold (e.g, 2/3 -> 2/4). This function returns the new share generated.

```
activity . appKey . generateNewShare ( result ->

{ } ) ;
```

Delete a Shareâ

The functiondeleteShare() deletes a share from the user's shares. This function returns the updated shareStore after the share has been deleted.

```
activity . appKey . deleteShare ( shareIndex , result ->

{ } ) ;
```

Using Modules for Further Operationsâ

To perform advanced operations and manipulate keys, tKey offers modules that can be utilized. As previously stated in theinitialization section, modules need to be configured beforehand to function properly with tKey. After configuration, the respective module's instance is accessible within your tKey instance and can be utilized for additional operations.

Modules Please visit theModules section to view a comprehensive list of available modules and their respective functions.
Edit this page Previous Initialize Next Modules