Using dApp Share in PnP iOS SDK

Web3Auth Infrastructure at a glanceâ

If you go through theWeb3Auth infrastructure you'll notice that to enable the non custodiality of Web3Auth, we split the private key into multiple parts, i.e.shares . These shares are a part of the off chain multisig, where multiple shares are stored in different places and can be used to reconstruct the private key dynamically in the user's frontend application. For a glance at the structure of the shares, these are the following:

1. ShareA
2. is managed by a login service via node operators
3. : This share is further split amongst a network of nodes and retrieved via
4. conventional authentication flows.
5. ShareB
6. is stored on the user's device
7. : Implementation is device and system specific. For example, on mobile devices, the share could be
8. stored in device storage secured via biometrics.
9. ShareC
10. is a recovery share
11. : An extra share to be kept by the user, possibly kept on a separate device, downloaded or based on user input with
12. enough entropy (eg. password, security questions, hardware device etc.).

Similar to existing 2FA systems, a user needs to prove ownership of at least 2 out of 3 (2/3) shares, in order to retrieve their private key. This initial setup provides several benefits.

The User Experience in Mobile Platformsâ

The user experience on mobile platforms is very different from the web platforms. This is because the user has to be redirected to a browser where they can login using their socials and then back to the app once they have been successfully authenticated. This user experience shifts the context between two applications, whereas, in the web platforms, the context remains within the browser only.

For the seamless login flow, we need to reconstruct the SharesA andB .Share B is managed by the login service and is provided on successful authentication. Whereas in web platforms,Share A is stored in the browser context. We can still store it in the browser context for mobile devices, but this has a few risks like users accidentally deleting browser data. This is a bigger problem in mobile devices since the user doesn't realize that the browser is being used to login within the app and clearing the browser data can cause their logins to fail. Hence, to tackle this issue, Web3Auth issues a dApp Share, ie. a backup share that can be stored by the app developer directly within their application and used to reconstruct the private key after successful login by the user.

dApp Share in iOSâ

Web3Auth issues a dApp Share, ie. a backup share that can be stored by the app developer directly within their application and used to reconstruct the private key after successful login by the user.

After a successful login from a user, the user details are returned as a response to the application in mobile devices.

Sample Response in iOSâ

{ "userInfo": { "email": "w3a-heroes@web3auth.com", "name": "Web3Auth Heroes", "profileImage": "https://lh3.googleusercontent.com/a/Ajjjsdsmdjmnm...", "verifier": "torus", "verifierId": "w3a-heroes@web3auth.com", "typeOfLogin": "google", "aggregateVerifier": "w3a-google-sapphire", "dappShare": "", // 24 words of seed phrase will be sent only incase of custom verifiers "idToken": "", "oAuthIdToken": "", // will be sent only incase of custom verifiers "oAuthAccessToken": "", // will be sent only incase of custom verifiers "isMfaEnabled": false // Returns whether the user has enabled MFA or not } } If you notice, the reponses has a field calleddappShare which is a 24 words seed phrase that can be used to reconstruct the private key. This dappShare is a suplement to theShare A and represents half of the private key. The application can store the dApp share in their own application local storage safely.

Now, while logging in, the user can use their social accounts to obtain one share, and the application provides the dApp Share, removing the need to store the share in the browser context and enabling user to login seamlessly. This can be done by passing over the stored dApp share value in the login function.

note One major thing to note here is that thedappShare is only available for custom verifiers and not the standard Web3Auth verifiers. This is done to make sure that an application only has access to the corresponding share to the private key of their application's user. Hence, to use dApp Share, one has to use the custom authentication feature of Web3Auth. Also, the dApp Share is only returned to users who have enabled 2FA to their account. Web3Auth ( ) . login (W3ALoginParams (

loginProvider : provider , dappShare =

"<24 words seed phrase>"))

Exampleâ

import

Foundation import

Web3Auth

class

ViewModel :

ObservableObject

{ var web3Auth :

Web3Auth ? @Published

var loggedIn :

Bool

=

false @Published

var user :

Web3AuthState ? @Published

var isLoading =

false @Published

var navigationTitle :

String

=

""

func

setup ( )

async

{ guard web3Auth ==

nil

else

{

return

} await

MainActor . run (body :

{ isLoading =

true navigationTitle =

"Loading" }) web3Auth =

await

```
Web3Auth ( W3AInitParams ( clientId : clientId , network : network ) )

await

MainActor . run ( body :

{ if

self . web3Auth ? . state !=

nil

{ user = web3Auth ? . state loggedIn =

true } isLoading =

false navigationTitle = loggedIn ?

"UserInfo"

:

"SignIn" } ) }

func

login ( provider :

Web3AuthProvider )

{ Task

{ do

{ let result =

try

await web3Auth . login ( W3ALoginParams ( // provider can be .GOOGLE, .FACEBOOK, .APPLE etc loginProvider : provider
, dappShare :

"<24 words seed phrase>" ) )

await

MainActor . run ( body :

{ user = result loggedIn =

true } )

}

catch

{ print ( "Error" ) } } } }
```

Edit this page Previous Multi Factor Authentication Next Web3Auth PnP React Native SDK