

# Using Core Kit SFA Flutter SDK

After successfully installing and initializing SingleFactorAuth, you can use it to authenticate your users and obtain their private and public keys.

The SingleFactorAuth instance natively provides the following functions:

- getKey(LoginParams)
- - Returns the private key using the verifier
- ,verifierId
- &idToken
- .
- getAggregateKey(LoginParams)
- - Returns the aggregate private key using the verifier
- ,verifierId
- &idToken
- .

## getKey()

[^](#)

To obtain a user's private key using the Web3Auth SFA Flutter SDK, you can call the `getKey()` function.

Variable Type Description Mandatory LoginParams object LoginParams Options Parameters Yes

### Returns [^](#)

Future < TorusKey

`getKey ( LoginParams loginParams )` On successful call, the `getKey()` function returns a `Future` instance.

### LoginParams

[^](#)

- Table
- Class

Parameter	Type	Mandatory	Description	verifier	String	Yes	The verifier string obtained from the Web3Auth Dashboard
verifierId	String	Yes	The verifierId of the user	idToken	String	Yes	The idToken of the user
aggregateVerifier	String	No	The aggregate verifier string obtained from the Web3Auth Dashboard class				

LoginParams

```
{ final
```

```
String verifier ; final
```

```
String verifierId ; final
```

```
String idToken ; final
```

```
String ? aggregateVerifier ;
```

```
LoginParams ( { required this . verifier , required this . verifierId , required this . idToken , this . aggregateVerifier , } ) ;
```

```
Map < String ,
```

```
dynamic
```

```
toJson ( )
```

```
{ return
```

```
{ 'verifier' : verifier , 'verifierId' : verifierId , 'idToken' : idToken , 'aggregateVerifier' : aggregateVerifier , } ; }
```

```
Future < TorusKey
```

```
getKey ( LoginParams loginParams )
```

```
async
```

```
{ try
```

```
{ Map < String ,
```

```
dynamic
```

## loginParamsJson

```
loginParams . toJson ( ) ; loginParamsJson . removeWhere ( ( key , value )
```

```
=
```

## value

```
null ) ; final
```

```
String torusKeyJson =
```

```
await _channel . invokeMethod ( 'getTorusKey' , jsonEncode ( loginParams ) , ) ;
```

```
return
```

```
torusKeyFromJson ( torusKeyJson ) ; }
```

```
on
```

```
PlatformException
```

```
catch
```

```
( e )
```

```
{ throw
```

```
_handlePlatformException ( e ) ; } } Usage Future < TorusKey
```

```
getKey ( )
```

```
{ return _singleFactorAuthFlutterPlugin . getKey ( LoginParams ( verifier :
```

```
'YOUR_VERIFIER_NAME' , verifierId :
```

```
'YOUR_VERIFIER_ID' , idToken :
```

```
'YOUR_ID_TOKEN' , ) , ) ; }
```

## getAggregateKey()

[â](#)

To obtain a user's Torus key using the Web3Auth SFA Flutter SDK, you can call the `getAggregateKey()` function.

Variable Type Description Mandatory LoginParams object LoginParams Options Parameters Yes

### Returns<sup>â</sup>

Future < TorusKey

`getAggregateKey ( LoginParams loginParams )` On successful call, the `getAggregateKey()` function returns a `Future` instance.

### LoginParams

[â](#)

- Table
- Class

Parameter Type Mandatory Description verifier String Yes The verifier string obtained from the Web3Auth Dashboard  
 verifierId String Yes The verifierId of the user idToken String Yes The idToken of the user aggregateVerifier String No The  
 aggregate verifier string obtained from the Web3Auth Dashboard class

LoginParams

```
{ final
```

```
String verifier ; final
```

```
String verifierId ; final
```

```
String idToken ; final
```

```
String ? aggregateVerifier ;
```

```
LoginParams ( { required this . verifier , required this . verifierId , required this . idToken , this . aggregateVerifier , } ) ;
```

```
Map < String ,
```

```
dynamic
```

```
toJson ( )
```

```
{ return
```

```
{ 'verifier' : verifier , 'verifierId' : verifierId , 'idToken' : idToken , 'aggregateVerifier' : aggregateVerifier , } ; }
```

```
Future < TorusKey
```

```
getAggregateKey ( LoginParams loginParams )
```

```
async
```

```
{ try
```

```
{ Map < String ,
```

```
dynamic
```

## loginParamsJson

```
loginParams . toJson ( ) ; loginParamsJson . removeWhere ( ( key , value )
```

```
=
```

## value

```
null ) ;
```

```
final
```

```
String torusKeyJson =
```

```
await _channel . invokeMethod ( 'getAggregateTorusKey' , jsonEncode ( loginParams ) , ) ;
```

```
return
```

```
torusKeyFromJson ( torusKeyJson ) ; }
```

```
on
```

```
PlatformException
```

```
catch
```

```
( e )
```

```

{ throw
_handlePlatformException ( e ) ; } } Usage Future < TorusKey
getAggregateKey ( )
{ return _singleFactorAuthFlutterPlugin . getAggregateKey ( LoginParams ( verifier :
'YOUR_VERIFIER_NAME' , verifierId :
'YOUR_VERIFIER_ID' , idToken :
'YOUR_ID_TOKEN' , aggregateVerifier :
'YOUR_AGGREGATE_VERIFIER_NAME' , ) ) ; } } Note Web3Auth SFA Flutter SDK only works for users who havenot
enabled MFA . MFA enabled users For MFA enabled users, you'll see an Error message.

```

## Session Management

We have included Session Management in this SDK, so calling the initialize function to get the TorusKey value without re-logging in the user if a user has an active session will return the TorusKey, otherwise, it will return null.

```

final
TorusKey ? torusKey =
await _singleFactorAuthFlutterPlugin . initialize ( ) ;

```

## Example

```

lib/main.dart
main.dart import
'dart:async' ; import
'dart:developer' ; import
'dart:io' ;
import
'package:flutter/material.dart' ; import
'package:single_factor_auth_flutter/input.dart' ; import
'package:single_factor_auth_flutter/output.dart' ; import
'package:single_factor_auth_flutter/single_factor_auth_flutter.dart' ; import
'./utils.dart' ;
void
main ( )
{ runApp ( const
MyApp ( ) ) ; }
class
MyApp
extends
StatefulWidget
{ const
MyApp ( { super . key } ) ;

```

```

@override State < MyApp
createState ( )

=

_MyAppState ( ) ; }

class _MyAppState extends
State < MyApp

{ final _singleFactorAuthFlutterPlugin =
SingleFactAuthFlutter ( ) ; String _result =
" ; bool logoutVisible =
false ; TorusNetwork torusNetwork =
TorusNetwork . testnet ;

@override void
initState ( )

{ super . initState ( ) ; initSdk ( ) ; }

Future < void
initSdk ( )

async

{ if

( Platform . isAndroid )

{ await

init ( ) ; initialize ( ) ; }

else

if

( Platform . isIOS )

{ await

init ( ) ; initialize ( ) ; }

else

{ } }

Future < void

init ( )

async

{ await _singleFactorAuthFlutterPlugin . init ( Web3AuthNetwork ( network : torusNetwork ) ) ; }

Future < void

initialize ( )

async

{ log ( "initialize() called" ) ; final

TorusKey ? torusKey = await _singleFactorAuthFlutterPlugin . initialize ( ) ; if

```

```
( torusKey !=  
null )  
{ setState ( ( )  
{ _result =  
"Private Key : { torusKey . privateKey } " ; } ) ; } }  
@override Widget  
build ( BuildContext context )  
{ return  
MaterialApp ( home :  
Scaffold ( appBar :  
AppBar ( title :  
const  
Text ( 'SingleFactorAuthFlutter Example' ) , ) , body :  
SingleChildScrollView ( child :  
Center ( child :  
Column ( mainAxisAlignment :  
MainAxisAlignment . center , children :  
[ const  
Padding ( padding :  
EdgeInsets . all ( 8.0 ) , ) , Visibility ( child :  
Column ( children :  
[ const  
SizedBox ( height :  
50 , ) , const  
Icon ( Icons . flutter_dash , size :  
80 , color :  
Color ( 0xFF1389fd ) , ) , const  
SizedBox ( height :  
40 , ) , const  
Text ( 'Web3Auth' , style :  
TextStyle ( fontWeight :  
FontWeight . bold , fontSize :  
36 , color :  
Color ( 0xFF0364ff ) , ) , const  
SizedBox ( height :  
10 , ) , const  
Text ( 'Welcome to SingleFactorAuthFlutter Demo' , style :
```

```

TextStyle ( fontSize :
14 ) , ) , const
SizedBox ( height :
20 , ) , const
Text ( 'Get TorusKey' , style :
TextStyle ( fontSize :
12 ) , ) , const
SizedBox ( height :
20 , ) , ElevatedButton ( onPressed :
_getKey ( getAggregateKey ) , child :
const
Text ( 'GetTorusKey' ) , ) , ElevatedButton ( onPressed :
( )
=
_initialize ( ) , child :
const
Text ( 'Get Session Response' ) , ) , ] , ) , ) , Padding ( padding :
const
EdgeInsets . all ( 8.0 ) , child :
Text ( _result ) , ) ] , ) , ) , ) , ) ; }
VoidCallback
_getKey ( Future < TorusKey
Function ( ) method )
{ return
( )
async
{ try
{ final
TorusKey response =
await
method ( ) ; setState ( ( )
{ _result =
"Private Key : { response . privateKey } " ; log ( response . publicAddress ) ; } ) ; }
on
PrivateKeyNotGeneratedException
{ log ( "Private key not generated" ) ; }
on

```

```

UnknownException

{ log ( "Unknown exception occurred" ) ; } ; }

Future < void

_initialize ( )

async

{ try

{ final

TorusKey ? response = await _singleFactorAuthFlutterPlugin . initialize ( ) ; setState ( ( )

{ _result =

"Private Key : { response ? . privateKey } " ; log ( response ! . publicAddress ) ; } ) ; }

on

PrivateKeyNotGeneratedException

{ log ( "Private key not generated" ) ; }

on

UnknownException

{ log ( "Unknown exception occurred" ) ; }

Future < TorusKey

getKey ( )

{ return _singleFactorAuthFlutterPlugin . getKey ( LoginParams ( verifier :

'torus-test-health' , verifierId :

'hello@tor.us' , idToken :

Utils ( ) . es256Token ( "hello@tor.us" ) , ) , ) ; }

Future < TorusKey

getAggregateKey ( )

{ return _singleFactorAuthFlutterPlugin . getAggregateKey ( LoginParams ( verifier :

'torus-test-health' , verifierId :

'hello@tor.us' , idToken :

Utils ( ) . es256Token ( "hello@tor.us" ) , aggregateVerifier :

'torus-test-health-aggregate' , ) , ) ; } } lib/utils.dart

```

This Dart code will produce a JSON Web Token (JWT) using the ES256 algorithm. You have the option to use your own implementation or follow our [Bring your own JWT token steps](#). Alternatively, you may choose to use any [social](#) / [federated identity](#) provider that is supported by Web3auth.

```

utils.dart import

'package:dart_jsonwebtoken/dart_jsonwebtoken.dart' ;

class

Utils

{ String

es256Token ( String email )

```



```

{ String token ;

/ Sign /

{ // Create a json web token final jwt =

JWT ( { "sub" :

"email|hello" , "aud" :

"torus-key-test" , "exp" :

DateTime . now ( ) . millisecond , "iat" :

DateTime . now ( ) . millisecond , "iss" :

"torus-key-test" , "email" : email , "nickname" : email . split ( "@" ) [ 0 ] , "name" : email , "picture" :

"" , "email_verified" :

true } ) ;

// Sign it final key =

ECPrivateKey ( "-----BEGIN PRIVATE KEY-----\n"
"MEECAQAwEwYHKoZIzj0CAQYIKoZIzj0DAQcEJzAIAgEBBCCD7oLrcKae+jVZPGx52Cb/IKhdKxpXjl9eGNa1MIY57A=="
"\n-----END PRIVATE KEY-----" ) ; token = jwt . sign ( key , algorithm :

JWTAlgorithm . ES256 ) ;

print ( 'Signed token: \n

token \n' ) ; return token ; } } Edit this page Previous Authentication

```