D4D4D4;--ch-t-background: #1E1E1E;--ch-t-lighter-inlineBackground: #1e1e1ee6;--ch-t-editor-background: #1E1E1E;--ch-t-editor-foreground: #D4D4D4;--ch-t-editor-rangeHighlightBackground: #ffffff0b;--ch-t-editor-infoForeground: #3794FF;--ch-t-editor-selectionBackground: #264F78;--ch-t-focusBorder: #007FD4;--ch-t-tab-activeBackground: #1E1E1E;--ch-t-tab-activeForeground: #ffffff;--ch-t-tab-inactiveBackground: #2D2D2D;--ch-t-tab-inactiveForeground: #ffffff80;--ch-t-tab-border: #252526;--ch-t-tab-activeBorder: #1E1E1E;--ch-t-editorGroup-border: #444444;--ch-t-editorGroupHeader-tabsBackground: #252526;--ch-t-editorLineNumber-foreground: #858585;--ch-t-input-background: #3C3C3C;--ch-t-input-foreground: #D4D4D4;--ch-t-icon-foreground: #C5C5C5;--ch-t-sideBar-background: #252526;--ch-t-sideBar-foreground: #D4D4D4;--ch-t-sideBar-border: #252526;--ch-t-list-activeSelectionBackground: #094771;--ch-t-list-activeSelectionForeground: #fffffe;--ch-t-list-hoverBackground: #2A2D2E; }

Passkeys Signer

In this guide, you will learn how to create a Passkey signer that can be added as a Safe owner and used to initialize any of the kits from the Safe{Core} SDK.

Note: Please always use a combination of passkeys and other authentication methods to ensure the security of your users' assets.

Prerequisites

- Node.js and npm(opens in a new tab)
- .
- Passkeys feature is available only in secure contexts(opens in a new tab)
- (HTTPS), in some or all supporting browsers(opens in a new tab)
- .

Install dependencies

npm yarn pnpm _10 npm install @safe-global/protocol-kit

Steps

Imports

Here are the necessary imports for this guide.

_10 import { extractPasskeyData } from '@safe-global/protocol-kit' In addition, you will need to import a web3 library of your choice to use in the "Get the provider and signer" section. In this guide, we are usingviem .

Create a passkey

Firstly, you need to generate a passkey credential using the WebAuthn API in a supporting browser environment.

_25 const RP_NAME = 'Safe Smart Account' _25 const USER_DISPLAY_NAME = 'User display name' _25 const USER_NAME = 'User name' _25 _25 const passkeyCredential = await navigator.credentials.create({ _25 publicKey: { _25 pubKeyCredParams: [_25 { _25 alg: -7, _25 type: 'public-key' _25 } _25], _25 challenge: crypto.getRandomValues(new Uint8Array(32)), _25 rp: { _25 name: RP_NAME _25 }, _25 user: { _25 displayName: USER_DISPLAY_NAME, _25 id: crypto.getRandomValues(new Uint8Array(32)), _25 name: USER_NAME _25 }, _25 timeout: 60_000, _25 attestation: 'none' _25 } _25 }) After generating thepasskeyCredential object, you need to create a new object with thePasskeyArgType type that will contain therawId and thecoordinates information.

_10 if (!passkeyCredential) { _10 throw Error('Passkey creation failed: No credential was returned.') _10 } _10 _10 const passkey = await extractPasskeyData(passkeyCredential) At this point, it's critical to securely store the information in thepasskey object in a persistent service. Losing access to this data will result in the user being unable to access their passkey and, therefore, their Safe Smart Account.

Get the provider and signer

Once the passkey is created, you can get theprovider andsigner , which is the externally-owned account of the user that was derived from its credentials.

viem You can instantiate the provider usingviem and the following imports:

_10 import { createWalletClient, http } from 'viem' _10 import { sepolia } from 'viem/chains' _10 const provider = createWalletClient({ _10 chain: sepolia, _10 transport: http('https://rpc.ankr.com/eth_sepolia') _10 }) _10 _10 const signer = passkey With theprovider andsigner you are ready to instantiate any of the kits from the Safe{Core} SDK and set up or use this signer as a Safe owner.

Recap and further reading

After following this guide, you are able to create a Safe signer using passkeys and get theprovider andsigner required to initialize the kits from the Safe{Core} SDK.

- Safe Passkeys contracts(opens in a new tab)

Magic Privy Was this page helpful?

Report issue