How to Integrate Pimlico with Passkey (WebAuthn) signer

This how-to guide will walk you through the steps to integrate Passkey (WebAuthn) signer with a smart account whose user operations are relayed and sponsored by Pimlico.

Passkey (WebAuthn) is a modern authentication method that allows users to sign in to websites and apps using their fingerprint, face, or other biometric information. For more information on how Passkey works, visitthe documentation page .

Install the required packages

```

npminstallviempermissionless

```

Create credentials

```

import{ useState }from"react" import{ createWebAuthnCredential, }from"viem/account-abstraction"

exportfunctionPasskeysDemo() { const[credential,setCredential]=useState(()=>
JSON.parse(localStorage.getItem("credential")) )

constcreateCredential=async()=>{ constcredential=awaitcreateWebAuthnCredential({ name:"Wallet" })
localStorage.setItem("credential",JSON.stringify(credential)) setCredential(credential) }

if(!credential) return( Create credential )

return(

Credential:{credential.id}

) }

```

We suggest using a paykeys server to share the same credential across multiple devices easily. To learn how to set up a paykeys server withsimplewebauthn , check out their guide. Zerodev also provides apaykeys server you can use.

Create Kernel Smart Account

```

import{ typeSmartAccountClient, createSmartAccountClient }from"permissionless" import{
typeToKernelSmartAccountReturnType, toKernelSmartAccount }from"permissionless/accounts" import{
entryPoint07Address, toWebAuthnAccount }from"viem/account-abstraction"

constpimlicoUrl=https://api.pimlico.io/v2{chain.id}/rpc?apikey={pimlicoApiKey} constpimlicoClient=createPimlicoClient({ chain,
transport:http(pimlicoUrl) })

exportfunctionPasskeysDemo() { const[smartAccountClient,setSmartAccountClient]= React.useState< SmartAccountClient<
Transport, Chain, ToKernelSmartAccountReturnType<"0.7">

() ... React.useEffect(()=>{ if(!credential)return toKernelSmartAccount({ client: publicClient, version:"0.3.1",
owners: [toWebAuthnAccount({ credential })], entryPoint: { address: entryPoint07Address, version:"0.7" }
}).then((account:ToKernelSmartAccountReturnType<"0.7">)=>{ setSmartAccountClient(
createSmartAccountClient({ account, paymaster: pimlicoClient, chain, userOperation: {
estimateFeesPerGas:async()=> (awaitpimlicoClient.getUserOperationGasPrice()) .fast },
bundlerTransport:http(pimlicoUrl) }) ) }) }, [credential]) ... }

```

Currently,permissionless supports passkeys only with kernel accounts.

Send a transaction

```
```

exportfunctionPasskeysDemo() { ... const[txHash,setTxHash]=React.useState()

constsendUserOperation=async(event: React.FormEvent)=>{ event.preventDefault() if(!smartAccountClient)return

constformData=newFormData(event.currentTarget) constto=formData.get("to")as0x{string}
constvalue=formData.get("value")asstring

consttxHash=awaitsmartAccountClient.sendTransaction({ calls: [{ to, value:parseEther(value) }], }) setTxHash(txHash) }

return(<>

Account

Address:{smartAccountClient?.account?.address}

Send User Operation

Send {txHash&&

Transaction Hash:{txHash}

}

) }

```

### Sign & verify a message

```

exportfunctionPasskeysDemo() { const[signature,setSignature]=React.useState()
const[isVerified,setIsVerified]=React.useState() ... constsignAndVerifyMessage=async()=>{ if(!smartAccountClient)return
constsignature=awaitsmartAccountClient.signTypedData(typedData)

constisVerified=awaitpublicClient.verifyTypedData({ ...typedData, address: smartAccountClient.account.address, signature })
setIsVerified(isVerified) setSignature(signature) } return(<>

Account

Address:{smartAccountClient?.account?.address}

Sign typed data

Sign typed data Test {signature&&(

Signature:

{signature}

)} {isVerified!==undefined&&(

Verified:{isVerified.toString()}

)}

) }

```
```