Building a Smart Passkey Wallet with AA

To accelerate the onboarding of the next billion to crypto.

Farhad Asgarov

Software Engineer, Clave

Section 1

Technical Background

Passkeys? Account Abstraction? Smart Wallet?

- What is a Passkey?
- What is Account Abstraction?
  - Different Verification Methods
  - Batch Transactions
  - Paymaster
  - Native AA
- Precompiled Contracts, RIP-7212

Today’s Goal

- Biometric authentication with Passkeys
- Simple transfer feature
- Gasless transactions with Paymaster
- Multicall to get multiple token balances
- Batch transfers

Section 2

Live Coding: Repository Setup

https://github.com/asgarovf/smart-wallet-starter

Section 3

Live Coding: Smart Contract Deployment

Contract Purposes

- BatchCaller: To send batch transactions
- Implementation: Main implementation of account
- Registry: To record the deployed user accounts
- GaslessPaymaster: To implement gasless transactions
- Proxy: To have an upgradable smart account
- PasskeyValidator: To validate Passkey signatures
- AccountFactory: To create (deploy) accounts

What Passkey Response Includes?

- Authenticator data: Contains information from the authenticator about the processing of a credential creation or authentication request

- Client data: Representing the client data that was passed to navigator.credentials.create()

- Signature: The assertion signature is created with the private key of the key pair that was created during the originating navigator.credentials.create() call and verified using the public key of that same key pair.

Building a P256 Message

```
function createMessage(
    bytes memory authenticatorData,
    bytes memory clientData
) private pure returns (bytes32 message) {
    bytes32 clientDataHash = sha256(clientData);
    message = sha256(bytes.concat(authenticatorData, clientDataHash));
}
```

Section 4

Live Coding: Building the Wallet

Thank you!

Farhad Asgarov

Software Engineer, Clave
farhad@clave.team
@asgarovf