uport

A technical overview of uPort

@oedbeo

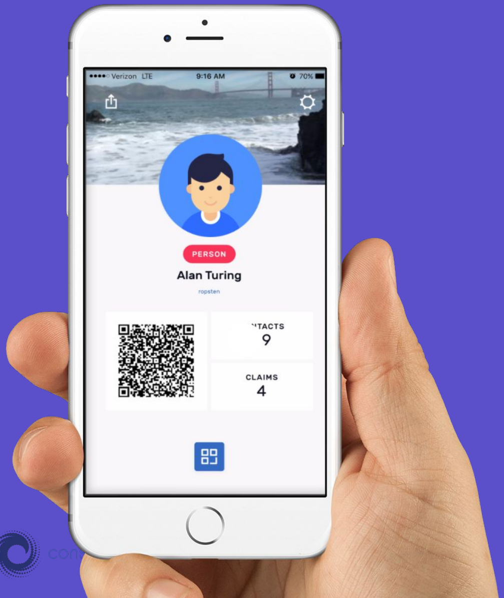
Building dapps for non-technical users is hard.

You'll need to consider the following:

- Account management and login?
- User friendly key management and recovery?
- Onboarding new customers without ether/Gas?
- Self-explaining transaction interface?
- Know your user/KYC?
- Transaction history? Reputation?
- … and more.

uPort is Ethereum’s user platform.

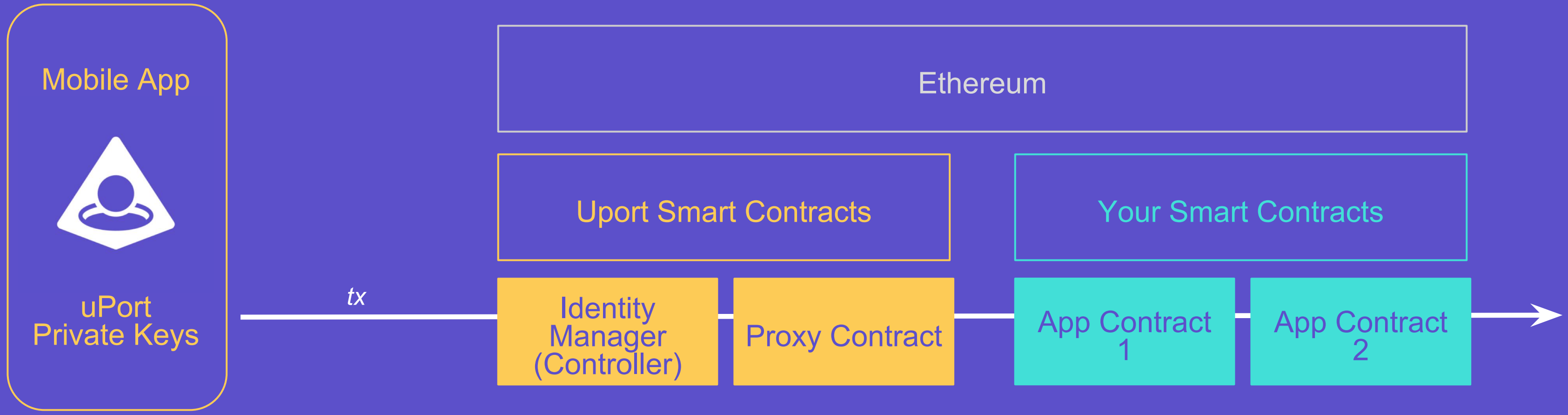
uPort is a self-sovereign identity, secure single sign-on, and digital signatures app – for use with both Ethereum and client/server apps.



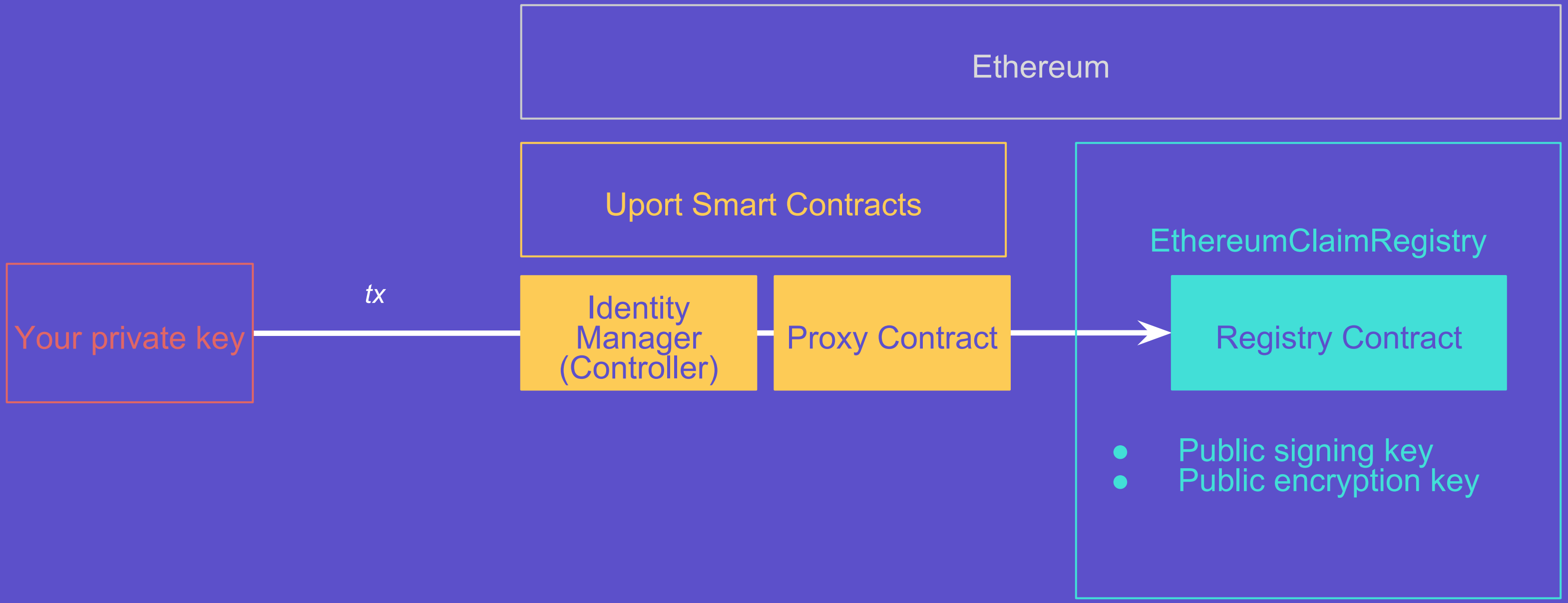
- Identity for Ethereum apps
- Passwordless, single sign-on
- Simple and secure key management
- User reputation system
- Browser-to-mobile app comms
- Identity verifications/KYC

Simple to integrate >

uPort uses a smart contract identity model.



A minimal PKI.



Ethereum

Uport Smart Contracts

Your private key

tx

Identity Manager (Controller)

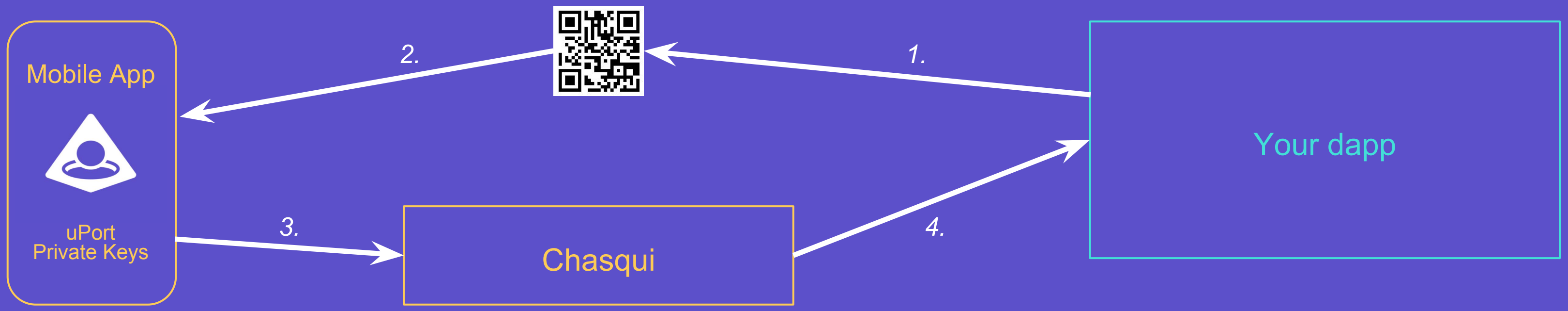
Proxy Contract

EthereumClaimRegistry

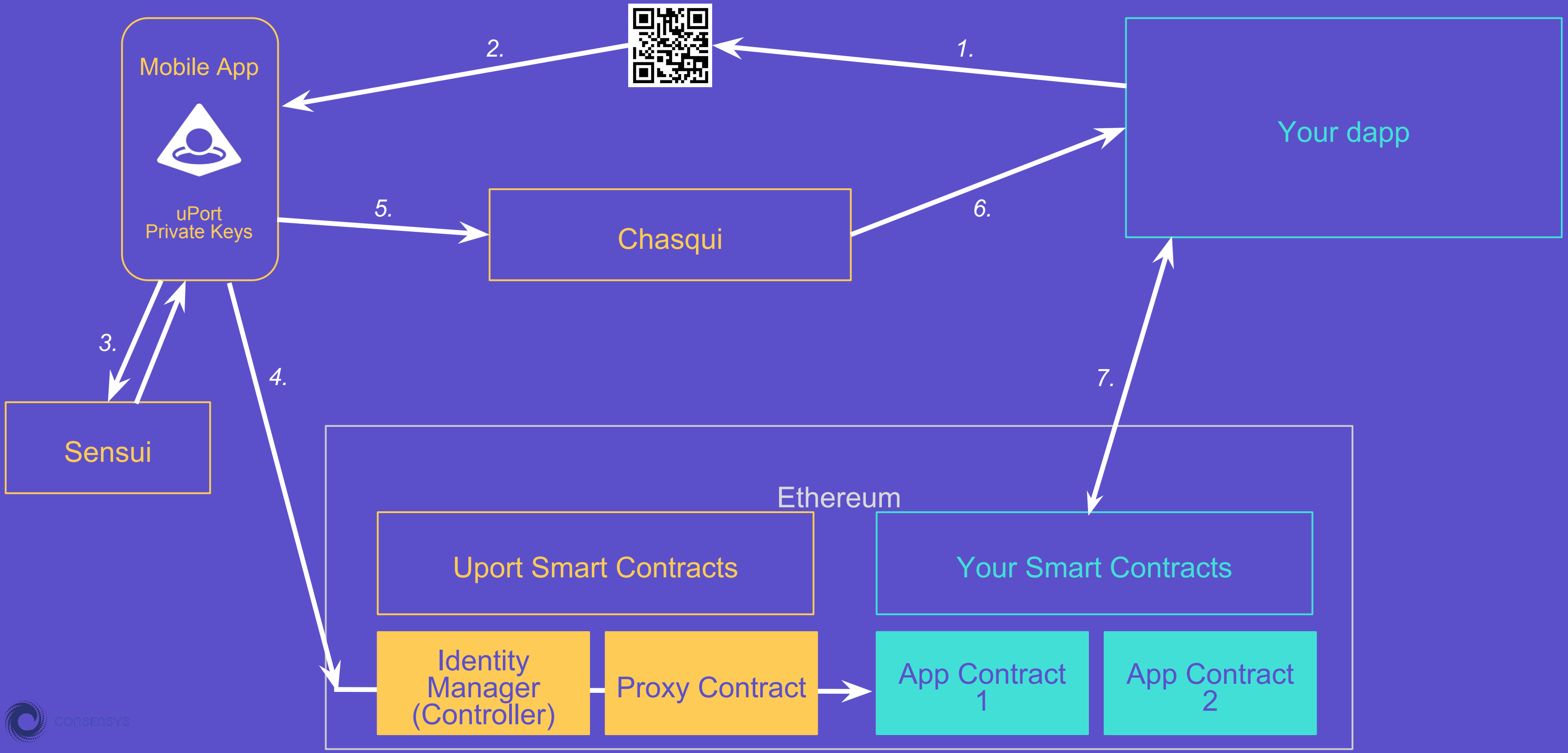
Registry Contract

- Public signing key
- Public encryption key

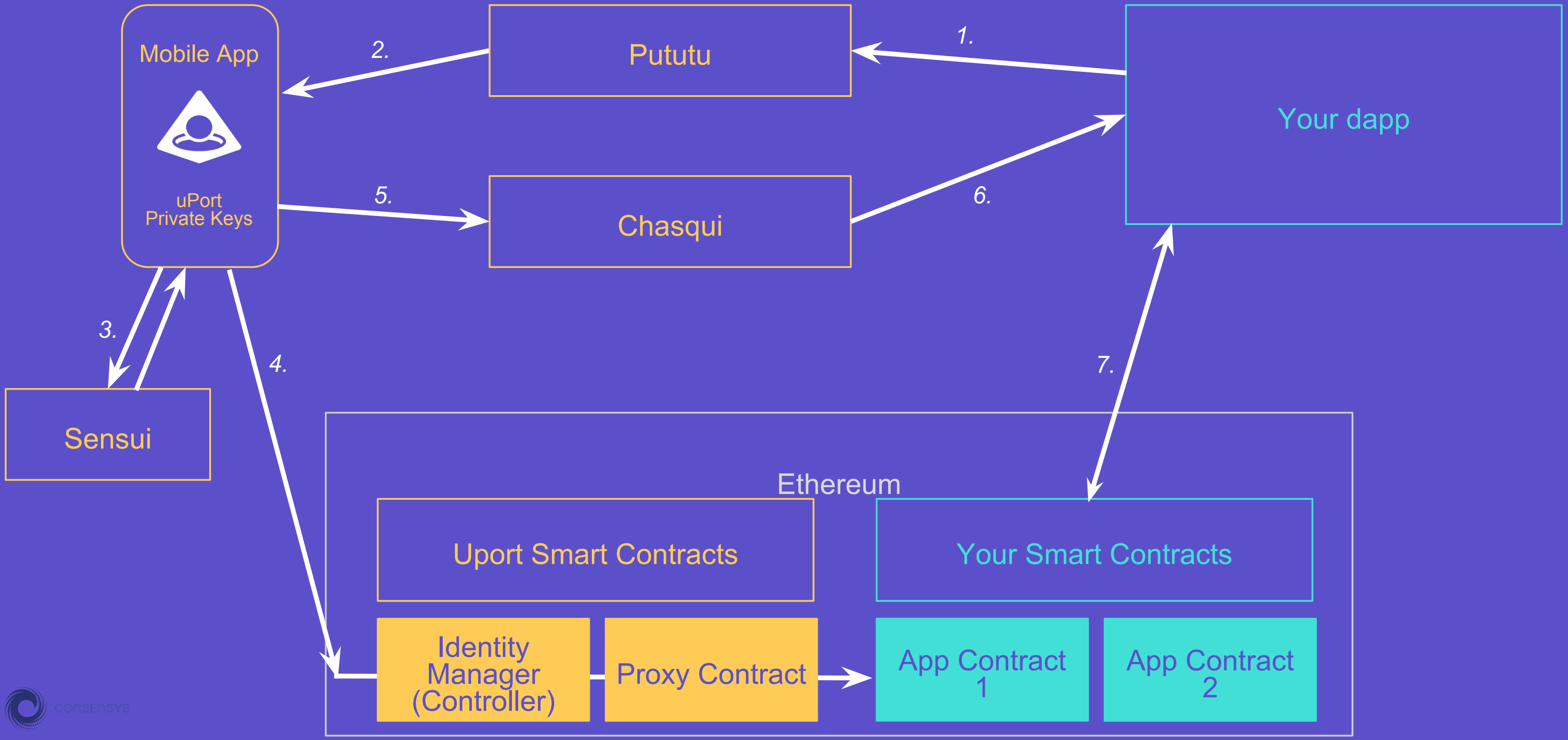
uPort Architecture.



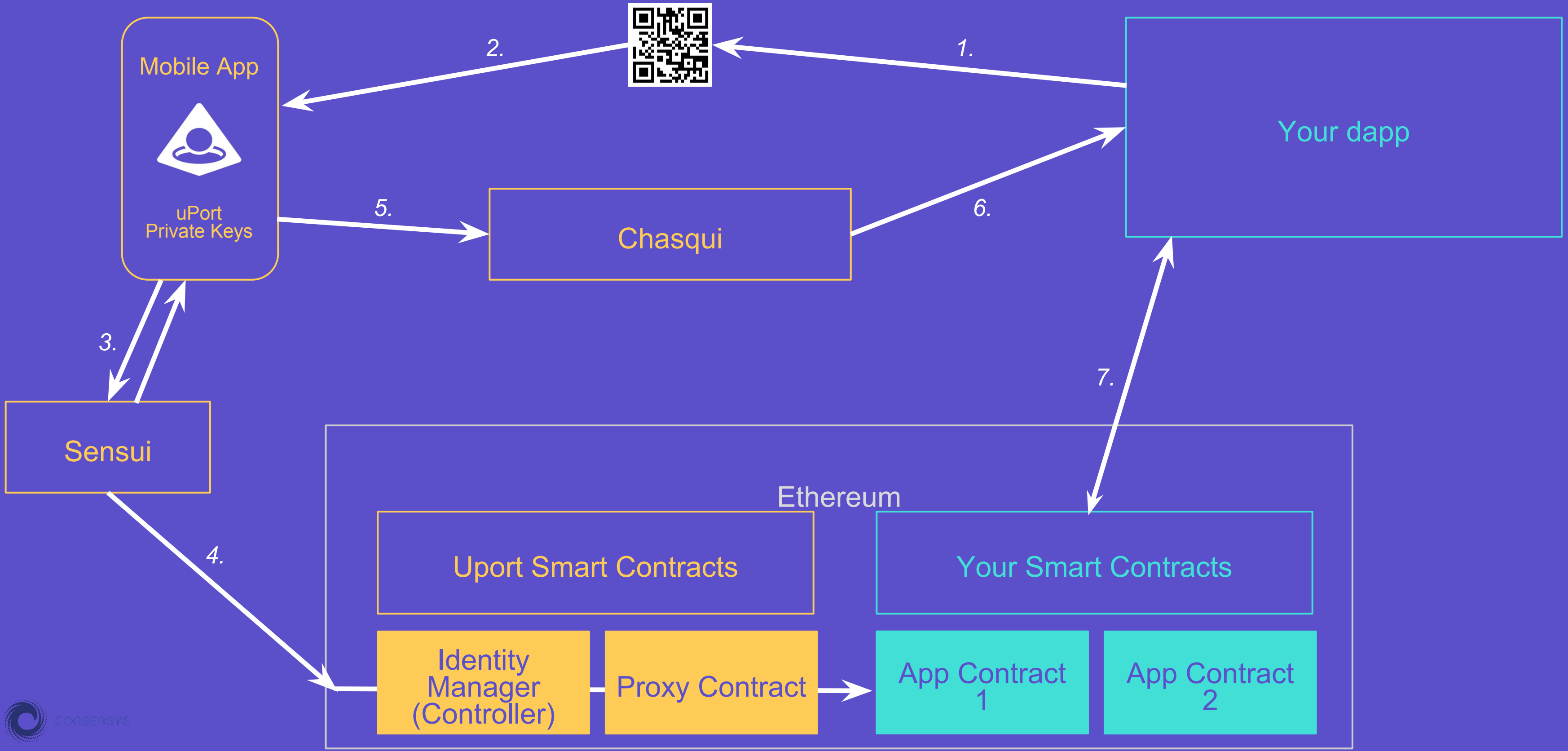
uPort Architecture.



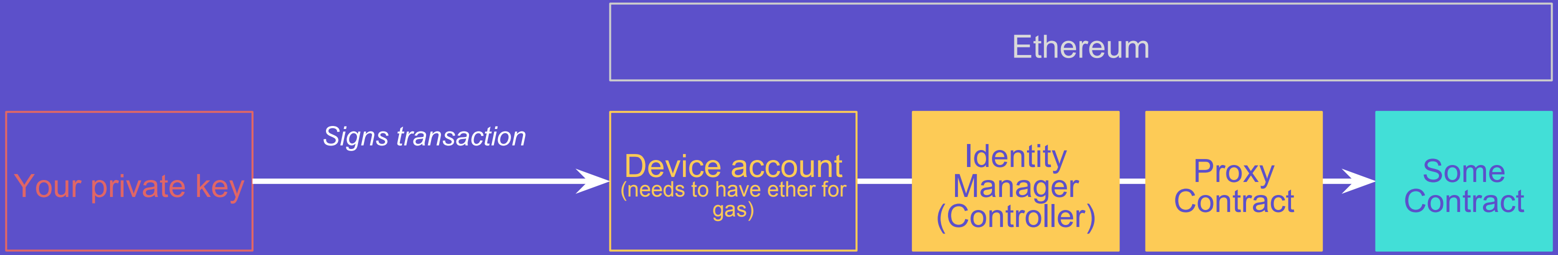
uPort Architecture.



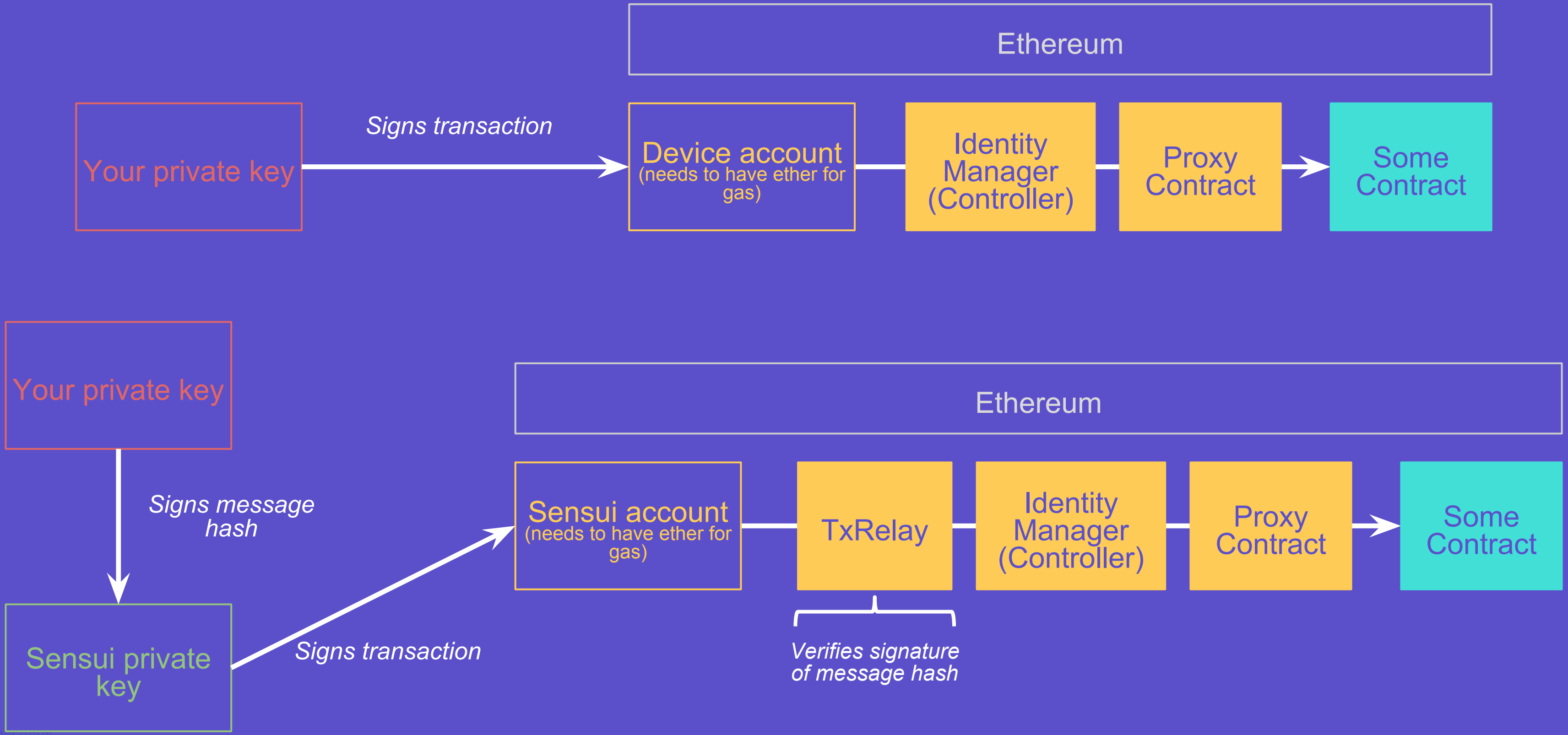
Meta Transactions.



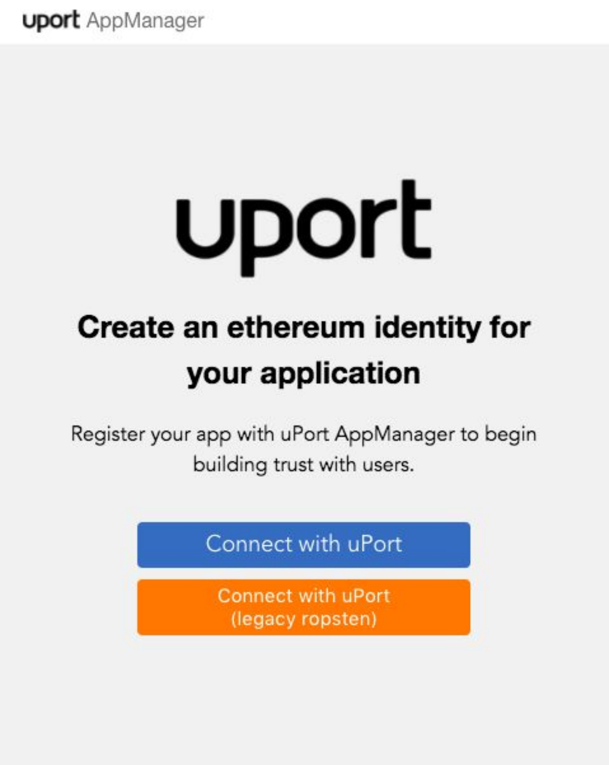
Meta Transactions.



Meta Transactions.



Developer Features



Create an ethereum identity for your application

Register your app with uPort AppManager to begin building trust with users.

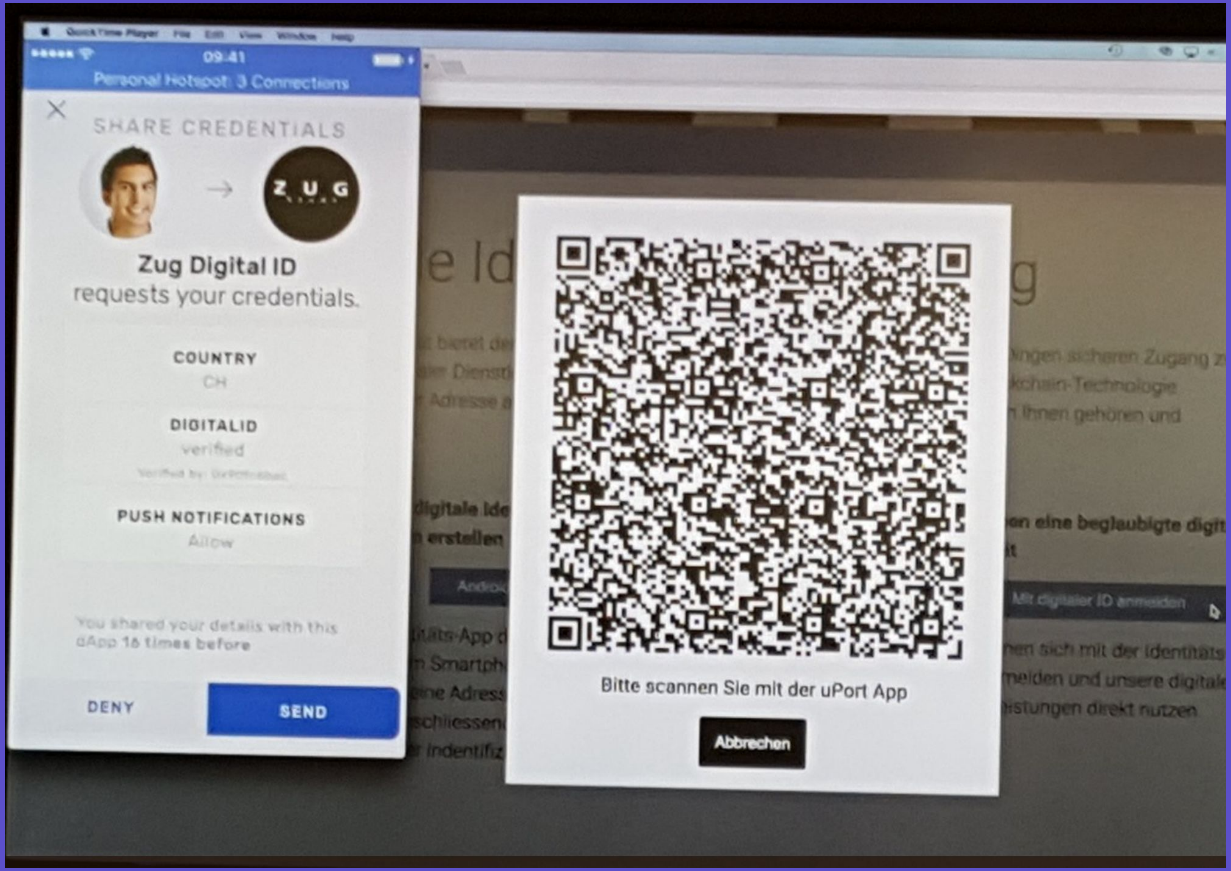
Scalable service infrastructure
We maintain your identity and transaction infrastructure so you can focus on your application.

Push Communication Protocol
Push protocol for transactions and data builds new interaction models with users, including asynchronous communication.

Transaction Card UI
Mobile transaction cards give users clarity and confidence in signing transactions.

Application Identity
Manage the identity of your application with our AppManager developer portal.

Developer Features


QuickTime Player File Edit View Window Help
09:41
Personal Hotspot: 3 Connections
Z U G

SHARE CREDENTIALS

Zug Digital ID
requests your credentials.

COUNTRY
CH

DIGITALID
verified

PUSH NOTIFICATIONS
Allow

You shared your details with this dApp 16 times before

DENY

SEND

Bitte scannen Sie mit der uPort App

Abbrechen

Transaction Card UI
Mobile transaction cards give users clarity and confidence in signing transactions.

Application Identity
Manage the identity of your application with our AppManager developer portal.

Integration demo.

Open Source Libraries

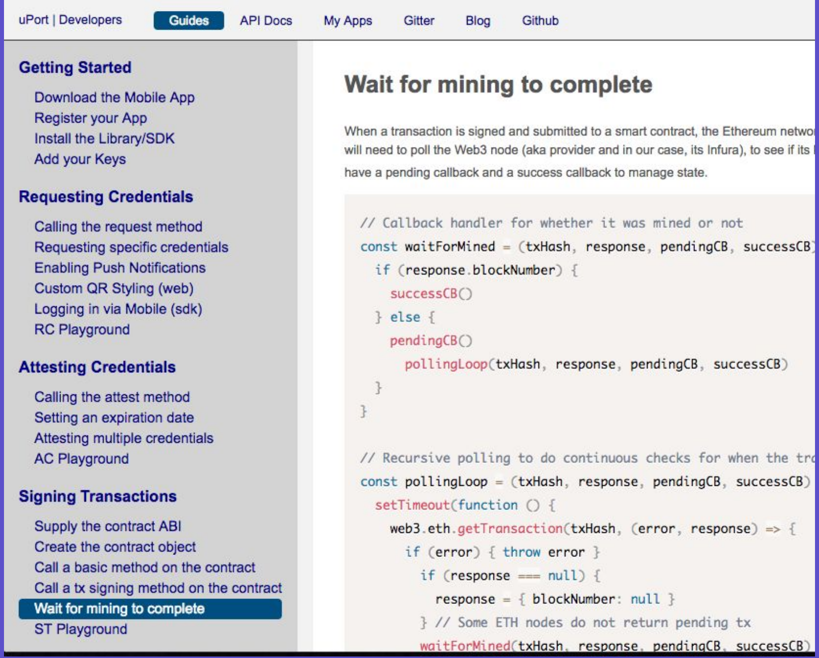
Connect is the client side library for integrating uPort so **users can communicate with your app**. It also allows for “sign in” to your app, issuing requests for user data, and sending transactions for them to sign.

Uport-js is a library used for **requesting private data from users, verifying signed tokens, and issuing attestations**. Users and services sign Json Web tokens (JWT) in order to share verifiable data privately. (**off-chain**)

Smart Contracts are open source and deployed - **uPort registry can be used to create on-chain attestations** or provide app-specific data.

Core protocol specs are open source and openly developed. The uport mobile app religiously follows these.

Documentation & Getting Started



Getting Started

- Download the Mobile App
- Register your App
- Install the Library/SDK
- Add your Keys

Requesting Credentials

- Calling the request method
- Requesting specific credentials
- Enabling Push Notifications
- Custom QR Styling (web)
- Logging in via Mobile (sdk)
- RC Playground

Attesting Credentials

- Calling the attest method
- Setting an expiration date
- Attesting multiple credentials
- AC Playground

Signing Transactions

- Supply the contract ABI
- Create the contract object
- Call a basic method on the contract
- Call a tx signing method on the contract
- Wait for mining to complete
- ST Playground

Wait for mining to complete

When a transaction is signed and submitted to a smart contract, the Ethereum netwo
will need to poll the Web3 node (aka provider and in our case, its Infura), to see if its
have a pending callback and a success callback to manage state.

```
// Callback handler for whether it was mined or not
const waitForMined = (txHash, response, pendingCB, successCB
  if (response.blockNumber) {
    successCB()
  } else {
    pendingCB()
      pollingLoop(txHash, response, pendingCB, successCB)
  }
}

// Recursive polling to do continuous checks for when the tr
const pollingLoop = (txHash, response, pendingCB, successCB)
  setTimeout(function () {
    web3.eth.getTransaction(txHash, (error, response) => {
      if (error) { throw error }
        if (response === null) {
          response = { blockNumber: null }
        } // Some ETH nodes do not return pending tx
        waitForMined(txHash, response, pendingCB, successCB)
```

All in one documentation site

- Learning guides & live embedded examples
- Library documentation
- App Manager (register your app)
- Developer chat (Gitter)
- Engineering blog
- Github organization.

developer.uport.me
(** The site is under active development)

Android	https://play.google.com/apps/testing/com.uportMobile
iOS	Apple AppStore

Thank you!

uport.me

consensys.net

oed@consensys.net