

OpenPassport

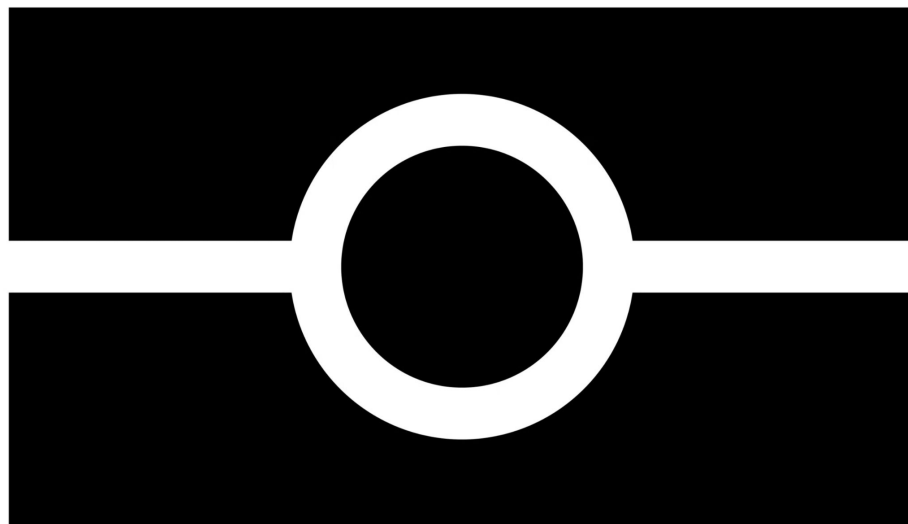
workshop

Rémi





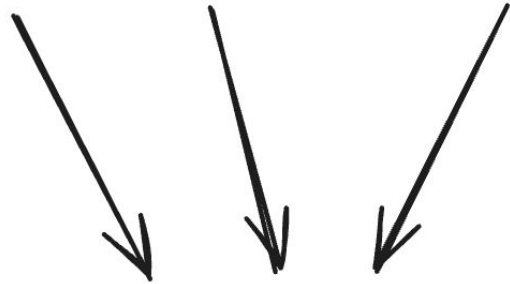




sybil resistance
selective disclosure
compliance

unforgeable

DG1 DG2 ... DG15

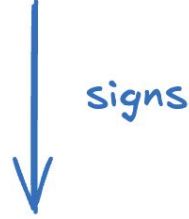


Hashed

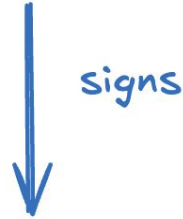


Signed

Country Signing Certificate Authority (CSCA)



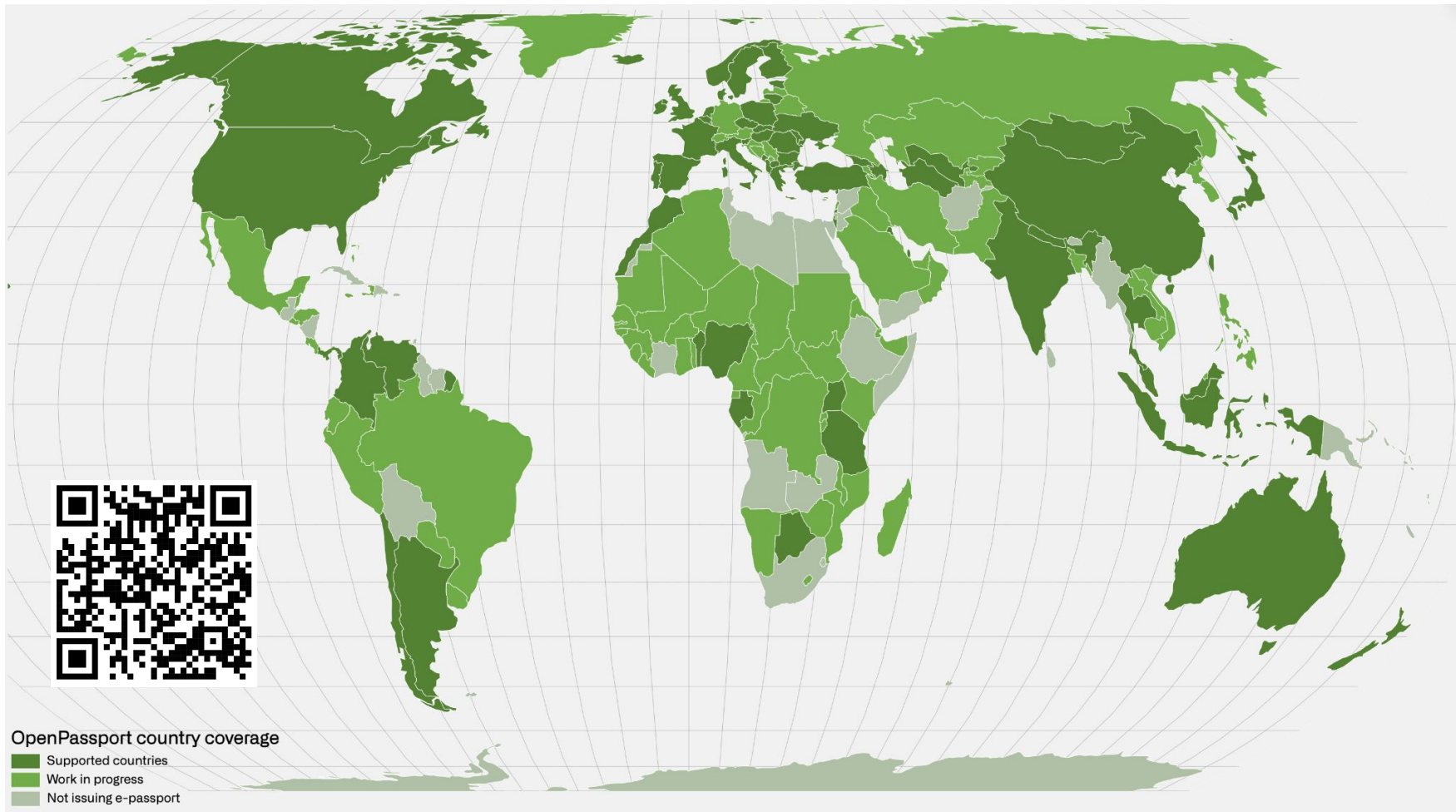
Document Signing Certificate (DSC)



ePassport

rsa
rsapss
secpr1
brainpool

sha1
sha256
sha384
sha512





demo

porn

User

DSC

Passport Proof


- nationality
- ofac check
- older than
- ...

nullifier

Application

Off-chain

Front-end



```
import { OpenPassportVerifier } from '@openpassport/core';

const scope = 'myExampleApp';
const openPassportVerifier: OpenPassportVerifier = new OpenPassportVerifier('prove_offchain', scope)
  .excludeCountries('Germany')
  .setMinimumAge(18);
```

age
nationality
OFAC

Front-end

```
import { OpenPassportQRcode } from '@openpassport/qrcode';

// ...

return (
  <div>
    <OpenPassportQRcode
      appName="Mock App"
      userId={userId}
      userIdType={'uuid'}
      openPassportVerifier={openPassportVerifier}
      onSuccess={(attestation) => {
        // send the code to the backend server
      }}
    />
  </div>
);
```


[Try out the Playground now!](#) ✕

OpenPassport playground



Name of the app

 Rainbow Chaser

Older than

18

Nationality



Check passport is not in OFAC list

Callback function


```
(appName, toast) => {  
  toast(`Congrats, you have been added to the group ${appName}`);  
} // triggered only if the proof is valid
```

WSS



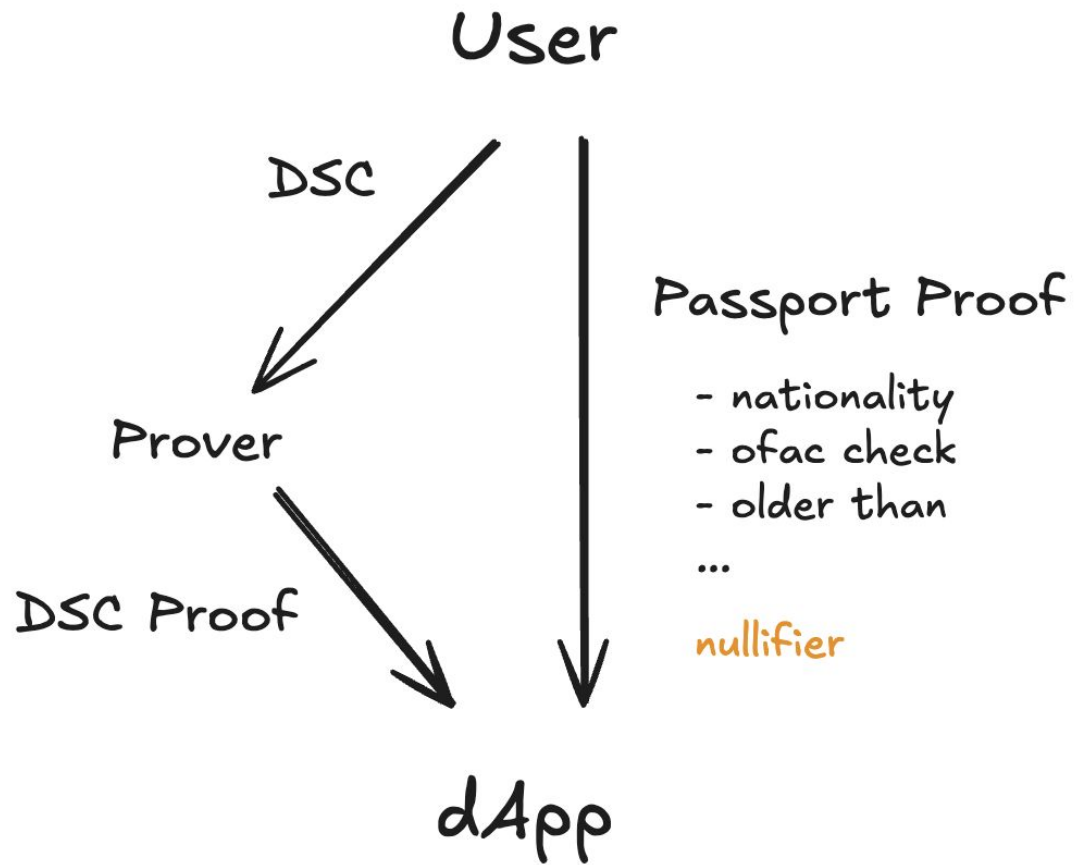
```
git clone https://github.com/zk-passport/websocket-server  
cd websocket-server  
  
yarn install  
yarn start
```

Back-end



```
import { OpenPassportVerifier, OpenPassportAttestation } from '@openpassport/core';  
// ...  
  
const openPassportVerifier = new OpenPassportVerifier('prove_offchain', 'myExampleApp');  
isValid = (await openPassportVerifier.verify(attestation as OpenPassportAttestation)).valid;
```

deFi



On-chain

Prover




```
pip install -r requirements.txt
```

```
./import_circuit.sh
```

```
modal deploy dsc_prover.py --name dsc_prover
```

Front-end



```
const openPassportVerifier: OpenPassportVerifier = new OpenPassportVerifier('prove_onchain', scope)
  .enableOFACCheck()
  .excludeCountries('Iran (Islamic Republic of)');
```

Front-end

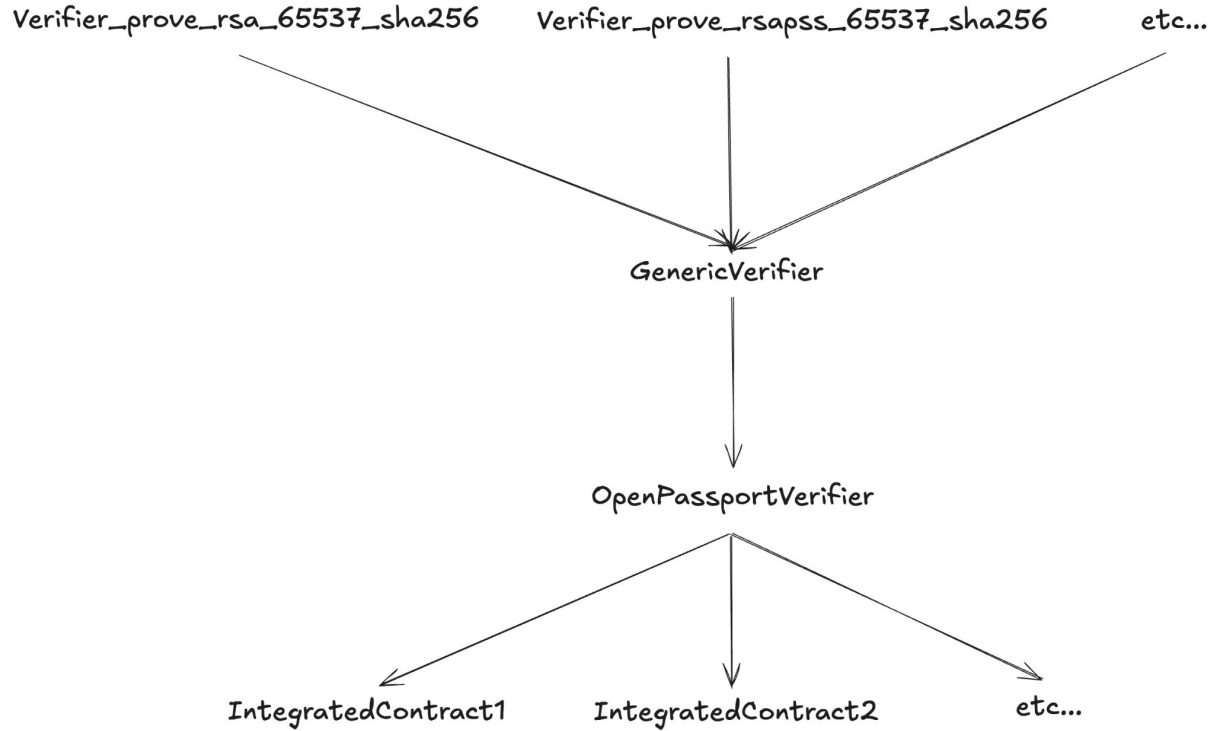
```
import { OpenPassportQRcode } from '@openpassport/qrcode';

// ...

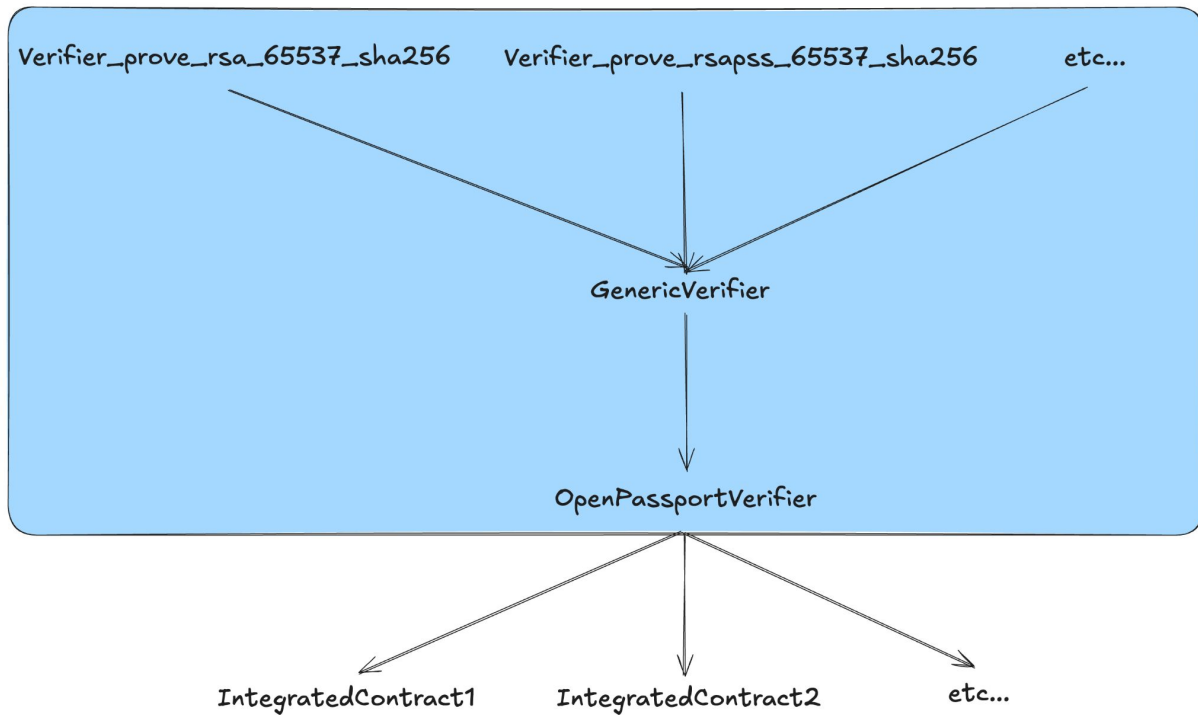
return (
  <div>
    <OpenPassportQRcode
      appName="Mock App"
      userId={userId}
      userIdType={'uuid'}
      openPassportVerifier={openPassportVerifier}
      onSuccess={(attestation) => {
        // send the code to the backend server
      }}
    />
  </div>
);
```


contract

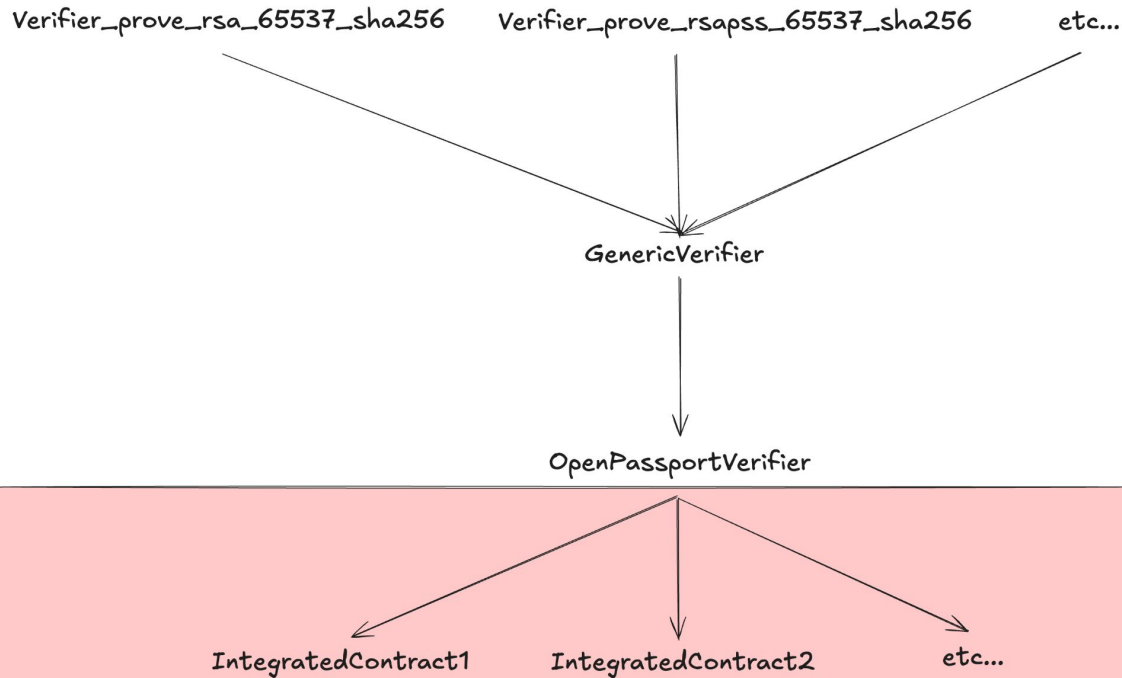
integration



Contract Integration



Contract Integration



Contract Integration

Contract Integration



```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.28;

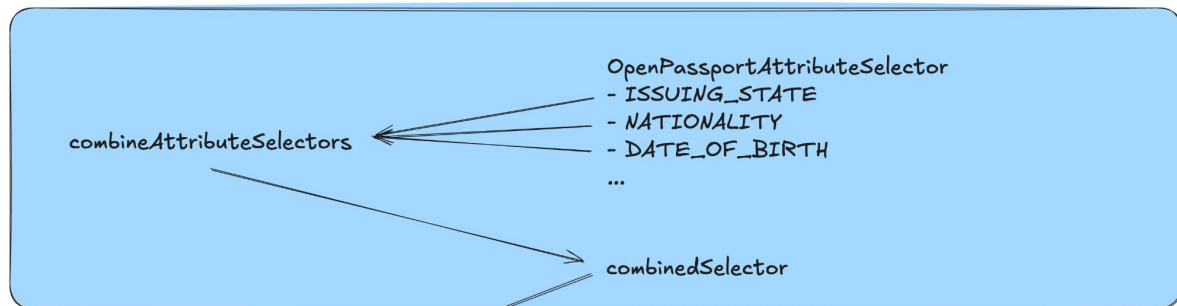
import {IOpenPassportVerifier} from "@openpassport-contracts/interfaces/IOpenPassportVerifier.sol";

contract ExampleContract {
    IOpenPassportVerifier public openPassportVerifier;
    constructor(
        address _openPassportVerifier,
    ) {
        openPassportVerifier =
            IOpenPassportVerifier(_openPassportVerifier);
    }
}
```

How to call



How to Call Contract



`verifyAndDiscloseAttributes`

`struct attestation`


```
{  
    uint256 proveVerifierId;  
    uint256 dscVerifierId;  
    IGenericVerifier.ProveCircuitProof pProof;  
    IGenericVerifier.DscCircuitProof dProof;  
}
```

`struct PassportAttributes {`

```
    string issuingState;  
    string name;  
    string passportNumber;  
    string nationality;  
    string dateOfBirth;  
    string gender;  
    string expiryDate;  
    uint256 olderThan;  
    bool ofacResult;  
    address pubkey;  
    bytes3[20] forbiddenCountries;  
}
```

How to Call Contract

How to Call Contract



```
uint256[] memory selectors = new uint256[](4);
uint256 index = 0;

selectors[index++] =
OpenPassportAttributeSelector.OLDER_THAN_SELECTOR;
selectors[index++] =
OpenPassportAttributeSelector.NATIONALITY_SELECTOR;
selectors[index++] =
OpenPassportAttributeSelector.OFAC_RESULT_SELECTOR;
uint256 combinedSelector =
OpenPassportAttributeSelector.combineAttributeSelectors(selectors);
```



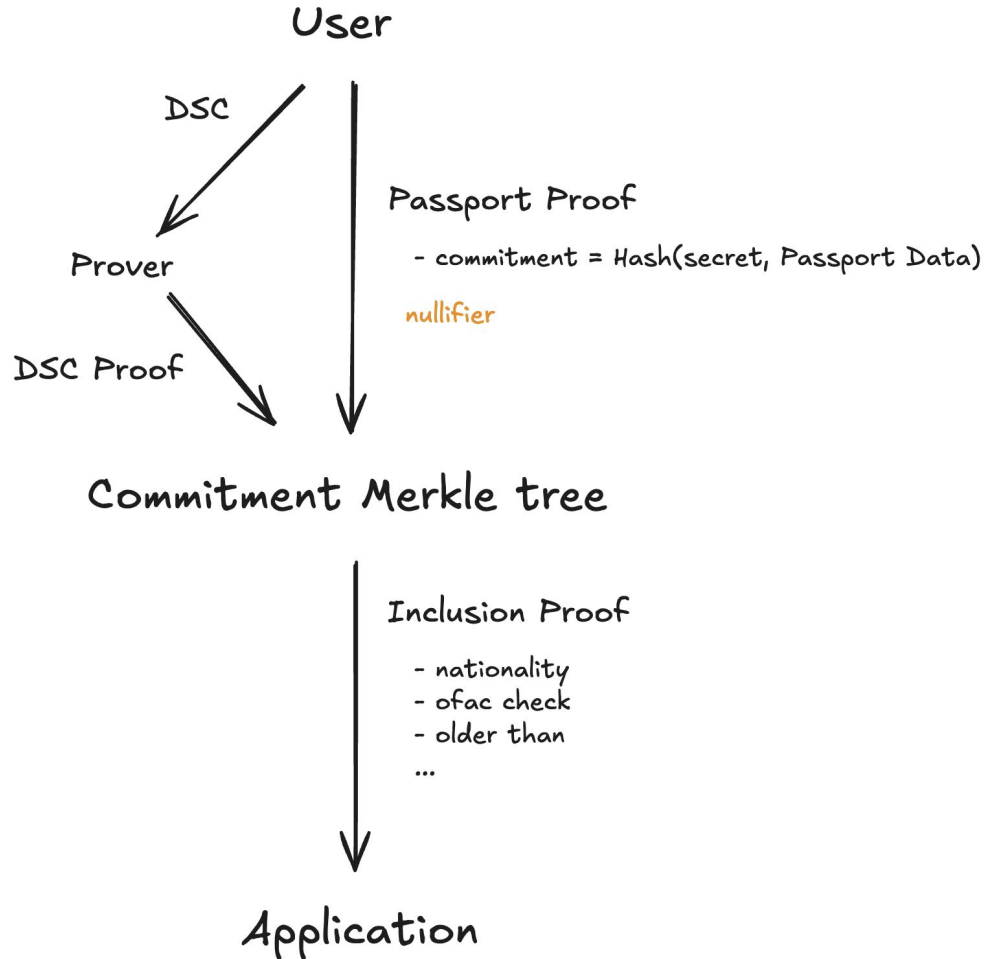
How to Call Contract

How to Call Contract



```
uint256 combinedSelector =  
OpenPassportAttributeSelector.combineAttributeSelectors(selectors);  
  
IOpenPassportVerifier.PassportAttributes memory passportAttributes =  
openPassportVerifier.verifyAndDiscloseAttributes(  
    attestation,  
    combinedSelector  
);  
return passportAttributes.nationality
```

vote



Register

Merkle tree reader

github.com/zk-passport/merkle-tree-reader

Cursive exhibition



Thank you!