D. I System

Decentralized Identity Verification Platform By Police

Documentation

Video Demonstration

➢Client

This project assumes that each portal has a unique view and access to the network with capabilities limited to their functionality.Hence this project has 3 different client views,each of which can be accessed via its individual URL as follows:

◆ User

http://localhost:3000/user

In this client,the transactions are done using a private key. Here it is not hard-coded with the client,in a working environment it should be given externally from a read only device.

It should include Name,Email,Date of birth,Address,Phone Number,Pin-code and Encryption Key are given as the inputs. User has an option to change encryption key and view data from the interface

If the user is verified then user can only edit the Data after having a limited time and reapply for verification.

◆ Police

http://localhost:3000/police

In this client,the transactions are done using a private key.Here it is hard-coded with the client,the key is secured by public key matching method.

In this part inside police UI we can see the transaction details i,e user details send by user.Only details allowed by user to share to police should be able to display,all other details like voter id no,aadhar no and other kinds of details are encrypted.Mainly the data are fetched from state to display.If user shares decryption key,police should be able to view all details using decryption key.

Here police can accept or reject user data . If rejection occurs users are not permitted to get verified public key and they cant even login to client part.Here rejection means that the user data is deleted from state.Accept data pass the user details to the validator and should be added to block .If police accept user should get a public key and they can login into client portal and check their data using this public key and decryption key so user should able to share public key in case of identification purposes for various activities.

◆ User Client

http://localhost:3000/client

It doesn't make any transaction.Only a verified user can log to the user client.In there the client can check the users by their public key.

➢Transaction Processors (TP)

● KnowYourCustomer 1.0

This TP provides 3 business logics in the chain.

◆ Add User Data

addUserData() function provides to create a record of user KYC details.
The recorded Data contains Encrypted KYC Data and Mini Data.

◆ Change Encryption Key

changeEncKey() function provides to update the record to corresponding state address.
The new recorded Data contains New Encrypted KYC Data and Mini Data.

◆ Verify User

verifyUser() function provides to update the record to corresponding state address.
In the new record,The Mini Data update with Verify Tag.

Code Flow

◆ CLIENT

● User client

On button click,

• The values after validation are read from the fields which should be passed in through
the createTransaction() function.

• The payload is set with the particular action, some encrypted personal details and some
other details which are encoded.
• The Address is set for user according to the publickey & combination of family name .
• Then the address is sent to TranscationHeader which includes the family name, family
version , batcherPublickey ,signer public key, Inputs, Outputs ,Dependencies ,Nonce and
Payloadsha512 string.
• Then the Transaction is signed with Transaction headerbyte.
• The batch which includes transactions is again passed to batchListBytes .Then it is sent
into the RestAPI using the function sendTransaction() and post data.

On 'Change Encryption key button' click,

The values from the fields which should be passed to the changeUserKey() function and arguments are newkey,oldkey and private key of user private key should be passed to getUserPublickey() function in-order to generate a public key and this public key is again passed to getUserAddress() function to get the user address.
User address should be passed to getState() function which returns data that is fetched from state in json format.
Decode the fetched data and separate payload data's
Decrypt the encrypted payload with old key and stores in a variable.Then again encrypt this variable with new key and pass the new payload to createTransaction() function.

The Address is set for user according to the publickey
Then the address is sent to TranscationHeader which includes the
familyname,
familyversion,
batcherPublickey,
signerpublickey,
Inputs,

Outputs,
Dependencies,
Nonce,
Payloadsha512 string.
Then the Transaction is signed with Transaction headerbyte.
The batch which includes transactions is again passed to batchListBytes.Then it is
sent into the RestAPI using the function sendTransaction() and post data.

- Police client

On button click there are 2 options Accept & Reject ,
• When accept or reject button was clicked action , userpublickey , private key has been passed to a function verifyUser()
• After checking the private key of police , creating a payload which contain action ,
userpublickey and submit to createTransaction() function
• where address is generated from user publickey
• Then the address is sent to TranscationHeader which includes the
family name,
familyversion,
batcherPublickey,
signerpublickey,
Inputs,
Outputs,
Dependencies,Nonce,
Payloadsha512 string.
Then the Transaction is signed with Transaction headerbyte.
The batch which includes transactions is again passed to batchListBytes.Then it is
sent into the RestAPI using the function sendTransaction() and post data.
Deleting userdata and Submiting user data’s by police should be worked on the basis

of TP.

- Client

When log in to client using a private key , this key should be passed to
getUserPublickey() function inorder to generate a public key and this public key is
again passed to getUserAddress() function to get the user address
Then using the user address fetch data from state
This fetched data is again converted to string using buffer class in js
if the private key generate an address that donot exist in state , the login page should
be displayed an alert otherwise it routes to UI were we can submit user publickey and
decryption key to check the data .
When ‘check user’ button clicks the key and decryption key should be passed to
onClick function clientData()
Then fetching data from state using the key has been passed and decrypt it and
displayed on the UI

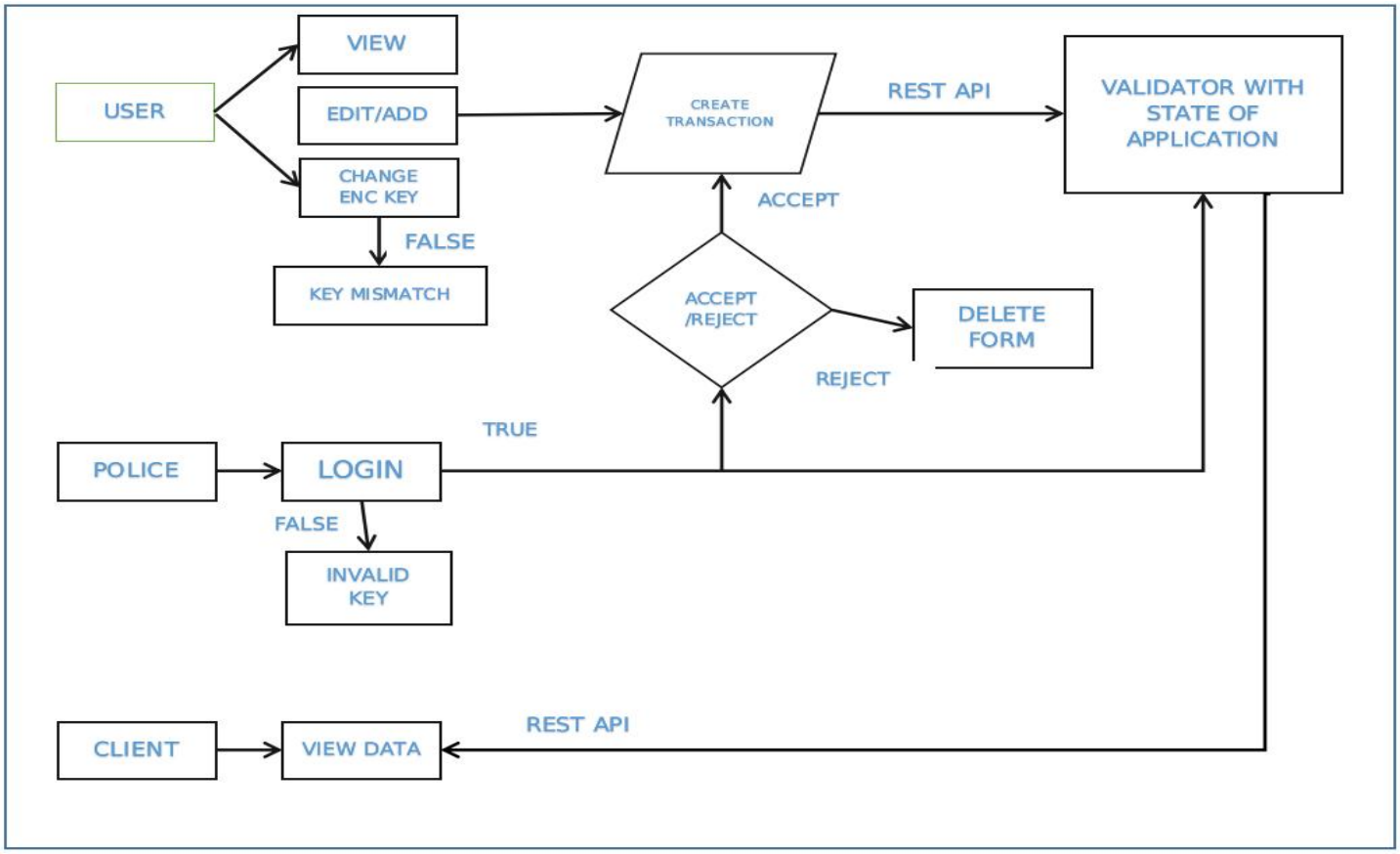
◆ TP

- KnowYourCustomer

Inside apply (),
We decode payload from transactionProcessRequest
transactionProcessRequest which holds valid transaction send by client.There are mainly 4 types of action includes inside payload ie `-1,0,1,2.`
if `action == -1`,
calls the addUserdata() function and pass args as `context,userPublickey,Payload[1]` were addUserdata() generates address using public key and pass this payload

to writeToState() function inside writeToState() , the address provided in dictionary be set in validator state to its corresponding value.

if `action == 0`,
pass the address and context as an arguments to deleteFromState() function which is used to unset the provided address from the validator state.
if `action == 1`,
pass args as `context,Payload[1]` which is userPublickey and Payload[0] which is action to verifyUser() .Then generates address using userpublickey and decode data from the address & create a newpayload with encrypted data and time and send to writeTostate() function to set a new value to the block with help of validator.
if `action == 2`,
this occurs when there is a request to change encryption key from user side .Payload and address should be passed to writeToState() function were the address provided in dictionary be set in validator state to its corresponding value.



Commonly Used Functions

- `getUserPublicKey()`

used to generate public key from a passed private key.

- `getUserData()`

it generates address with publickey & hash of family name.

- `getState()`

used to return data from fetched state.

- `getUserAddress()`

used to generate address with public key and hash of family name.

- `encrypt()`

used to encrypt the payload using a encryption key.

- `decrypt()`

used to decrypt the payload using a decryption key.