**Installation**

* Install Ganache and Truffle suite.

<https://archive.trufflesuite.com/ganache/>

* Install the requirements: pip install -r requirements.txt

**Execution**

**The code consists of two bash scripts “bash.sh” and “bash2.sh” to automatically run the complete code. The step by step procedure to run the code without bash scripts are as follows:**

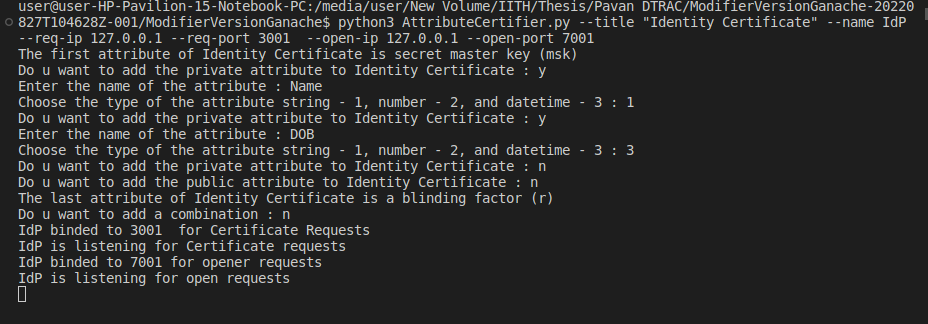
**Certificate Setup by Certifier:**

1. Create ROOT directory and update the path to root directory by updating the variable “root\_dir” in every file.

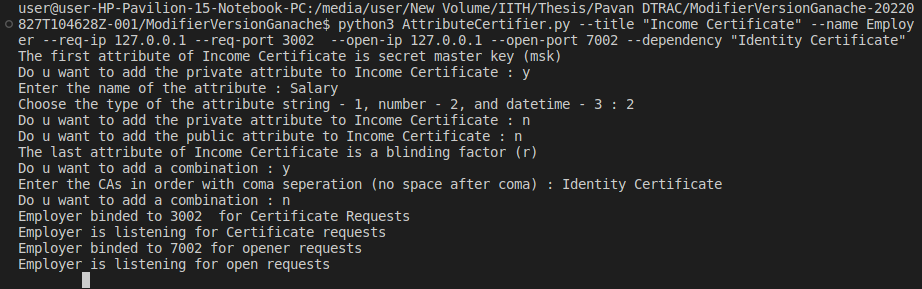


1. Run AttributeCertifier.py (Commands are at line 16 and 17)
   1. We are creating 2 types of Certifiers here.
   2. Choose public and private attributes for both commands 16/17
   3. If both 16 and 17 need to be combined, give combination as “y” and name as 16’s (eg: Identity Certificate)
      1. Combination is the other Vcert the one being created depends on.
      2. It is given for 2nd Vcert

Terminal 1:



Terminal 2:



**Credential Setup by Admin:**

1. Open Ganache and make address changes. In Ganache there are total 10 account addresses we get for a project.

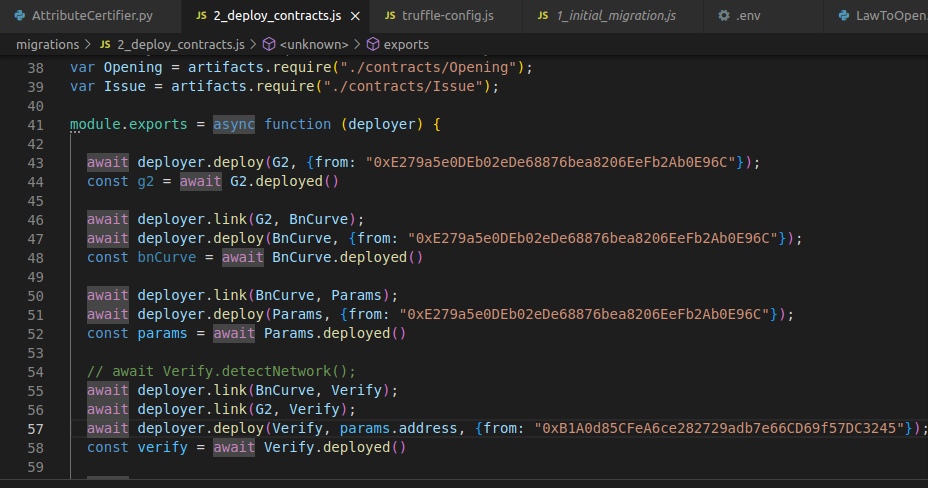
Address 1of Ganache is for Admin/Protocol Initiator. Change the “from:….” Addresses of /migrations/2\_deploy\_contracts.js: Lines 43/47/51/62/65/68 with this address

Addresses 2-4 for Validators

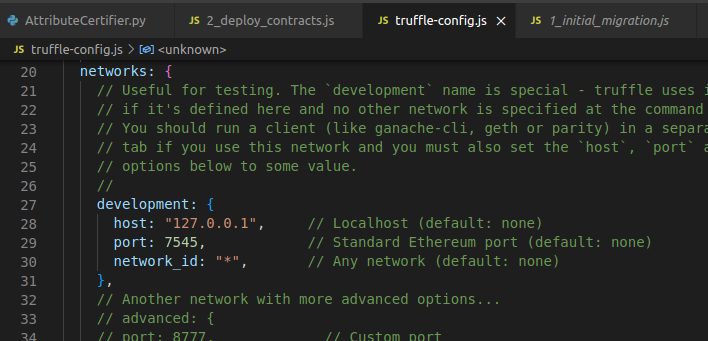
Addresses 5-7 for Openers

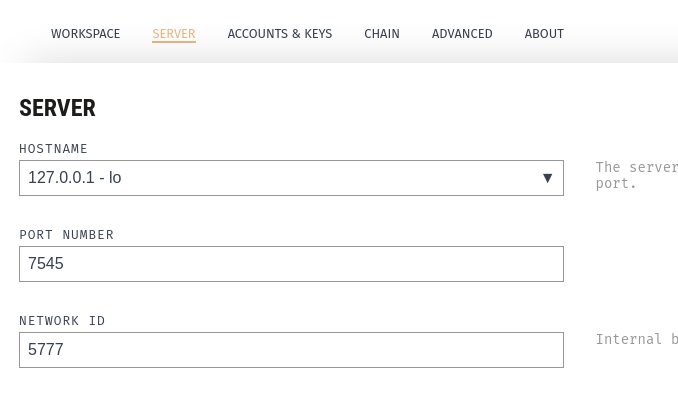
Address 8 for Service Provider. Change the “from:….” Address in /migrations/2\_deploy\_contracts.js: Line 57 (as verifier) with this address.

Addresses 9-10 for Users



1. Make sure truffle-config.js has same development setup as given in Ganache





1. Create a .env file in same location
   1. **Install hdwallet-provider:** npm install @truffle/hdwallet-provider
   2. **Install dotenv:**  npm install dotenv
   3. Take MNEMONIC from Ganache and add in .env
   4. Make modifications to Solidcore version (SOLC\_VERSION) in .env file to what is provided in truffle-config.js file
   5. You may change the Addresses that follow, but skipping it for the time being.
   6. NOTE:
      1. Do these installations at same location as the API
      2. The IP and Port in config should be same as that in Ganache
2. Deploy the Smart Contract: truffle migrate --reset –compile-all

Compiling your contracts...

===========================

> Compiling ./contracts/Issue.sol

> Compiling ./contracts/Migrations.sol

> Compiling ./contracts/Opening.sol

> Compiling ./contracts/Params.sol

> Compiling ./contracts/Request.sol

> Compiling ./contracts/Verify.sol

> Compiling ./libraries/BN256G2.sol

> Compiling ./libraries/G.sol

> Compilation warnings encountered:

Warning: Unnamed return variable can remain unassigned. Add an explicit return with value to all non-reverting code paths or name the variable.

--> project:/libraries/G.sol:76:32:

|

76 | internal view returns (G1Point memory)

| ^^^^^^^^^^^^^^

> Artifacts written to /media/user/New Volume/IITH/Thesis/Pavan DTRAC/ModifierVersionGanache-20220827T104628Z-001/ModifierVersionGanache/build/contracts

> Compiled successfully using:

- solc: 0.8.13+commit.abaa5c0e.Emscripten.clang

Starting migrations...

======================

> Network name: 'development'

> Network id: 5777

> Block gas limit: 20000000000 (0x4a817c800)

1\_initial\_migration.js

======================

Replacing 'Migrations'

----------------------

> transaction hash: 0xc9f55f1faa618c78dc49286dc4b3f579b4cbc4a3100d0c09ff8ceaf6a2b18b54

> Blocks: 0 Seconds: 0

> contract address: 0x14401B5DeC515E09f68cAB59b2fBaaC91448acBB

> block number: 22

> block timestamp: 1662652557

> account: 0xE279a5e0DEb02eDe68876bea8206EeFb2Ab0E96C

> balance: 99.999999979592397

> gas used: 204952 (0x32098)

> gas price: 0.000001 gwei

> value sent: 0 ETH

> total cost: 0.000000000204952 ETH

> Saving migration to chain.

> Saving artifacts

-------------------------------------

> Total cost: 0.000000000204952 ETH

2\_deploy\_contracts.js

=====================

Replacing 'BN256G2'

-------------------

> transaction hash: 0xada6e947a16941fc3bdcaed05b2a162aa8d350fe80545eb990ff9f83fb571caf

> Blocks: 0 Seconds: 0

> contract address: 0x6A73B51CD3F4c99bB1945561C0b6b48f4E5933dF

> block number: 24

> block timestamp: 1662652560

> account: 0xE279a5e0DEb02eDe68876bea8206EeFb2Ab0E96C

> balance: 99.999999978960054

> gas used: 590031 (0x900cf)

> gas price: 0.000001 gwei

> value sent: 0 ETH

> total cost: 0.000000000590031 ETH

Replacing 'G'

-------------

> transaction hash: 0x23b2068f8b39f191f337edc62bbb411fb73b7dffc63b2bfee69f78bd6a585120

> Blocks: 0 Seconds: 0

> contract address: 0x7c203De106D9C3a7fC5B29F4502Db8dfd68911D0

> block number: 25

> block timestamp: 1662652561

> account: 0xE279a5e0DEb02eDe68876bea8206EeFb2Ab0E96C

> balance: 99.999999978887837

> gas used: 72217 (0x11a19)

> gas price: 0.000001 gwei

> value sent: 0 ETH

> total cost: 0.000000000072217 ETH

Replacing 'Params'

------------------

> transaction hash: 0x6f5baf9736aaf43799a70e908ed17048c777a68548c755160c63a582aaf0cd7b

> Blocks: 0 Seconds: 0

> contract address: 0x8F77352961c0824005abd9859Bc44b1f131657B1

> block number: 26

> block timestamp: 1662652562

> account: 0xE279a5e0DEb02eDe68876bea8206EeFb2Ab0E96C

> balance: 99.999999977006013

> gas used: 1881824 (0x1cb6e0)

> gas price: 0.000001 gwei

> value sent: 0 ETH

> total cost: 0.000000001881824 ETH

Linking

-------

\* Contract: Verify <--> Library: G (at address: 0x7c203De106D9C3a7fC5B29F4502Db8dfd68911D0)

Linking

-------

\* Contract: Verify <--> Library: BN256G2 (at address: 0x6A73B51CD3F4c99bB1945561C0b6b48f4E5933dF)

Replacing 'Verify'

------------------

> transaction hash: 0x5b34de317ea8896d68a17ccfba0fa4e4d07e5d946c33958f07968c682fea9b81

> Blocks: 0 Seconds: 0

> contract address: 0x116aD166cbf3ac8692F32b792DAA3FB8e567F2Ae

> block number: 27

> block timestamp: 1662652564

> account: 0xB1A0d85CFeA6ce282729adb7e66CD69f57DC3245

> balance: 99.999999994328562

> gas used: 2835725 (0x2b450d)

> gas price: 0.000001 gwei

> value sent: 0 ETH

> total cost: 0.000000002835725 ETH

Linking

-------

\* Contract: Request <--> Library: G (at address: 0x7c203De106D9C3a7fC5B29F4502Db8dfd68911D0)

Linking

-------

\* Contract: Request <--> Library: BN256G2 (at address: 0x6A73B51CD3F4c99bB1945561C0b6b48f4E5933dF)

Replacing 'Request'

-------------------

> transaction hash: 0x39bc7df57fef49483751e6a4a69fd288e3bc6848ae4cf11ed11dcaf99fedf19e

> Blocks: 0 Seconds: 0

> contract address: 0x0891D09A3140616cE5c9E0BcC93311c6bda8d4e2

> block number: 28

> block timestamp: 1662652565

> account: 0xE279a5e0DEb02eDe68876bea8206EeFb2Ab0E96C

> balance: 99.99999997264464

> gas used: 4361373 (0x428c9d)

> gas price: 0.000001 gwei

> value sent: 0 ETH

> total cost: 0.000000004361373 ETH

Replacing 'Issue'

-----------------

> transaction hash: 0x22a84e2f567d696d74526a1fe044b7c7eec2eb2997e23400f9766d846c4c41d2

> Blocks: 0 Seconds: 0

> contract address: 0x27F7003078b34Fb29Fca88381f48EC07bf06936c

> block number: 29

> block timestamp: 1662652567

> account: 0xE279a5e0DEb02eDe68876bea8206EeFb2Ab0E96C

> balance: 99.999999972262445

> gas used: 382195 (0x5d4f3)

> gas price: 0.000001 gwei

> value sent: 0 ETH

> total cost: 0.000000000382195 ETH

Replacing 'Opening'

-------------------

> transaction hash: 0x34591286fe90721566d94f546e67f7484fda01ee7556b090d3e52759c8133961

> Blocks: 0 Seconds: 0

> contract address: 0xfABd5118B5c118a2Cf833B92937894004f5B9EB2

> block number: 30

> block timestamp: 1662652568

> account: 0xE279a5e0DEb02eDe68876bea8206EeFb2Ab0E96C

> balance: 99.999999971850021

> gas used: 412424 (0x64b08)

> gas price: 0.000001 gwei

> value sent: 0 ETH

> total cost: 0.000000000412424 ETH

0xfABd5118B5c118a2Cf833B92937894004f5B9EB2

0x27F7003078b34Fb29Fca88381f48EC07bf06936c

Smart Contract addresses

0x0891D09A3140616cE5c9E0BcC93311c6bda8d4e2. 0x8F77352961c0824005abd9859Bc44b1f131657B1

0x116aD166cbf3ac8692F32b792DAA3FB8e567F2Ae

> Saving migration to chain.

> Saving artifacts

-------------------------------------

> Total cost: 0.000000010535789 ETH

Summary

=======

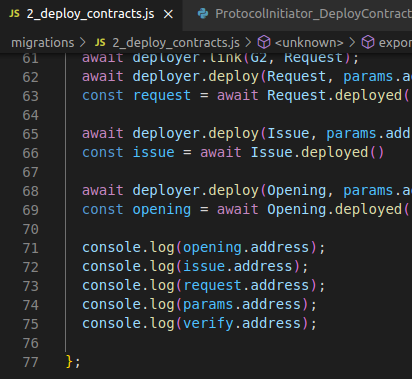
> Total deployments: 8

> Final cost: 0.000000010740741 ETH

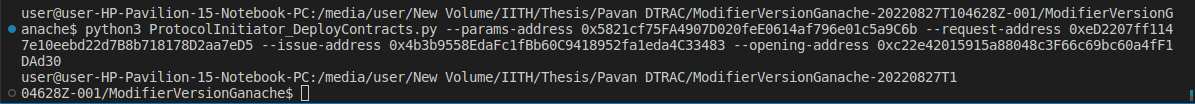
Smart Contracts (SC) are deployed now.

NOTE: Warnings can be ignored

1. The last 5 addresses are of following smart contracts:
   1. Opening
   2. Issue
   3. Request
   4. Params
   5. Verify

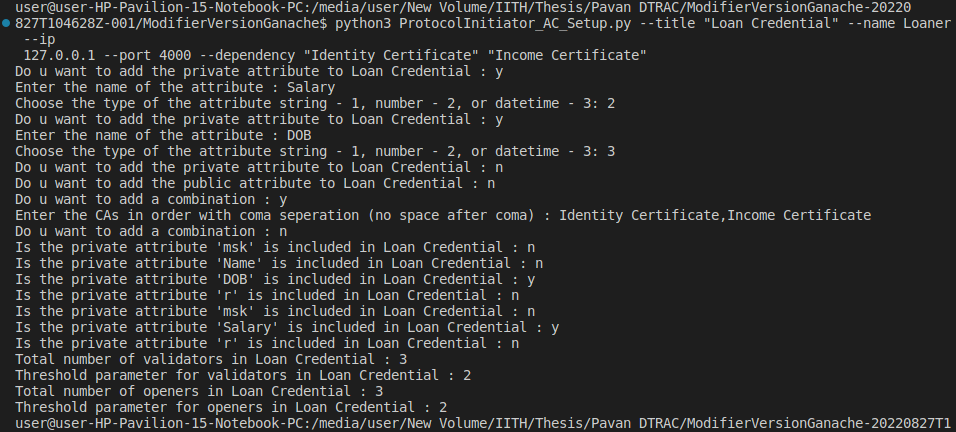


1. Change the address in command line arguments in the python files with the corresponding Smart Contract addresses:
   1. SP.py: verify SC address is above one; bank address is SP address from Ganache
   2. ProtocolInitiator\_DeployContracts.py:
2. Now run ProtocolInitiator\_DeployContracts.py
   1. This does the deployed SC publicly available

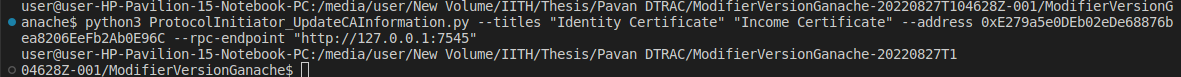


* 1. No outputs will be shown

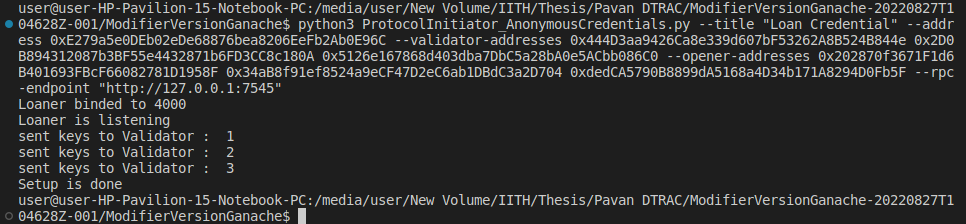
1. Run ProtocolInitiator\_AC\_Setup.py
   1. This sets up parameters for AC
   2. Make sure to give DOB before Salary



1. Run ProtocolInitiator\_UpdateCAInformation.py
   1. Updates CA public parameters and sets up SC



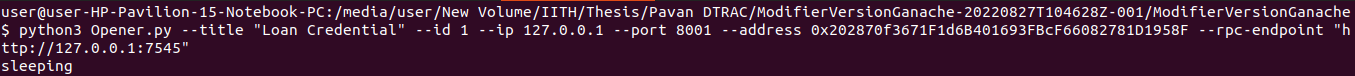
* 1. No output

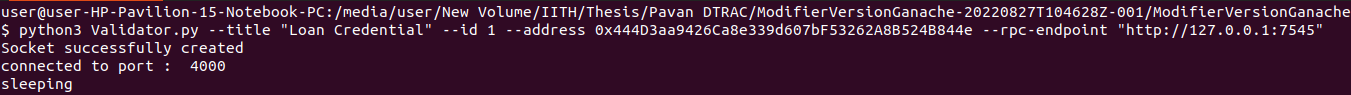
1. Run ProtocolInitiator\_AnonymousCredentials.py
   1. Sends the addresses of the Validators to SC such that only those Validators can access Issue SC; similarly the Opener addresses so that they can only access Opening SC.
   2. Actual key sending happens here..
2. These all were Admin’s Job

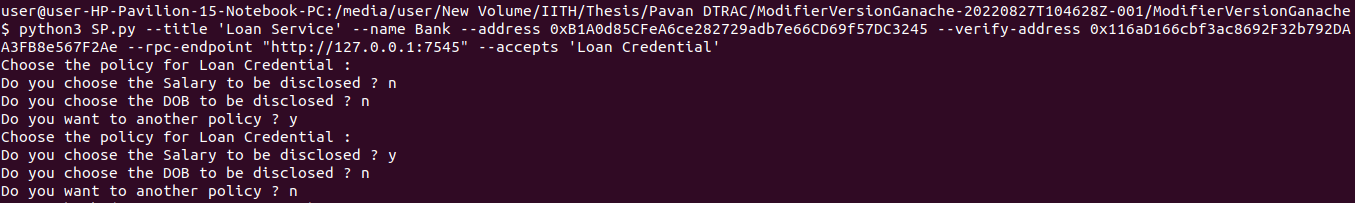
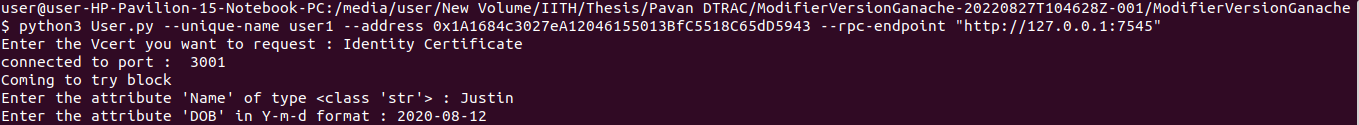
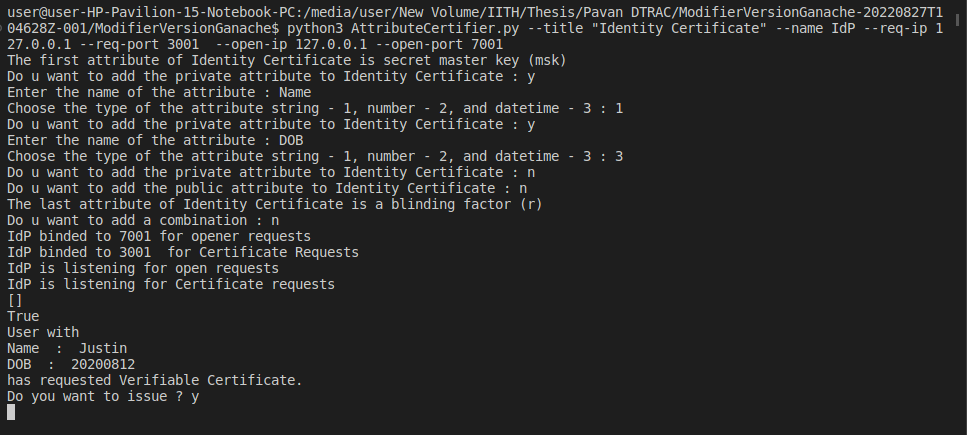
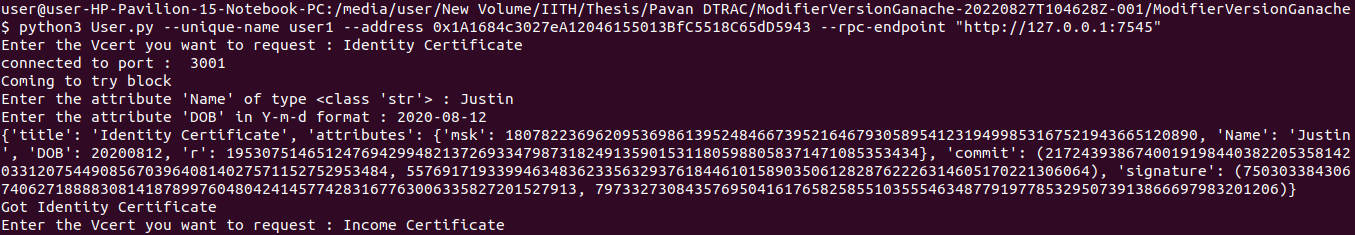
Validators and Openers:

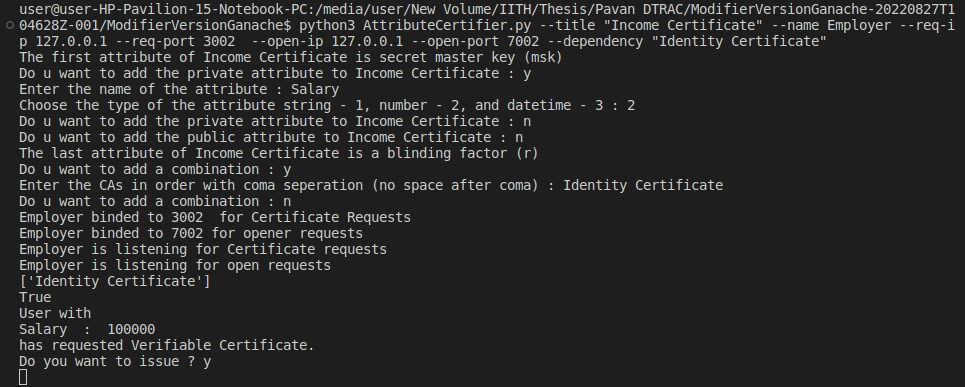
1. Update addresses Validator.py and Opener.py (Only for the first time)
2. Run all the 3 validators and all the 3 openers together (due to epoch constaints, all has to run within 2 mins) in separate terminal windows

Opener:

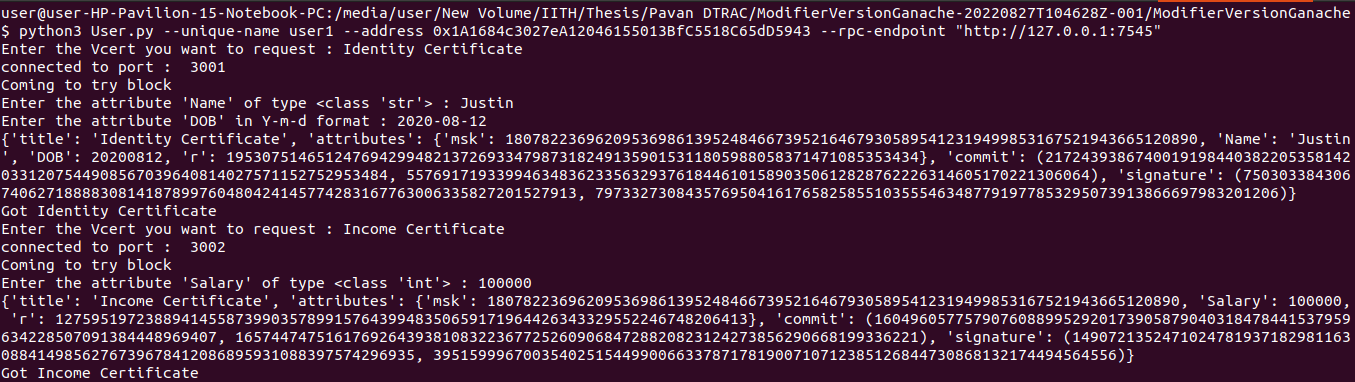
 Validator:



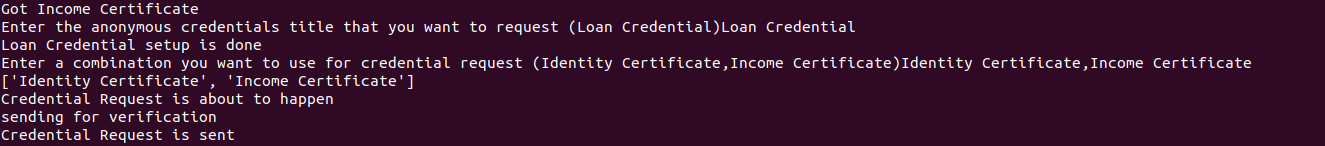
1. Now Run SP.py
   1. Do the policy configurations.
2. Run User.py
   1. User enters the Vcert type.
   2. Respective attribute values of the Vcert will be required by the user to enter
   3. This request now goes to the respective Vcert authority, which was the very first terminal in Vstudio.
   4. Now that gets the question to Approve, which may do the verification offline and then approve:
   5. Once accepted, the user gets the response, which is the committed attributes and the signed commitment
      1. User sends committment of the attributes and the attributes in plaintext, and the ZKPoK
      2. The Certifier verifies this and sends the response of signed commitment
      3. Now at user end, this reponse is displayed:
      4. Now user again gets a request to send for any more Vcerts.
      5. User sends same for 2nd Vcert.
      6. Response received at Certifier 2:



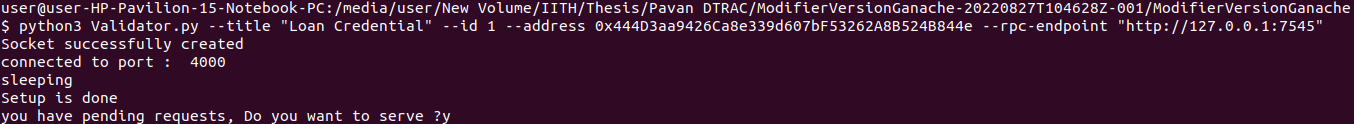
At User:

* + 1. Now user has both Vcerts. Note that the msk is same.
    2. User is now asked to enter the Service (called anonymous credential title) that it wants to avail; followed by Vcert combination to use.

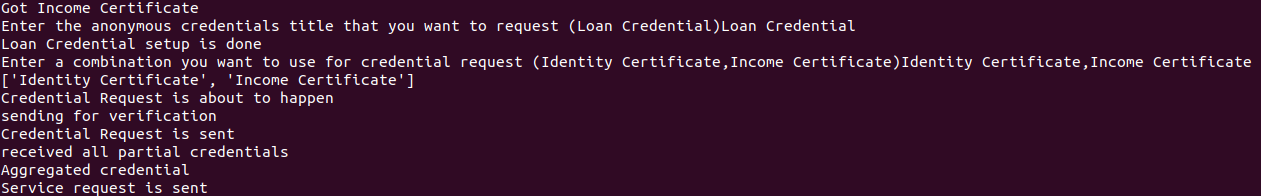
Loan Credential Setup is done takes some time. Not if we remove the comment for sleep

NOTE: This takes a long time to execute like 5 mins

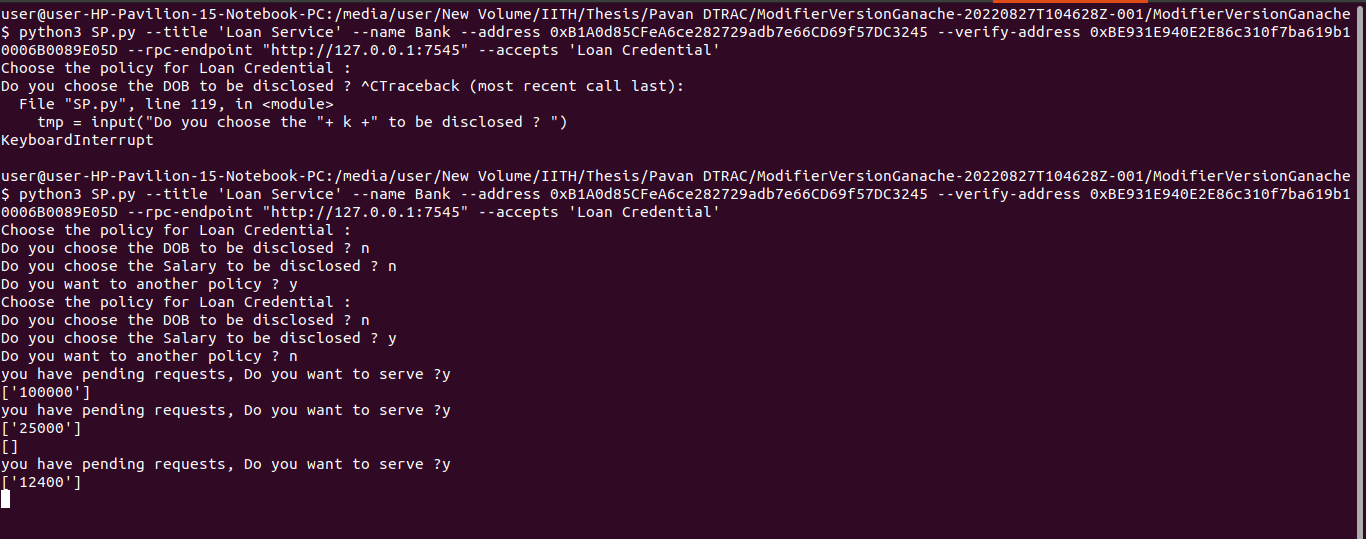
* + 1. Now this request for Vcert goes to each of the Validators.



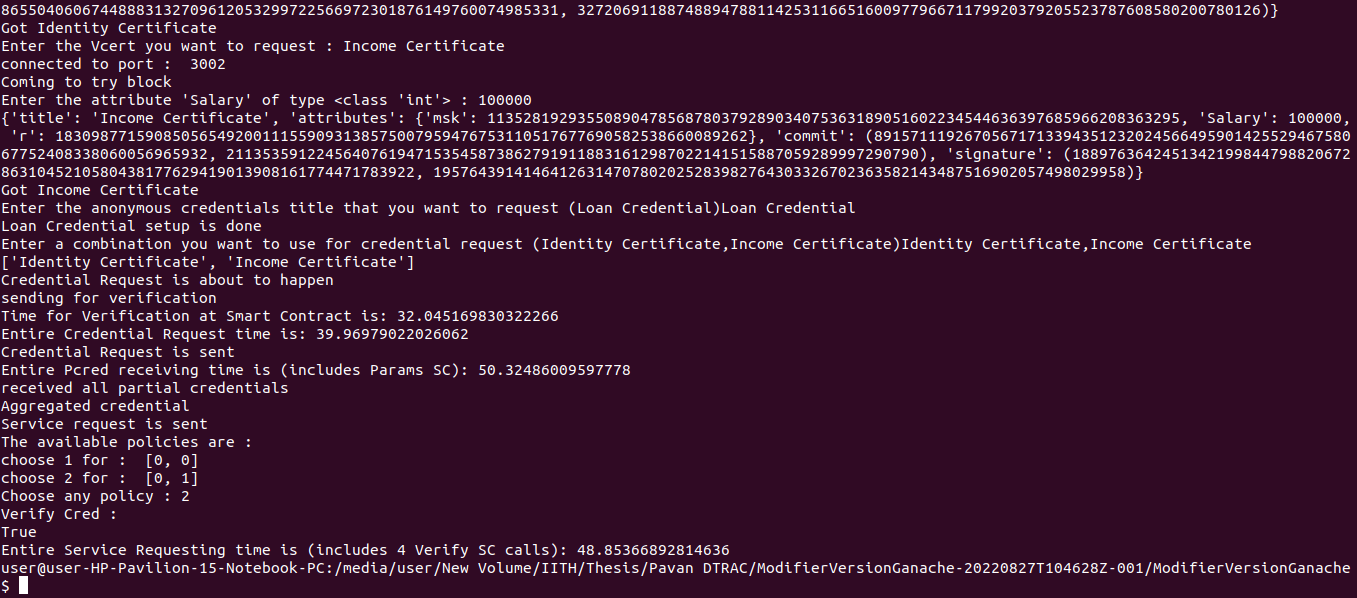
* + 1. Once threshold number of validators have certified, and shared Pcred to User, User aggregates them and sends to the service.



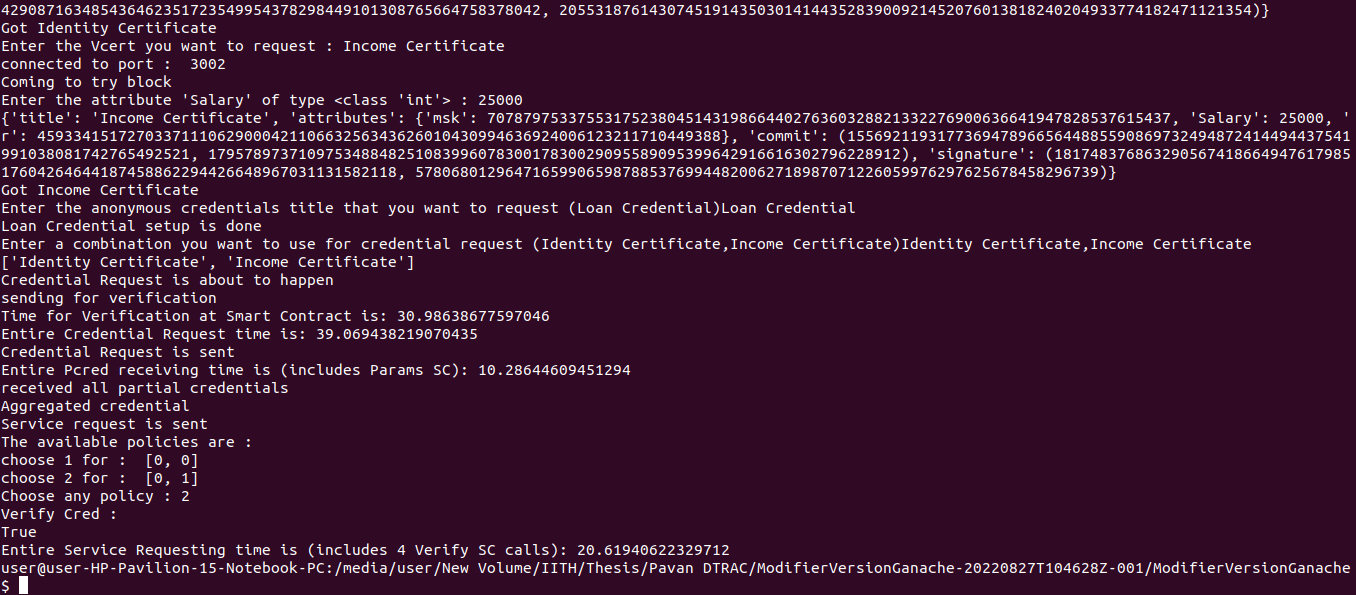
* + 1. Now the User is asked to choose the policy he prefers which SP has pre configured.
    2. Upon choosing the policy, the aggregated credential is sent for Verification to SP.
    3. Verification happens on-chain and is displayed the status of certification to User. We see the Verification has failed as the attribute order was not maintained while giving attribute details at Step 8 of running ProtocolInitiator\_AC\_Setup.py.
    4. After successful verification, user can avail any SP service.
    5. Note that we can run this User any no. of times.
    6. There is also respective Update in SP.py:



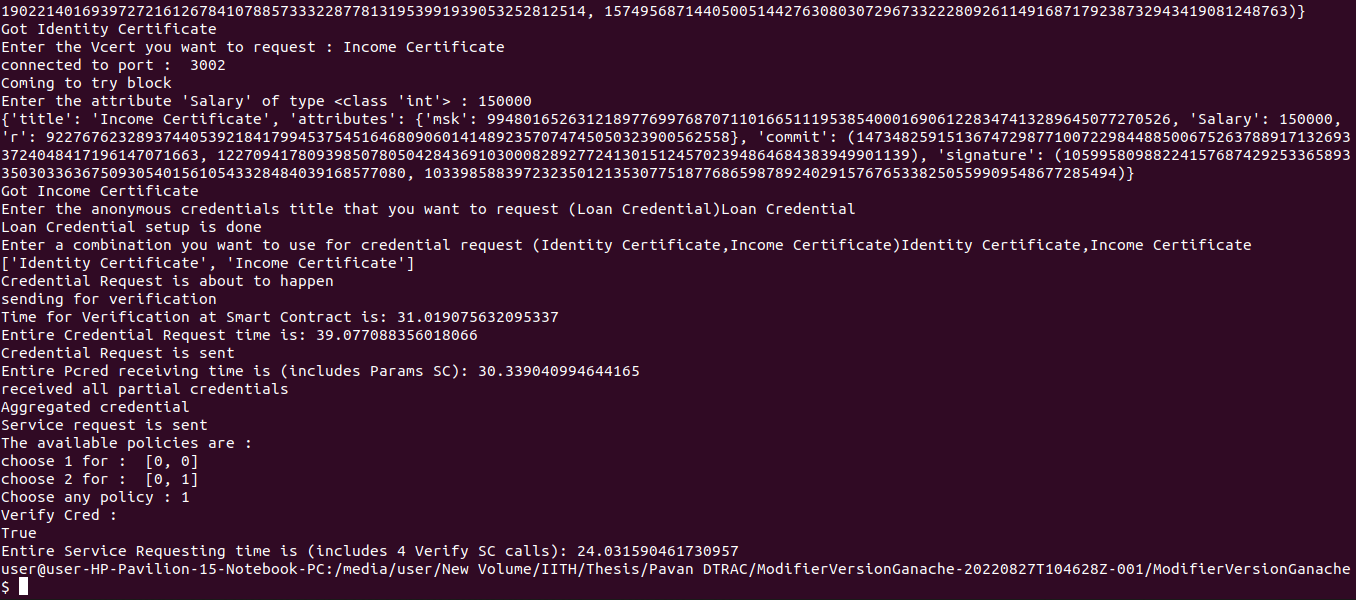
Successful Verification:

The Entire Cred Request time is high bcoz there was wait at validator.

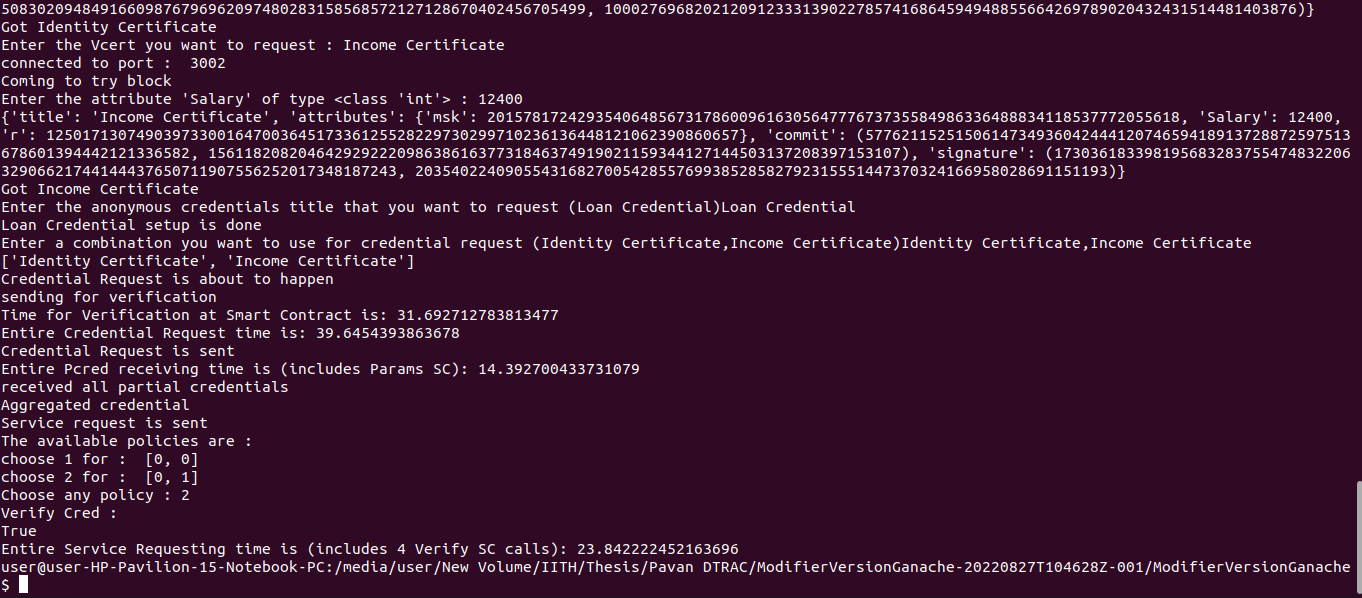
2nd try



3rd Try:



4th try with all sleeps in User.py commented: Improvement in Params SC is there as there was a sleep of 15 sec between Pcreds.



Every time the code is to be re-run, delete the files inside ROOT and rerun everything.