Digital certification using Blockchain

Marulasidda Swamy K S and Naveenkumar Adi Raman

University of Montreal

Proof of Concept

Issuing Phase:

Get the source code and install Required packages

1. Run

```
git clone https://github.com/blockchain-certificates/cert-issuer.git && cd cert-issuer
```

2. Make below changes to fix compilation issues:

```
Go to : ~/.local/lib/python3.6/site-packages/merkletools-1.0.2.dist-info and edit METADATA and metadata.json
```

3. Downgrade jsonschema to the version 2.6

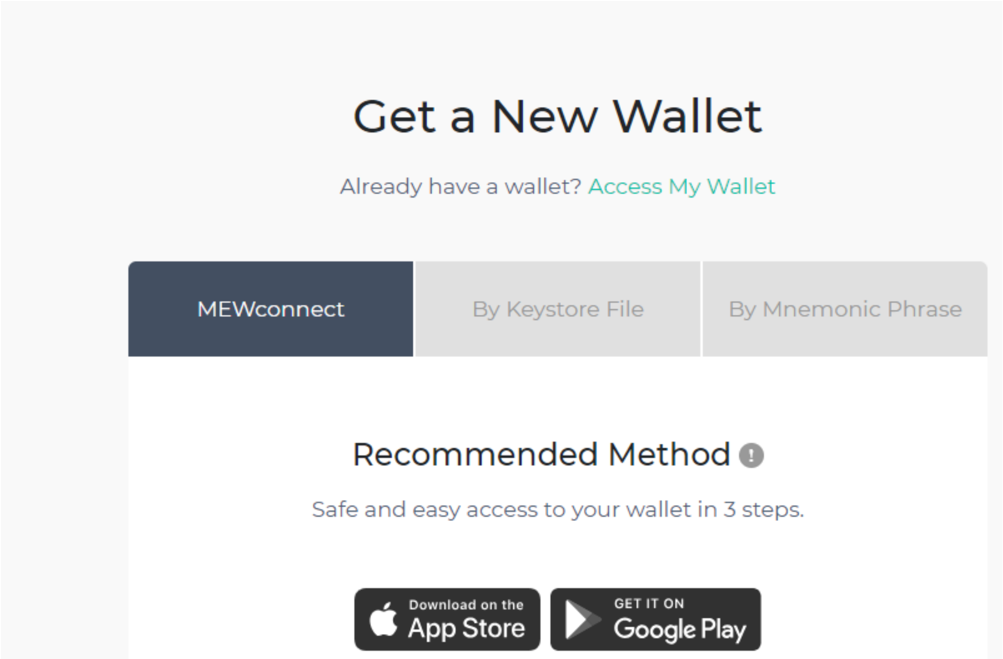
```
pip uninstall jsonschema
pip install jsonschema==2.6
```

4. Run

```
python setup.py experimental –blockchain=ethereum
python setup.py install
```

Create an Ethereum issuing address

1. In Ethereum a public/private key pair is the same across all test/main networks. So, we don't need to have a new pair if later we need to run on the main network.
2. Go to https://www.myetherwallet.com/



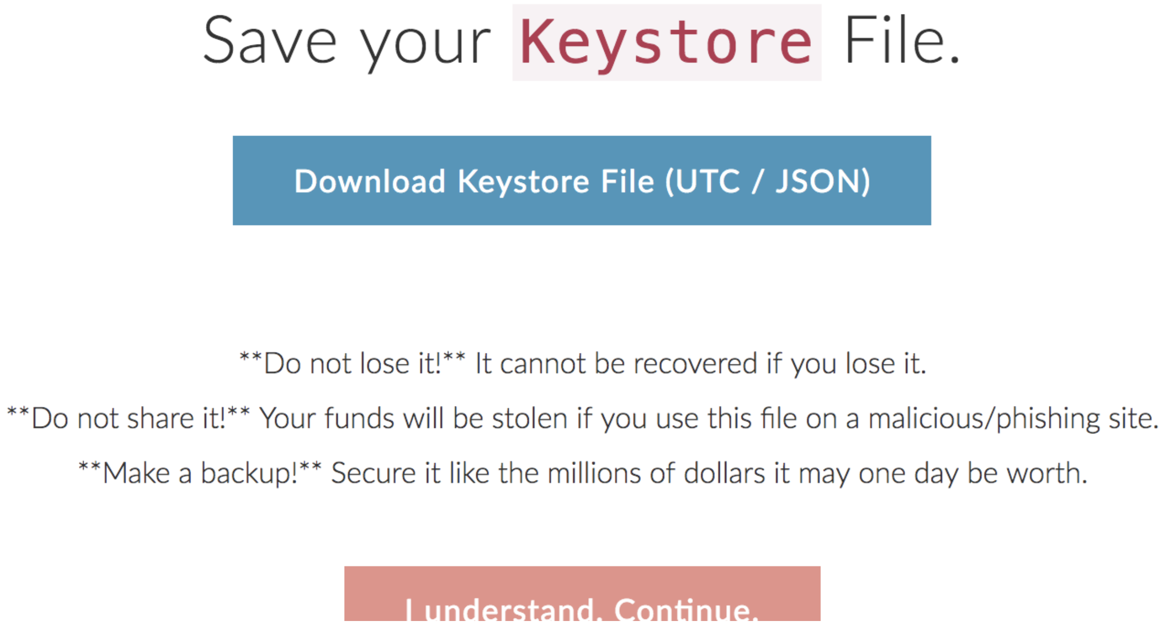
Get a New Wallet

Already have a wallet? Access My Wallet

Recommended Method

Safe and easy access to your wallet in 3 steps.

3. Download and save issuer credentials (Keystore file).



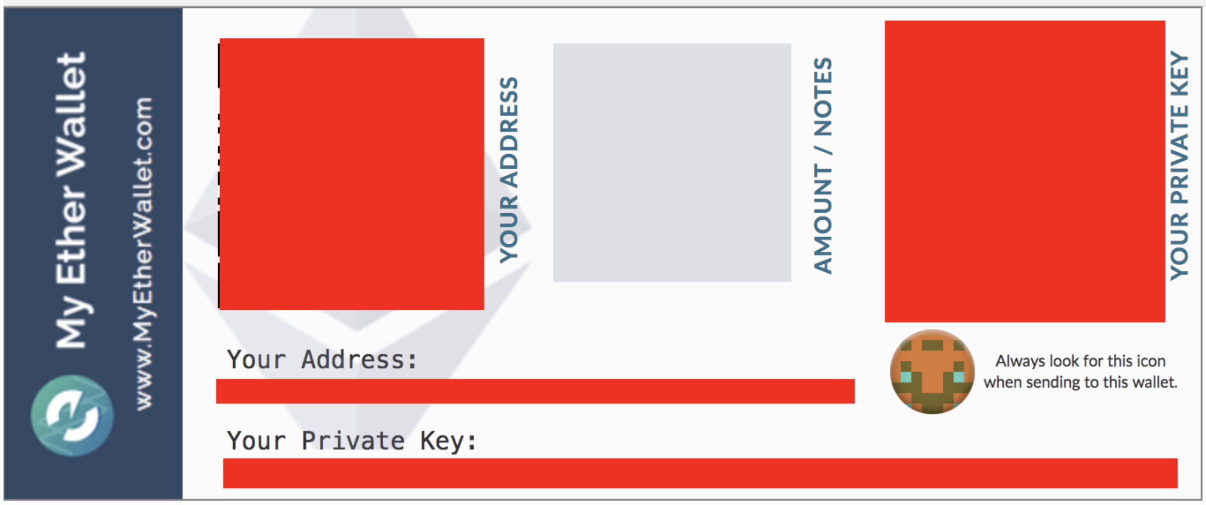
Save your Keystore File.

Do not lose it! It cannot be recovered if you lose it.

Do not share it! Your funds will be stolen if you use this file on a malicious/phishing site.

Make a backup! Secure it like the millions of dollars it may one day be worth.

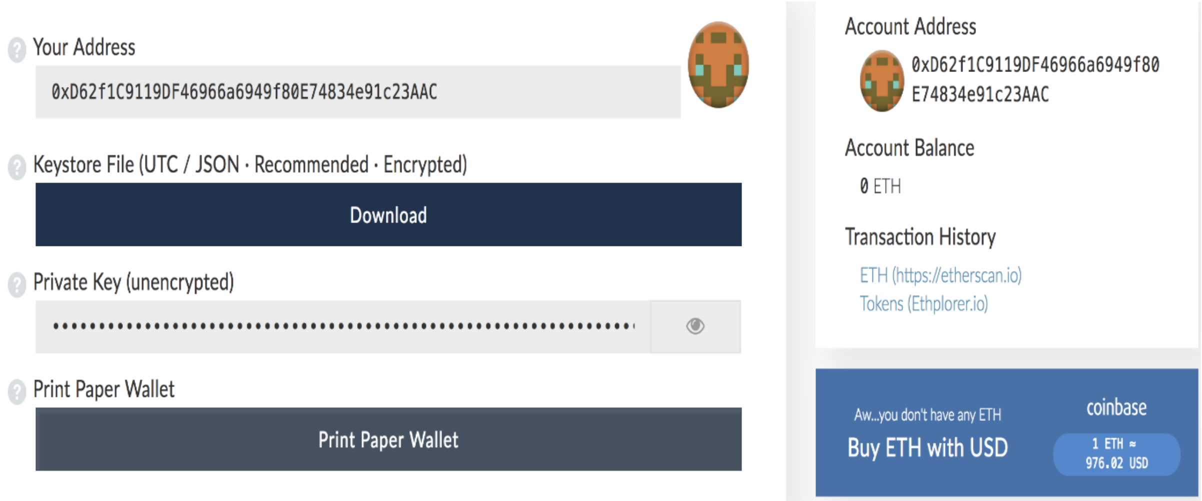
4. We can print a paper wallet.



Your Address: [redacted]

Your Private Key: [redacted]

5. Your address can be known as your Public Key. It is what you share with people so they can send you Ether or Tokens.
6. You may use the Keystore file or your Private key to continue.



Your Address

0xD62f1C9119DF46966a6949f80E74834e91c23AAC

Keystore File (UTC / JSON · Recommended · Encrypted)

Download

Private Key (unencrypted)

••

Print Paper Wallet

Print Paper Wallet

Account Address

0xD62f1C9119DF46966a6949f80
E74834e91c23AAC

Account Balance

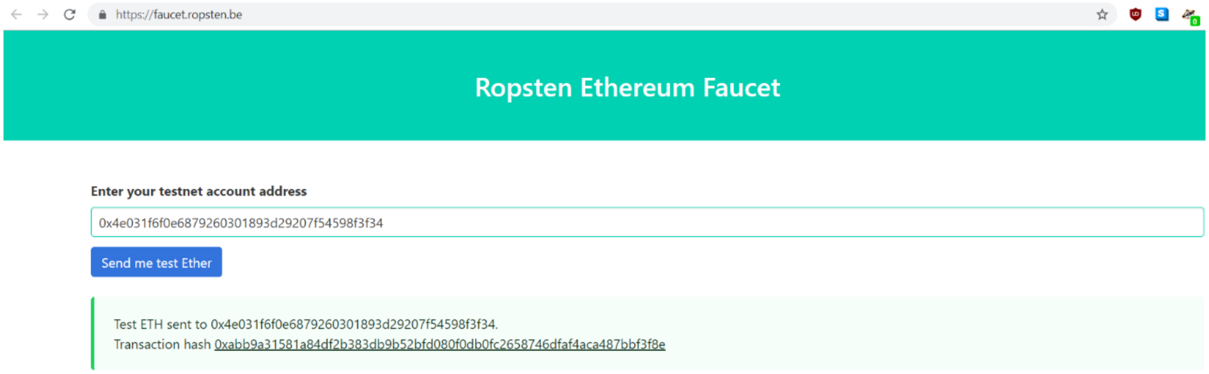
0 ETH

Transaction History

ETH (https://etherscan.io)
Tokens (Ethplorer.io)

Aw...you don't have any ETH
Buy ETH with USD
coinbase
1 ETH ≈ 976.02 USD

7. Before we can issue certificates, we need to have sufficient funds to cover the transaction fee. Why transaction fee? It is paid to miners for their work. We can get some free coins from the network. You can request test coins by searching for "TESTNET Faucet" and entering your issuing public address.
8. Go to http://faucet.ropsten.be:3001/. Enter your public key and click "Send me 3 test ether!".



Ropsten Ethereum Faucet

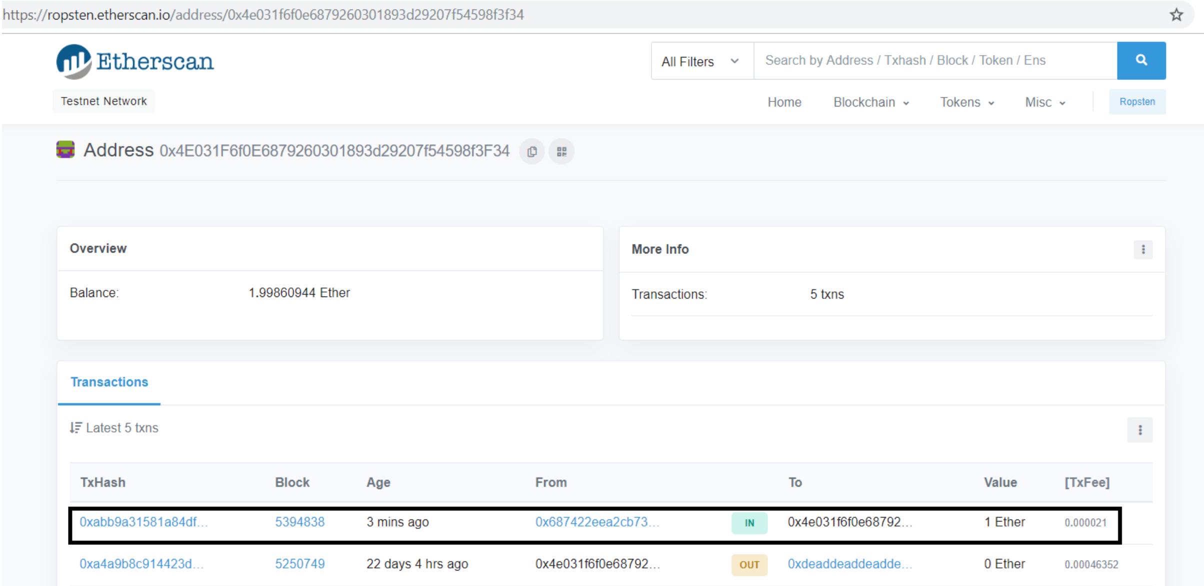
Enter your testnet account address

0x4e031f6f0e6879260301893d29207f54598f3f34

Send me test Ether

Test ETH sent to 0x4e031f6f0e6879260301893d29207f54598f3f34.
Transaction hash 0xabb9a31581a84df2b383db9b52bfd080f0db0fc2658746dfaf4aca487bbf3f8e

9. Check the balance in your account.


https://ropsten.etherscan.io/address/0x4e031f6f0e6879260301893d29207f54598f3f34
Etherscan
Testnet Network
All Filters
Search by Address / Txhash / Block / Token / Ens
Home
Blockchain
Tokens
Misc
Ropsten

Address 0x4E031F6F0E6879260301893d29207f54598f3F34

Overview

Balance: 1.99860944 Ether

More Info

Transactions: 5 txns

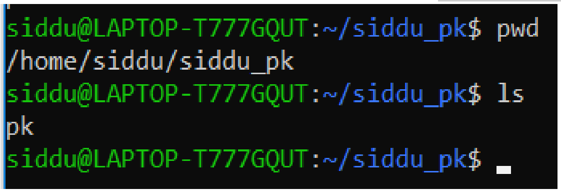
Transactions

Latest 5 txns

TxHash	Block	Age	From		To	Value	[TxFee]
0xabb9a31581a84df...	5394838	3 mins ago	0x687422eea2cb73...	IN	0x4e031f6f0e68792...	1 Ether	0.000021
0xa4a9b8c914423d...	5250749	22 days 4 hrs ago	0x4e031f6f0e68792...	OUT	0xdeaddeaddeadde...	0 Ether	0.00046352

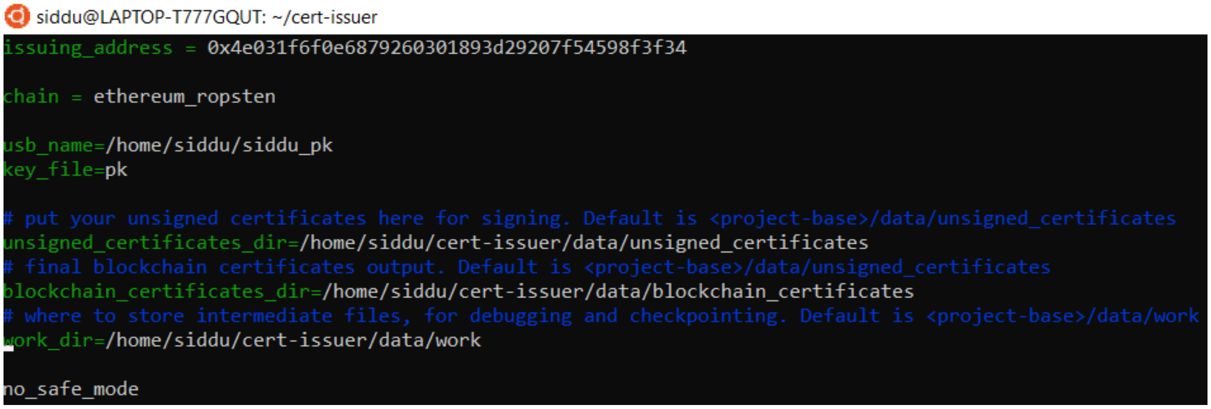
Configuring cert-issuer

1. We need to save you the private key to somewhere. It is recommended to store the private key on a USB stick and unplug it when not used. But to save time plugging in and out, we will save the private key locally. For example, `/home/siddu/siddu-pk` folder and `pk` file.



```
siddu@LAPTOP-T777GQUT:~/siddu_pk$ pwd
/home/siddu/siddu_pk
siddu@LAPTOP-T777GQUT:~/siddu_pk$ ls
pk
siddu@LAPTOP-T777GQUT:~/siddu_pk$ _
```

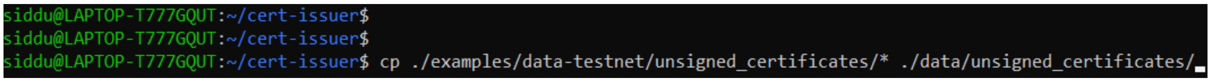
2. Then, you should see a `conf_ethtest.ini` under the `cert-issuer` root folder. Open it using any of your favorite editors. Fill in the issuing_address, choose ethereum_ropsten as the blockchain, enter the usb_name and key_file as the private key storage location, and set the three dirs. Remember to rename the config to conf.ini.



```
issuing_address = 0x4e031f6f0e6879260301893d29207f54598f3f34

chain = ethereum_ropsten

usb_name=/home/siddu/siddu_pk
key_file=pk

# put your unsigned certificates here for signing. Default is <project-base>/data/unsigned_certificates
unsigned_certificates_dir=/home/siddu/cert-issuer/data/unsigned_certificates
# final blockchain certificates output. Default is <project-base>/data/unsigned_certificates
blockchain_certificates_dir=/home/siddu/cert-issuer/data/blockchain_certificates
# where to store intermediate files, for debugging and checkpointing. Default is <project-base>/data/work
work_dir=/home/siddu/cert-issuer/data/work

no_safe_mode
```

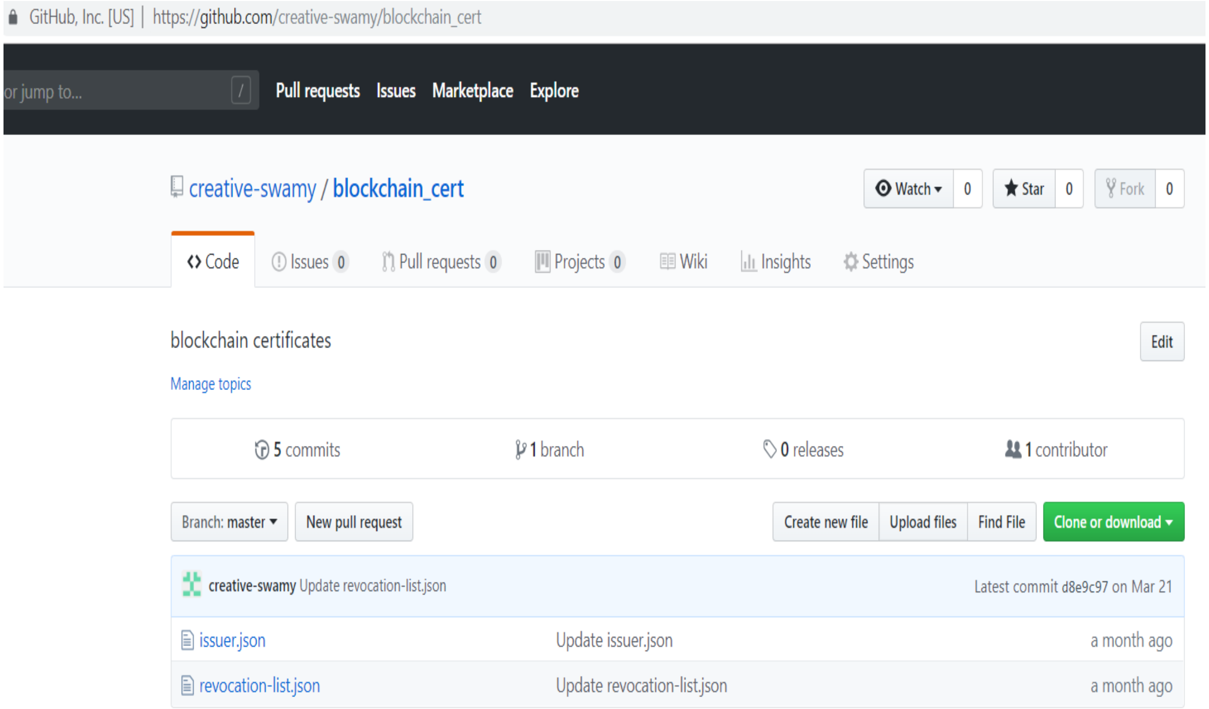
Issuing a certificate

1. Let us use some sample certificates:



```
siddu@LAPTOP-T777GQUT:~/cert-issuer$
siddu@LAPTOP-T777GQUT:~/cert-issuer$
siddu@LAPTOP-T777GQUT:~/cert-issuer$ cp ./examples/data-testnet/unsigned_certificates/* ./data/unsigned_certificates/
```

2. Here you need to edit the unsigned certificates, replace "msBCHdwaQ7N2ypBYupkp6uNxtr9Pg76imj" with your public key.
3. Then, you need to find a place to host the issuer profile. From Open Badges, a Profile is a collection of information that describes the entity or organization using Open Badges. The main usage is to verify the cert is issued from a proved person. Since we are using our owned public key, we need to host the profile file by ourselves.
4. Download two sample files, one is issuer profile and another one is revocation list.
   a. https://www.blockcerts.org/samples/2.0/issuer-testnet.json
   b. https://www.blockcerts.org/samples/2.0/revocation-list-testnet.json
5. Then, open the two files, and you will see a compressed JSON. What you need to do is first replace "msBCHdwaQ7N2ypBYupkp6uNxtr9Pg76imj" with your public key in issuer-testnet.json file, and upload it to somewhere, like GitHub, where you need to get a stable link for direct download. Then, replace the links in the two JSON files with the correct one.



GitHub, Inc. [US] | https://github.com/creative-swamy/blockchain_cert

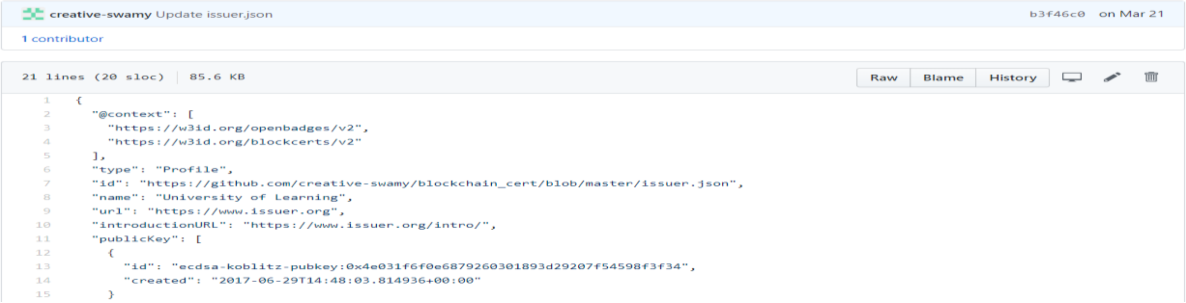
creative-swamy / blockchain_cert

blockchain certificates

5 commits | 1 branch | 0 releases | 1 contributor

creative-swamy Update revocation-list.json — Latest commit d8e9c97 on Mar 21

issuer.json	Update issuer.json	a month ago
revocation-list.json	Update revocation-list.json	a month ago

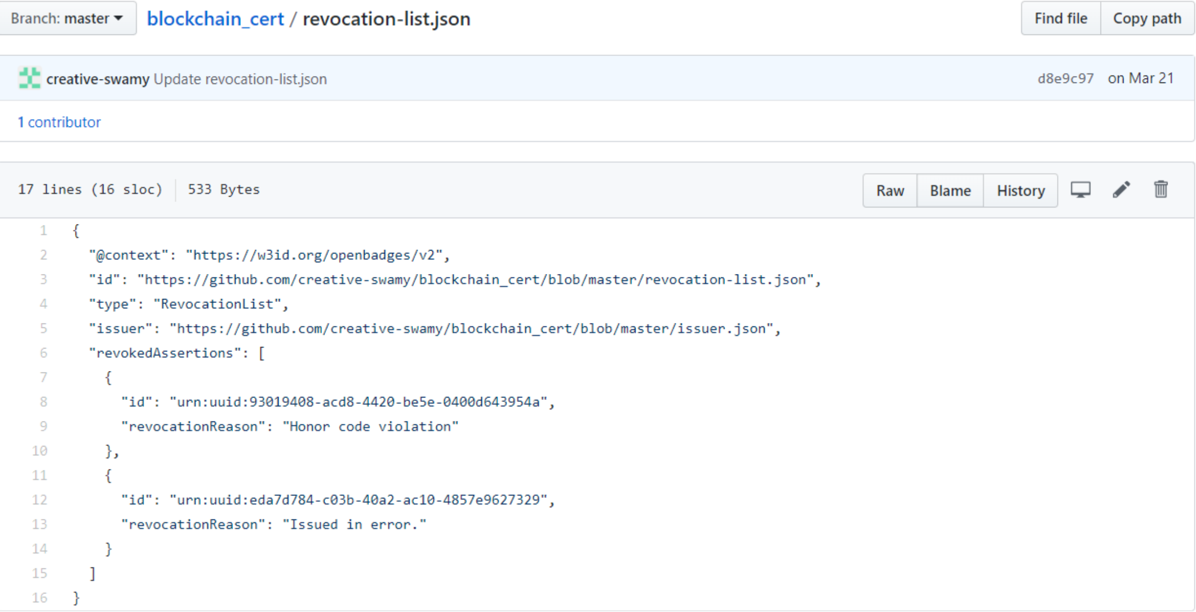


creative-swamy Update issuer.json — b3f46c0 on Mar 21

1 contributor

21 lines (20 sloc) | 85.6 KB

```
{
  "@context": [
    "https://w3id.org/openbadges/v2",
    "https://w3id.org/blockcerts/v2"
  ],
  "type": "Profile",
  "id": "https://github.com/creative-swamy/blockchain_cert/blob/master/issuer.json",
  "name": "University of Learning",
  "url": "https://www.issuer.org",
  "introductionURL": "https://www.issuer.org/intro/",
  "publicKey": [
    {
      "id": "ecdsa-koblitz-pubkey:0x4e031f6f0e6879260301893d29207f54598f3f34",
      "created": "2017-06-29T14:48:03.814936+00:00"
    }
```



blockchain_cert / revocation-list.json

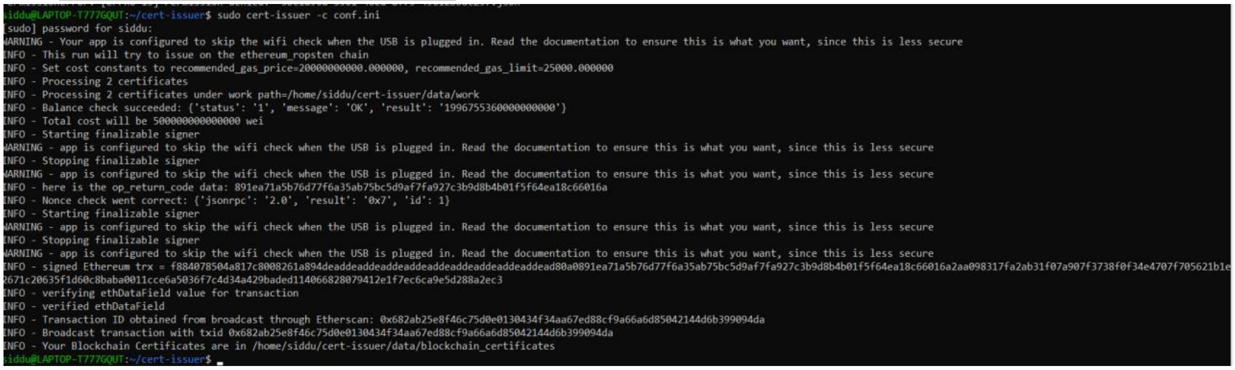
creative-swamy Update revocation-list.json — d8e9c97 on Mar 21

1 contributor

17 lines (16 sloc) | 533 Bytes

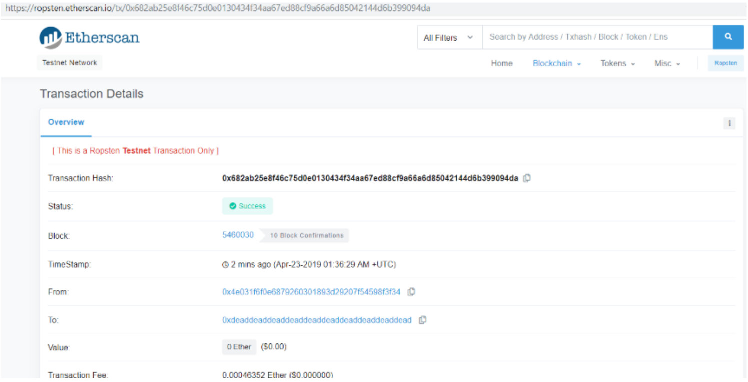
```
{
  "@context": "https://w3id.org/openbadges/v2",
  "id": "https://github.com/creative-swamy/blockchain_cert/blob/master/revocation-list.json",
  "type": "RevocationList",
  "issuer": "https://github.com/creative-swamy/blockchain_cert/blob/master/issuer.json",
  "revokedAssertions": [
    {
      "id": "urn:uuid:93019408-acd8-4420-be5e-0400d643954a",
      "revocationReason": "Honor code violation"
    },
    {
      "id": "urn:uuid:eda7d784-c03b-40a2-ac10-4857e9627329",
      "revocationReason": "Issued in error."
    }
  ]
}
```

6. Run `cert-issuer -c conf.ini` to see if everything is working.



```
siddu@LAPTOP-T777GQUT:~/cert-issuer$ sudo cert-issuer -c conf.ini
[sudo] password for siddu:
WARNING - Your app is configured to skip the wifi check when the USB is plugged in. Read the documentation to ensure this is what you want, since this is less secure
INFO - This run will try to issue on the ethereum_ropsten chain
INFO - Set cost constants to recommended_gas_price=20000000000.000000, recommended_gas_limit=25000.000000
INFO - Processing 2 certificates
INFO - Processing 2 certificates under work path=/home/siddu/cert-issuer/data/work
INFO - Balance check succeeded: {'status': '1', 'message': 'OK', 'result': '1996755360000000000'}
INFO - Total cost will be 500000000000000 wei
INFO - Starting finalizable signer
WARNING - app is configured to skip the wifi check when the USB is plugged in. Read the documentation to ensure this is what you want, since this is less secure
INFO - Stopping finalizable signer
WARNING - app is configured to skip the wifi check when the USB is plugged in. Read the documentation to ensure this is what you want, since this is less secure
INFO - here is the op_return_code data: 891ea71a5b76d77f6a35ab75bc5d9af7fa927c3b9d8b4b01f5f64ea18c66016a
INFO - Nonce check went correct: {'jsonrpc': '2.0', 'result': '0x7', 'id': 1}
INFO - Starting finalizable signer
WARNING - app is configured to skip the wifi check when the USB is plugged in. Read the documentation to ensure this is what you want, since this is less secure
INFO - Stopping finalizable signer
WARNING - app is configured to skip the wifi check when the USB is plugged in. Read the documentation to ensure this is what you want, since this is less secure
INFO - signed Ethereum trx = f884078504a817c8008261a894deaddeaddeaddeaddeaddeaddeaddeaddeaddead80a0891ea71a5b76d77f6a35ab75bc5d9af7fa927c3b9d8b4b01f5f64ea18c66016a2aa098317fa2ab31f07a907f3738f0f34e4707f705621b1e
2671c20635f1d60c8baba0011cce6a5036f7c4d34a429baded114066828079412e1f7ec6ca9e5d288a2ec3
INFO - verifying ethDataField value for transaction
INFO - verified ethDataField
INFO - Transaction ID obtained from broadcast through Etherscan: 0x682ab25e8f46c75d0e0130434f34aa67ed88cf9a66a6d85042144d6b399094da
INFO - Broadcast transaction with txid 0x682ab25e8f46c75d0e0130434f34aa67ed88cf9a66a6d85042144d6b399094da
INFO - Your Blockchain Certificates are in /home/siddu/cert-issuer/data/blockchain_certificates
siddu@LAPTOP-T777GQUT:~/cert-issuer$
```

7. Read the debug info, it is useful. You will find a string of txid, which is your transaction ID. Now you can check this transaction using Etherscan.



https://ropsten.etherscan.io/tx/0x682ab25e8f46c75d0e0130434f34aa67ed88cf9a66a6d85042144d6b399094da

Transaction Details

[This is a Ropsten **Testnet** Transaction Only]

Transaction Hash: 0x682ab25e8f46c75d0e0130434f34aa67ed88cf9a66a6d85042144d6b399094da

Status: Success

Block: 5460030 10 Block Confirmations

TimeStamp: 2 mins ago (Apr-23-2019 01:36:29 AM +UTC)

From: 0x4e031f6f0e6879260301893d29207f54598f3f34

To: 0xdeaddeaddeaddeaddeaddeaddeaddeaddeaddead

Value: 0 Ether ($0.00)

Transaction Fee: 0.00046352 Ether ($0.000000)

8. Now you get a blockchain-based certificate under data/blockchain_certificates.

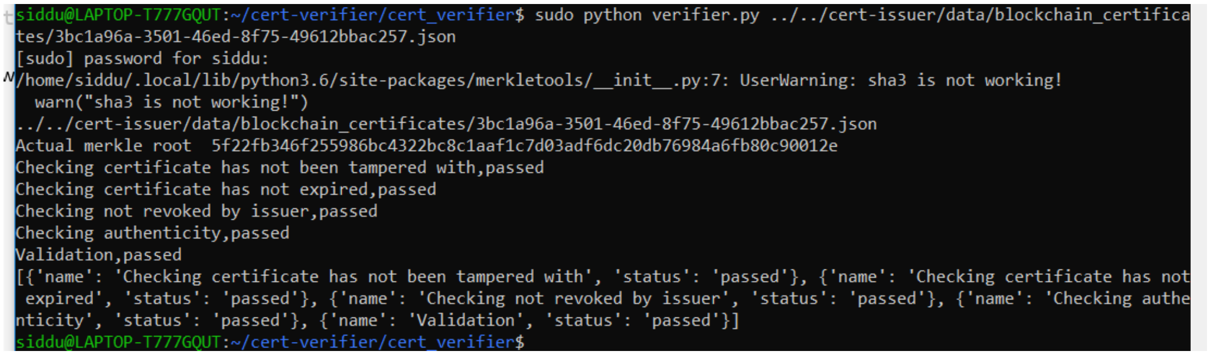
Verification Phase

Traditional way of verifying the certificate using Blockcerts verifier code

1. Clone cert-verifier repo from GitHub:

```
git clone https://github.com/blockchain-certificates/cert-verifier.git
&& cd cert-verifier
```

2. Use certificates which are issued by cert-issuer (from certificate issuing step) and run :

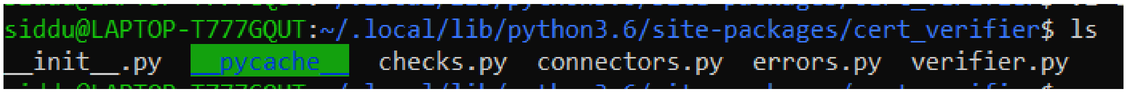


```
siddu@LAPTOP-T777GQUT:~/cert-verifier/cert_verifier$ sudo python verifier.py ../../cert-issuer/data/blockchain_certifica
tes/3bc1a96a-3501-46ed-8f75-49612bbac257.json
[sudo] password for siddu:
/home/siddu/.local/lib/python3.6/site-packages/merkletools/__init__.py:7: UserWarning: sha3 is not working!
  warn("sha3 is not working!")
../../cert-issuer/data/blockchain_certificates/3bc1a96a-3501-46ed-8f75-49612bbac257.json
Actual merkle root  5f22fb346f255986bc4322bc8c1aaf1c7d03adf6dc20db76984a6fb80c90012e
Checking certificate has not been tampered with,passed
Checking certificate has not expired,passed
Checking not revoked by issuer,passed
Checking authenticity,passed
Validation,passed
[{'name': 'Checking certificate has not been tampered with', 'status': 'passed'}, {'name': 'Checking certificate has not
 expired', 'status': 'passed'}, {'name': 'Checking not revoked by issuer', 'status': 'passed'}, {'name': 'Checking authe
nticity', 'status': 'passed'}, {'name': 'Validation', 'status': 'passed'}]
siddu@LAPTOP-T777GQUT:~/cert-verifier/cert_verifier$
```

3. If the certificate issued is genuine and validated, then all verification steps show the result as "passed" (As depicted in the above screenshot).

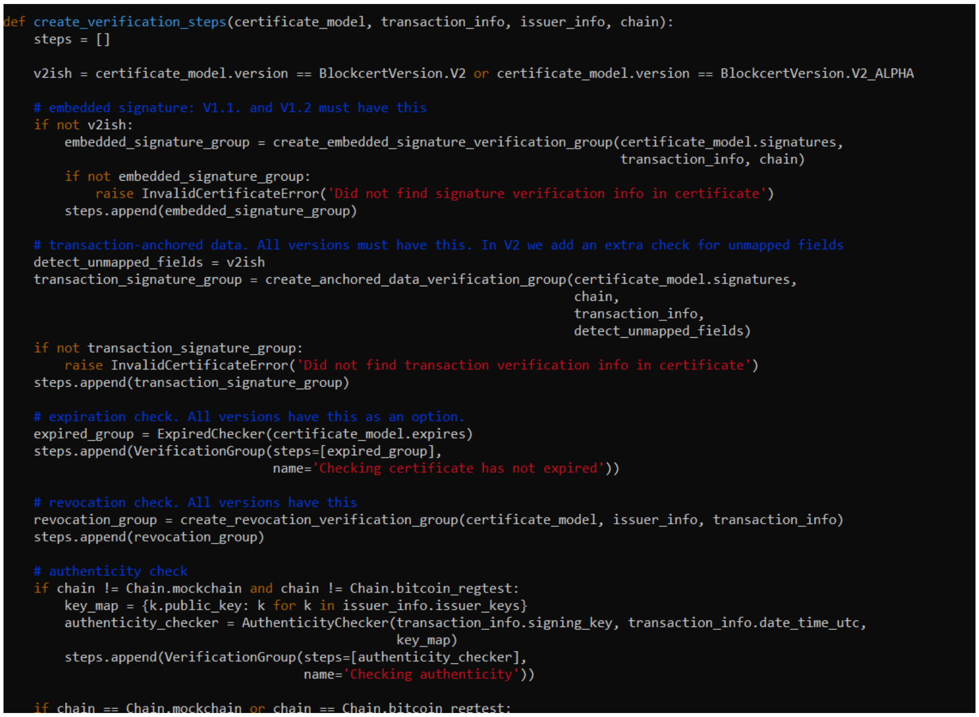
Observations made with the traditional way of verifying the certificate

1. Steps followed to complete verification of a given certificate:
   a. Check if the certificate has been modified/tampered.
   b. Check if the certificate has been expired.
   c. Check if the certificate has been revoked by the issuer.
   d. Check if the issued certificate is authentic and genuine.
2. The complete verification process is implemented in **checks.py** file



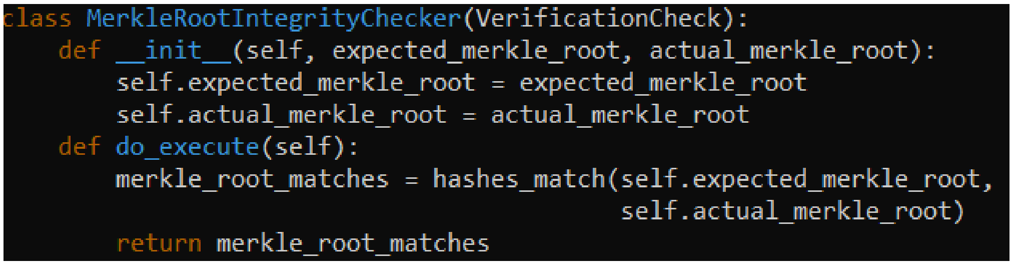
```
siddu@LAPTOP-T777GQUT:~/.local/lib/python3.6/site-packages/cert_verifier$ ls
__init__.py  __pycache__  checks.py  connectors.py  errors.py  verifier.py
```

3. The function which implements verification :



```
def create_verification_steps(certificate_model, transaction_info, issuer_info, chain):
    steps = []

    v2ish = certificate_model.version == BlockcertVersion.V2 or certificate_model.version == BlockcertVersion.V2_ALPHA

    # embedded signature: V1.1. and V1.2 must have this
    if not v2ish:
        embedded_signature_group = create_embedded_signature_verification_group(certificate_model.signatures,
                                                                                transaction_info, chain)

        if not embedded_signature_group:
            raise InvalidCertificateError('Did not find signature verification info in certificate')
        steps.append(embedded_signature_group)

    # transaction-anchored data. All versions must have this. In V2 we add an extra check for unmapped fields
    detect_unmapped_fields = v2ish
    transaction_signature_group = create_anchored_data_verification_group(certificate_model.signatures,
                                                                          chain,
                                                                          transaction_info,
                                                                          detect_unmapped_fields)

    if not transaction_signature_group:
        raise InvalidCertificateError('Did not find transaction verification info in certificate')
    steps.append(transaction_signature_group)

    # expiration check. All versions have this as an option.
    expired_group = ExpiredChecker(certificate_model.expires)
    steps.append(VerificationGroup(steps=[expired_group],
                                   name='Checking certificate has not expired'))

    # revocation check. All versions have this
    revocation_group = create_revocation_verification_group(certificate_model, issuer_info, transaction_info)
    steps.append(revocation_group)

    # authenticity check
    if chain != Chain.mockchain and chain != Chain.bitcoin_regtest:
        key_map = {k.public_key: k for k in issuer_info.issuer_keys}
        authenticity_checker = AuthenticityChecker(transaction_info.signing_key, transaction_info.date_time_utc,
                                                   key_map)
        steps.append(VerificationGroup(steps=[authenticity_checker],
                                       name='Checking authenticity'))

    if chain == Chain.mockchain or chain == Chain.bitcoin_regtest:
```

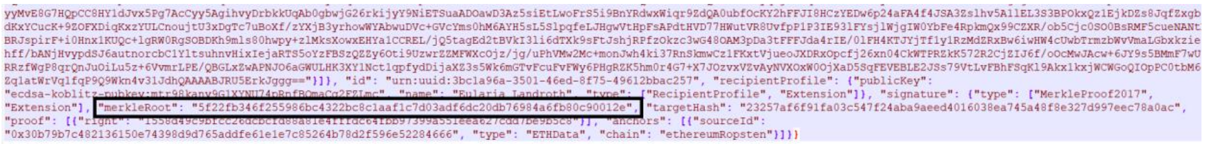
4. Merkle root verification is also one of the steps involved in the verification process:



```
class MerkleRootIntegrityChecker(VerificationCheck):
    def __init__(self, expected_merkle_root, actual_merkle_root):
        self.expected_merkle_root = expected_merkle_root
        self.actual_merkle_root = actual_merkle_root
    def do_execute(self):
        merkle_root_matches = hashes_match(self.expected_merkle_root,
                                           self.actual_merkle_root)
        return merkle_root_matches
```

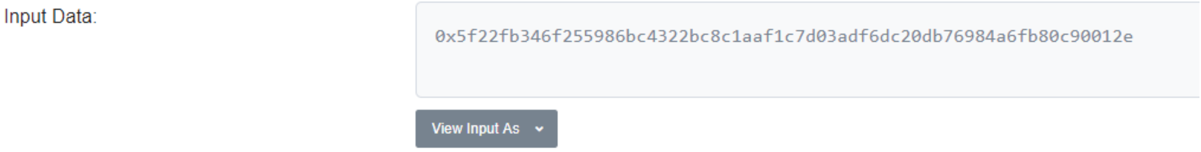
   Where in Merkle root present in the shared certificate is compared with the Merkle root present in the transaction details.

5. Merkle root reference in the certificate:



```
yyMvE8G7HQpCC8HY1dJvx5Pg7AcCyy5AgihvyDrbkkUqAb0gbwjG26rkijyY9NiETSuaADOawD3Az5siEtLwoFrS5i9BnYRdwxWiqr9ZdQA0ubfOcKY2hFFJI8HCzYEDw6p24aFA4f4JSA3Zslhv5A1lEL3S3BPOkxQz1EjkDZs8JqfZxgb
dKxYCucK+9ZOFXDiqKxzYULCnoujtU3xDgTc7uBoXf/zYXjB3yrhowWYAbwuDVc+GVcYms0hM6AYH5sL5SlpqfeLJHgwVtHpFsAPdtHVD77HWutVR8UvfpP1P3IE93lFYsjlWjgIW0YbFe4RpkmQx99CZXR/ob5Cjc0SO0BsRMF5cueNANt
BRJspirF+i0Hnx1KUQc+1gRW0RgSOBDKh9mls80hwpy+zlMksXowxEHYa1CCREL/jQ5tagEd2tBVkI3li6dTXk9sFtJshjRPfzOkzc3wG48OAM3pDa3tFFFJda4rIE/0lFH4KTJYjTf1y1RzMdZRxBw6iwHW4cUwbTrmzbWvVmaLGbxkzie
hff/bANjHvypdSJ6autnorcbC1YltsuhnvHixIejaRTS5oYzFBSzQZZy6Oti9UzwrZZMFWXcOjz/jg/uPhVMw2Mc+monJwh4ki37RnSkmwCz1FKxtVjueoJXDRxOpcfj26xn04CkWTPRZkK572R2CjZIJ6f/oOcMwJAcw+6JY9s5BMmF7wU
RRzfWgP8qrQnJuOiLu5z+6VvmrLPE/QBGLxZwAPNJO6aGWULHK3XY1Nct1qpfydDijaXZ3s5Wk6mGTvFcuFvFWy6PHgRZK5hm0r4G7+X7JOzvxVZvAyNVXOxW0OjXaD5SqFEVEBLE2JSs79VtLvFBhFSqKl9Akx1kxjWCWGoQIOpPC0tbM6
ZqlatWrVqlfqP9Q9Wkn4v31JdhQAAAABJRU5ErkJggg=="}]}, "id": "urn:uuid:3bc1a96a-3501-46ed-8f75-49612bbac257", "recipientProfile": {"publicKey":
"ecdsa-koblitz-pubkey:mtr98kany9G1XYNU74pRnfBOmaCq2EZLmc", "name": "Eularia Landroth", "type": ["RecipientProfile", "Extension"]}, "signature": {"type": ["MerkleProof2017",
"Extension"], "merkleRoot": "5f22fb346f255986bc4322bc8c1aaf1c7d03adf6dc20db76984a6fb80c90012e", "targetHash": "23257af6f91fa03c547f24aba9aeed4016038ea745a48f8e327d997eec78a0ac",
"proof": [{"right": "1558d49c9bfcc26dcbcfd88a81e4fffdc64fbb97399a551eea627cdd7be9b5c8"}], "anchors": [{"sourceId":
"0x30b79b7c482136150e74398d9d765addfe61e1e7c85264b78d2f596e52284666", "type": "ETHData", "chain": "ethereumRopsten"}]}}
```

6. Merkle root reference in the blockchain transaction :



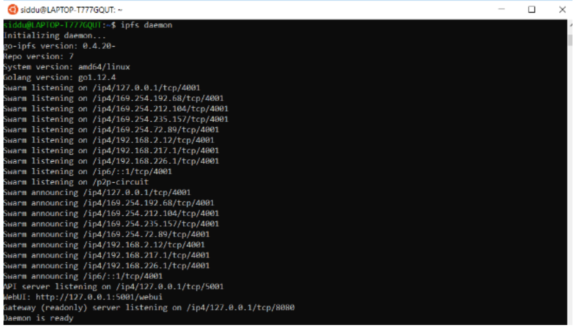
Input Data:

```
0x5f22fb346f255986bc4322bc8c1aaf1c7d03adf6dc20db76984a6fb80c90012e
```

7. Hash of the Merkle root present in the certificate and the Blockchain transaction should match.
8. Please note that the Merkle root hash is shared in the certificate can be exploited by hackers. So, in our proposal, we are demonstrating about how to securely share Merkle root details to the verifier.
9. This way of sharing the Merkle root can facilitate implementing the version management for revoked certificates.

Securely Sharing Merkle root of the certificate to verifier - Proposed Work

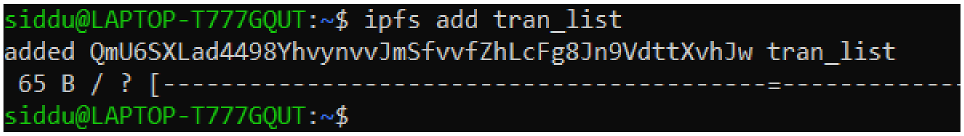
1. The cert-issuer need to modify the issued certificate by changing the Merkle root entry in the certificate.
2. Merkle root entry is replaced with IPFS reference link, where the Merkle root of the certificate is saved in an encrypted format.
3. Steps to encrypt the Merkle root: (It must be done by cert-issuer. For POC, we have done it manually replacing the Merkle root with IPFS reference link).
   a. Follow this for installing IPFS on Ubuntu terminal: https://blog.siderus.io/how-to-get-ipfs-on-ubuntu-debian-linux-d7920c1a42b7
   b. Run IPFS daemon:



```
siddu@LAPTOP-T777GQUT:~$ ipfs daemon
Initializing daemon...
go-ipfs version: 0.4.20-
Repo version: 7
System version: amd64/linux
Golang version: go1.12.4
Swarm listening on /ip4/127.0.0.1/tcp/4001
Swarm listening on /ip4/169.254.192.68/tcp/4001
Swarm listening on /ip4/169.254.212.104/tcp/4001
Swarm listening on /ip4/169.254.235.157/tcp/4001
Swarm listening on /ip4/169.254.72.89/tcp/4001
Swarm listening on /ip4/192.168.2.12/tcp/4001
Swarm listening on /ip4/192.168.217.1/tcp/4001
Swarm listening on /ip4/192.168.226.1/tcp/4001
Swarm listening on /ip6/::1/tcp/4001
Swarm listening on /p2p-circuit
Swarm announcing /ip4/127.0.0.1/tcp/4001
Swarm announcing /ip4/169.254.192.68/tcp/4001
Swarm announcing /ip4/169.254.212.104/tcp/4001
Swarm announcing /ip4/169.254.235.157/tcp/4001
Swarm announcing /ip4/169.254.72.89/tcp/4001
Swarm announcing /ip4/192.168.2.12/tcp/4001
Swarm announcing /ip4/192.168.217.1/tcp/4001
Swarm announcing /ip4/192.168.226.1/tcp/4001
Swarm announcing /ip6/::1/tcp/4001
API server listening on /ip4/127.0.0.1/tcp/5001
WebUI: http://127.0.0.1:5001/webui
Gateway (readonly) server listening on /ip4/127.0.0.1/tcp/8080
Daemon is ready
```

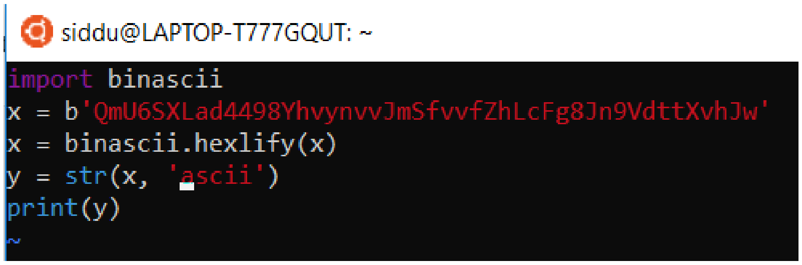
   c. Save Merkle root in a file and encrypt using recipient public key : (Please follow steps guided here to generate public and secure keys: https://medium.com/@mycoralhealth/learn-to-securely-share-files-on-the-blockchain-with-ipfs-219ee47df54c).
   d. When we encrypt using verifier public key, the only verifier will be able to decrypt the file and hence it is secure.

e. When the file containing Merkle root of the certificate is encrypted and uploaded to IPFS, a reference link is returned by IPFS.



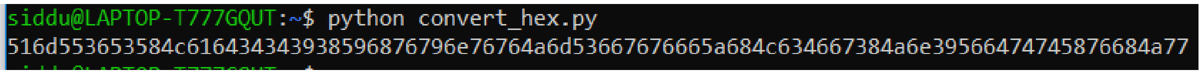
```
siddu@LAPTOP-T777GQUT:~$ ipfs add tran_list
added QmU6SXLad4498YhvynvvJmSfvvfZhLcFg8Jn9VdttXvhJw tran_list
 65 B / ? [-------------------------------------------=------------
siddu@LAPTOP-T777GQUT:~$
```

f. Blockcerts code can only handle hexadecimal values and hence we need to convert the IPFS reference link to the hexadecimal form: (Sample code is given below)



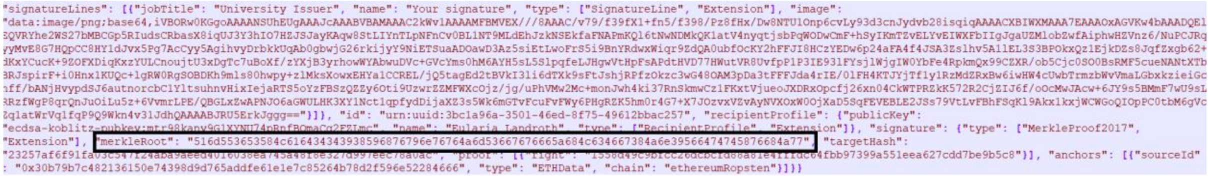
siddu@LAPTOP-T777GQUT: ~

```
import binascii
x = b'QmU6SXLad4498YhvynvvJmSfvvfZhLcFg8Jn9VdttXvhJw'
x = binascii.hexlify(x)
y = str(x, 'ascii')
print(y)
~
```



```
siddu@LAPTOP-T777GQUT:~$ python convert_hex.py
516d553653584c61643434393859687679 6e76764a6d53667676665a684c634667384a6e39566474745876684a77
```

g. Replace hexadecimal formatted IPFS reference link in the certificate :



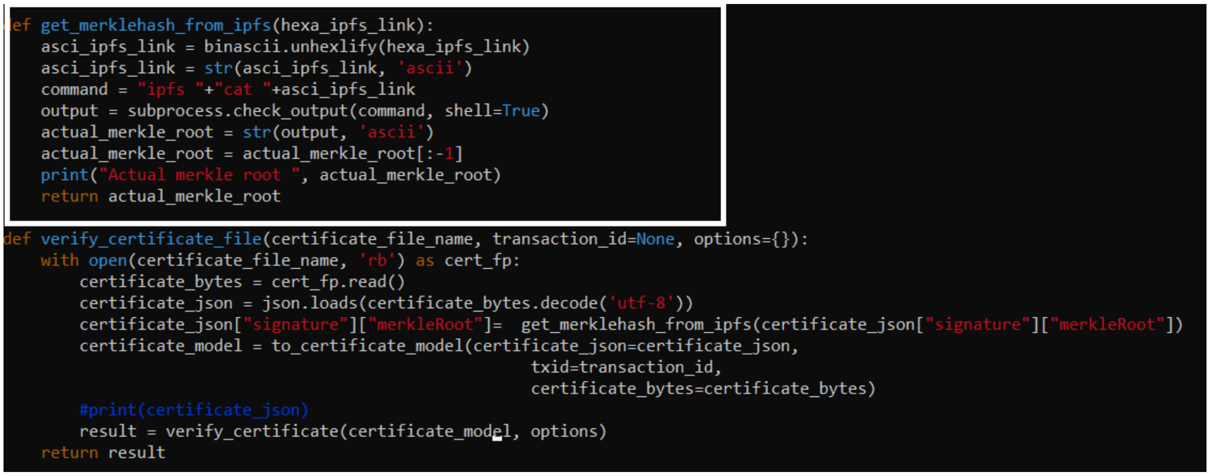
"signatureLines": [{"jobTitle": "University Issuer", "name": "Your signature", "type": ["SignatureLine", "Extension"], "image":
"data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAJcAAABVBAMAAAC2kWv1AAAAMFBMVEX///8AAAC/v79/f39fX1+fn5/f398/Pz8fHx/Dw8NTU1Onp6cvLy93d3cnJydvb28isqiqAAAACXBIWXMAAA7EAAAOxAGVKw4bAAADQEl
EQVRYhe2WS27bMBCGp5RIudsCRbasX8iqUJ3Y3hIO7HZJSJayKAqw8StLIYnTLpNFnCv0BL1NT9MLdEhJzkNSEkfaFNAPmKQl6tNwNDMkQKlatV4nyqtjsbPqWODwCmF+hSyIKmTZvELYvEIWXFbIIgJgaUZMlobZwfAiphwHZVnz6/NuPCJRq
yyMvE8G7HQpCC8HY1dJvx5Pg7AcCyy5AgihvyDrbkkUqAb0gbwjG26rkijyY9NiETSuaADOawD3Az5siEtLwoFrS5i9BnYRdwxWiqr9ZdQA0ubfOcKY2hFFJI8HCzYEDw6p24aFA4f4JSA3Zslhv5A1lEL3S3BPOkxQz1EjkDZs8JqfZxgb62+
dKxYCucK+9ZOFXDiqKxzYULCnoujtU3xDgTc7uBoXf/zYXjB3yrhowWYAbwuDVc+GVcYms0hM6AYH5sL5SlpqfeLJHgwVtHpFsAPdtHVD77HWutVR8UvfpP1P3IE93lFYsjlWjgIW0YbFe4RpkmQx99CZXR/ob5Cjc0SO0BsRMF5cueNANtXTb
BRJspirF+i0Hnx1KUQc+lgRW0RgSOBDKh9mls80hwpy+zlMksXowxEHYa1CCREL/jQ5tagEd2tBVkI3li6dTXk9sFtJshjRPfzOkzc3wG48OAM3pDa3tFFFJda4rIE/0lFH4KTJYjTfly1RzMdZRxBw6iwHW4cUwbTrmzbWvVmaLGbxkzieiGc
nff/bANjHvypdSJ6autnorcbC1YltsuhnvHixIejaRTS5oYzFBSzQZZy6Oti9UzwrZZMFWXcOjz/jg/uPhVMw2Mc+monJwh4ki37RnSkmwCz1FKxtVjueoJXDRxOpcfj26xn04CkWTPRZkK572R2CjZIJ6f/oOcMwJAcw+6JY9s5BMmF7wU9sL
RRzfWgP8qrQnJuOiLu5z+6VvmrLPE/QBGLxZwAPNJO6aGWULHK3XY1Nct1qpfydDijaXZ3s5Wk6mGTvFcuFvFWy6PHgRZK5hm0r4G7+X7JOzvxVZvAyNVXOxW0OjXaD5SqFEVEBLE2JSs79VtLvFBhFSqKl9Akx1kxjWCWGoQIOpPC0tbM6gVc
ZqlatWrVq1fqP9Q9Wkn4v31JdhQAAAABJRU5ErkJggg=="}]], "id": "urn:uuid:3bc1a96a-3501-46ed-8f75-49612bbac257", "recipientProfile": {"publicKey":
"ecdsa-koblitz-pubkey:mtr98kany9G1XYNU74pRnfBOmaCq2FZLmc", "name": "Eularia Landroth", "type": ["RecipientProfile", "Extension"]}, "signature": {"type": ["MerkleProof2017",
"Extension"], "merkleRoot": "516d553653584c61643434393859687679 6e76764a6d53667676665a684c634667384a6e39566474745876684a77", "targetHash":
"23257af6f91fa03c547f24aba9aeed4016038ea745a4818e327d997eec78a0ac", "proof": [{"right": "1558d49c9bfcc26dcbcfd88a81e4ffffdc64fbb97399a551eea627cdd7be9b5c8"}], "anchors": [{"sourceId"
: "0x30b79b7c482136150e74398d9d765addfe61e1e7c85264b78d2f596e52284666", "type": "ETHData", "chain": "ethereumRopsten"}]}}

h. Issue the modified certificate to the student.

i. At the verifier side, verifier code needs to fetch the actual Merkle root from IPFS.

j. To fetch the actual hash of the Merkle root, verifier needs to convert the hexadecimal IPFS link to actual IPFS ink.

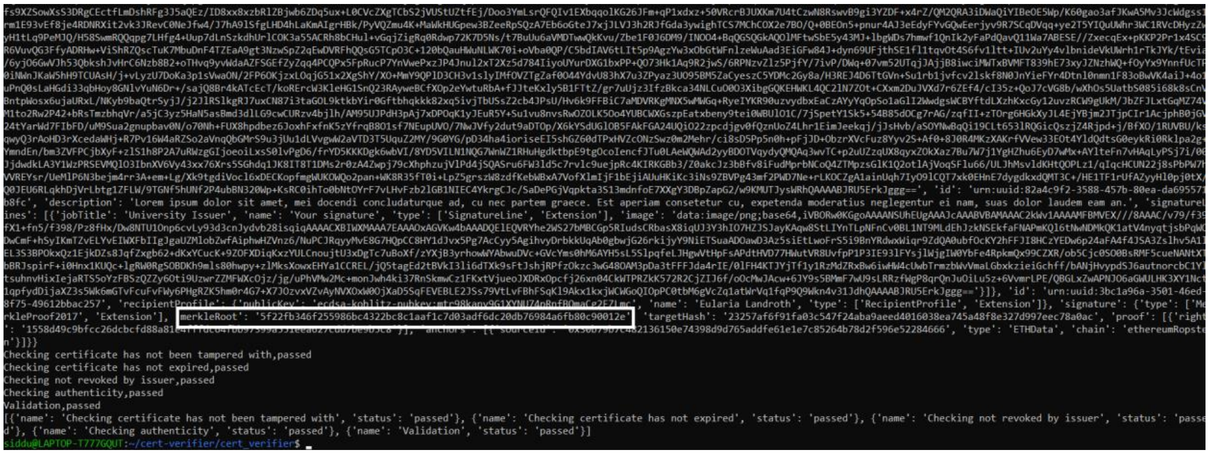
k. Using actual IPFS link, need to fetch the actual hash of the Merkle root. (**<u>Main code: verifier.py)</u>**



```
ef get_merklehash_from_ipfs(hexa_ipfs_link):
    asci_ipfs_link = binascii.unhexlify(hexa_ipfs_link)
    asci_ipfs_link = str(asci_ipfs_link, 'ascii')
    command = "ipfs "+"cat "+asci_ipfs_link
    output = subprocess.check_output(command, shell=True)
    actual_merkle_root = str(output, 'ascii')
    actual_merkle_root = actual_merkle_root[:-1]
    print("Actual merkle root ", actual_merkle_root)
    return actual_merkle_root

def verify_certificate_file(certificate_file_name, transaction_id=None, options={}):
    with open(certificate_file_name, 'rb') as cert_fp:
        certificate_bytes = cert_fp.read()
        certificate_json = json.loads(certificate_bytes.decode('utf-8'))
        certificate_json["signature"]["merkleRoot"]=  get_merklehash_from_ipfs(certificate_json["signature"]["merkleRoot"])
        certificate_model = to_certificate_model(certificate_json=certificate_json,
                                                       txid=transaction_id,
                                                       certificate_bytes=certificate_bytes)

        #print(certificate_json)
        result = verify_certificate(certificate_model, options)
    return result
```

l. With the actual hash of the Merkle root inserted, the verification process will be executed, and the result should be a pass.

m. Screen shot of the output :



```
fs9XZSowXsS3DRgCEctfLmDshRFg3J5aQEz/ID8xx8xzbR1ZBjwb6ZDq5ux+L0CVcZXgTCbS2jVUStUZtfEj/Doo3YmLsrQFQIv1EXbqqolKG26JFm+qP1xdxz+50VRcrBJUXKm7U4tCzwN8RswvB9gi3YZDF+x4rZ/QM2QRA3iDWaQiYIBeOE5Wp/K60gao3afJKwA5Mv3JcWdgss1
rm1E93vEf8je4RDNRXit2vk3JRevC0NeJfw4/J7hA91SfgLHD4hLaKmAIgrHBk/PyVQZmu4K+MaWkHUGpew3BZeeRpSQzA7Eb6oGteJ7xjJLVJ3h2RJfGda3ywighTCS7MChCOX2e7BO/Q+0BEOn5+pnur4AJ3eEdyFYvGQwEerjyv9R7SCqDVqq+ye2T5YIQuUWhr3WC1RVcDHyzZw
yH1tLq9PeMJQ/H58SwmRQQqpg7LHfg4+Uup7dLnSzkdhUrlCOK3a55ACRh8bCHul+vGqjZigRq0Rdwp72K7D5Ns/t7BuUu6aVMDTwwQkKvu/Zbe1F0J6DM9/INOO4+BqQGSQGkAQOlMFtwSbE5y43MJ+lbgWDs7hmwf1QnIk2yFaPdQavQ11Wa7ABESE//ZxecqEx+pKKP2Pr1x4SC9
R6VuvQG3FfyADRHw+ViShRZQscTuK7MbuDnF4TZEaA9gt3NzwSpZ2qEwDVRFhQQsG5TCpO3C+120bQauHWuNLWK70i+oVba0QP/C5bdIAV6tLIt5p9AgzYw3xObGtWFnlzeWuAad3EiGFw84J+dyn69UFjthSE1fl1tqvOt4S6fv1ltt+IUv2uYy4vlbnideVkUWrh1rTkJYk/tEvia
/6yjO6GwVJh53QbkshJvHrC6Nzb8B2+oTHvq9yvWdaAZFSGEfZyZqq4PCQPx5FpRucP7YnVwePxzJP4Jnul2xT2Xz5d784IiyoUYurDXG1bxPP+QO73Hk1Aq9R2jwS/6RPNzvZlz5PjfY/7ivP/DWq+07vm52UTqjJAjjB8iwciMWTxBVMFT839hE73xyJZNzhWQ+fOyYx9YnnfUcTF
0iNWnJKaW5hH9TCUAsH/j+vLyzU7DoKa3p1sVwaON/2FP6OKjzxLOqjG51x2XgShY/XO+MmY9QPlD3CH3v1slyIMfOVZTgZaf0O44YdvU83hX7u3ZPyaz3UO95BM5ZaCyeszC5YDMc2Gy8a/H3REJ4D6TtGVn+Su1rb1jvfcv2lskf8N0JnYieFYr4Dtnl0nmn1F83oBwVK4aiJ+4o1
uPnQ0sLaHGdi33qbHoy8GNlvYuN6Dr+/sajQ8Br4kATcEcT/koRErcW3KleHG1SnQ23RAyweBCfXOp2eYwtuRbA+fJJteKxly5B1FTtZ/gr7uUjz3IfzBkca34NLCuO0O3XibgGQKEHWKL4QC2lN7ZOt+CXxm2DuJVXd7r6ZEf4/cI35z+QoJ7cVG8b/wXhOs5UatbS085i68k8sCnV
BntpWosx6ujaURxL/NKyb9baQtrSyjJ/j2JlRSlkgRJ7uxCN87i3taGOL9ktkbYir0Gftbhqkkk82xq5ivjTbUSsZ2cb4JPsU/Hv6k9FFBiC7aMDVRKgMNX5wMWGq+RyeIYKR90uzvydbxEaCzAYyYqOpSo1aGlI2WwdgsWCBYftdLXzhKxcGy12uvzRCW9gUkM/JbZFJLxtGqMZ74V
M1to2Rw2P42+bRsTmzbhqVr/a5jC3yz5HaN5asBmd3dlLG9cwCURzv4bjlh/AM95UJPdH3pAj7xDPOqK1yJEuR5Y+Su1vu8nvsRwOZOLK5Oo4YUBCWXGszpEatxbeny9tei0WBUlO1C/7jSpetY1Sk5+54B85dOCg7rAG/zqfII+zTOrg6HGkXyJL4EjYBjm2JTjpCIr1AcjphB0jGV
24tYarWd7FIbFD/uM9Sua2gnupbav0N/o70Nh+FUX8hpdbez6JoxhFxfnK5zYfrqB8O1sf7NEupUVO/7NwJVfy2dut9aDTOp/X6kYSdUGlOB5FAkFGA24UQiO22zpcdjgv0fQznUoZ4Lhr1EimJeekqj/jJsHvb/aSOYNwBqQi19CLt653lRQGicQszjZ4Rjpd+j/BfXO/1RUVBU/ks
qwyQ3rAoHD3rXcedaWHj+R7Pv16W4aRZSo2aVnqQbGMrS9u3jUu1dLVvgwW2aVTD3T5UquZ2MY/9G0YG/pD34ha4ioriseEI5shGZ60dTPxHVZcONzSwz0m2Mehr/ci8sD5Pp5n0h+pFjJD+ObzrXVcFuz8Yyv2S+Af0+8J0R4MKzXAKrfVVew33EOt4YldQdtsG0eykRi0Rklpa2g+
YmndEn/bm3ZVFPCjbXyF+zlS1h8P2A7uRWzgGIjoeoiLxsS0lvPgD6/frYD5KKKDgk6wbVI/8YD5VILN1NQG7WnWZ1RHuHgdktbpE9tgOcoIencfJTu0LAeWQWAd2yyBDOTVqydyQMQAq3wvTC+p2uUZzqUX8qyxZOkXaz7Bu7W7j1YgHZhu6EyD7wMx+AYlteFn7vHAqLyPSj7i/06
JjdwdkLA3Y1WzPRSEVMQlO3IbnXV6Vy43xx76Xrs5SGhdq1JK8IT8T1DMs2r0zA4Zwpj79cXhphzujVlPd4jSQASru6FW3ld5c7rvlc9uejpRc4KIRKGBb3/Z0akcJz3bBfv8iFudMprbNCoQ4ZTMpzsGlK1Q2otlAjVoqSFlu66/ULJhMsvldKHtQOPLz1/qIqcHCUN22j8sPbPW7H
VVREYsr/UeMlP6N3bejm4rr3A+em+Lg/Xk9tgdiVocl6xDECKopfmgWUKOWQo2pan+WK8R35fT0i+LpZ5grszW8zdfKebWBxA7VofXlmIjF1bEjiAUuHKiKc3iNs9ZBVPg43mf2PWD7Ne+rLKOCZgA1ainUqh7IyO9lCQT7xk0EHnE7dygdkxdQMT3C+/HE1TF1rUfAZyyHl0pj0tX/
Q0JEU6RLqkhDjVrLbtg1ZFLW/9TGNf5hUNf2P4ubBN320Wp+KsRC0ihTo0bNtOYrF7vLHvFzb2lGB1NIEC4YkrgCJc/SaDePGjVqpkta3S13mdnfoE7XXgY3DBpZapG2/w9KMUTJysWRhQAAAABJRU5ErkJggg==', 'id': 'urn:uuid:82a4c9f2-3588-457b-80ea-da695571
b8fc', 'description': 'Lorem ipsum dolor sit amet, mei docendi concludaturque ad, cu nec partem graece. Est aperiam consetetur cu, expetenda moderatius neglegentur ei nam, suas dolor laudem eam an.', 'signatureL
ines': [{'jobTitle': 'University Issuer', 'name': 'Your signature', 'type': ['SignatureLine', 'Extension'], 'image': 'data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAJcAAABVBAMAAAC2kWv1AAAAMFBMVEX///8AAAC/v79/f39
fX1+fn5/f398/Pz8fHx/Dw8NTU1Onp6cvLy93d3cnJydvb28isqiqAAAACXBIWXMAAA7EAAAOxAGVKw4bAAADQElEQVRYhe2WS27bMBCGp5RIudsCRbasX8iqUJ3Y3hIO7HZJSJayKAqw8StLIYnTLpNFnCv0BL1NT9MLdEhJzkNSEkfaFNAPmKQl6tNwNDMkQK1atV4nyqtjsbPqWC
DwCmF+hSyIKmTZvELYvEIWXFbIIgJgaUZMlobZwfAiphwHZVnz6/NuPCJRqyyMvE8G7HQpCC8HY1dJvx5Pg7AcCyy5AgihvyDrbkkUqAb0gbwjG26rkijyY9NiETSuaADOawD3Az5siEtLwoFrS5i9BnYRdwxWiqr9ZdQA0ubfOcKY2hFFJI8HCzYEDw6p24aFA4f4JSA3Zslhv5A1I
EL3S3BPOkxQz1EjkDZs8JqfZxgb62+dKxYCucK+9ZOFXDiqKxzYULCnoujtU3xDgTc7uBoXf/zYXjB3yrhowWYAbwuDVc+GVcYms0hM6AYH5sL5SlpqfeLJHgwVtHpFsAPdtHVD77HWutVR8UvfpP1P3IE93lFYsjlWjgIW0YbFe4RpkmQx99CZXR/ob5Cjc0SO0BsRMF5cueNANtXT
bBRJspirF+i0Hnx1KUQc+lgRW0RgSOBDKh9mls80hwpy+zlMksXowxEHYa1CCREL/jQ5tagEd2tBVkI3li6dTXk9sFtJshjRPfzOkzc3wG48OAM3pDa3tFFFJda4rIE/0lFH4KTJYjTf1y1RzMdZRxBw6iwHW4cUwbTrmzbWvVmaLGbxkzieiGchff/bANjHvypdSJ6autnorcbC1Y1
tsuhnvHixIejaRTS5oYzFBSzQZZy6Oti9UzwrZZMFWXcOjz/jg/uPhVMw2Mc+monJwh4ki37RnSkmwCz1FKxtVjueoJXDRxOpcfj26xn04CkWTPRZkK572R2CjZIJ6f/oOcMwJAcw+6JY9s5BMmF7wU9sLRRzfWgP8qrQnJuOiLu5z+6VvmrLPE/QBGLxZwAPNJO6aGWULHK3XY1Nct
1qpfydDijaXZ3s5Wk6mGTvFcuFvFWy6PHgRZK5hm0r4G7+X7JOzvxVZvAyNVXOxW0OjXaD5SqFEVEBLE2JSs79VtLvFBhFSqKl9Akx1kxjWCWGoQIOpPC0tbM6gVcZq1atWrVq1fqP9Q9Wkn4v31JdhQAAAABJRU5ErkJggg=='}]}, 'id': 'urn:uuid:3bc1a96a-3501-46ed-
8f75-49612bbac257', 'recipientProfile': {'publicKey': 'ecdsa-koblitz-pubkey:mtr98kanv9G1XYNU74pRnfBQmaCe2EZLmc', 'name': 'Eularia Landroth', 'type': ['RecipientProfile', 'Extension']}, 'signature': {'type': ['Me
rkleProof2017', 'Extension'],  merkleRoot': '5f22fb346f255986bc4322bc8c1aaf1c7d03adf6dc20db76984a6fb80c90012e'  'targetHash': '23257af6f91fa03c547f24aba9aeed4016038ea745a48f8e327d997eec78a0ac', 'proof': [{'right
': '1558d49c9bfcc26dcbcfd88a81e4fffdc04fbb97399a331eea027cdd7be9b5c8'}], 'anchors': [{'sourceId': '0x30b79b7c482136150e74398d9d765addfe61e1e7c85264b78d2f596e52284666', 'type': 'ETHData', 'chain': 'ethereumRopste
n'}]}}
Checking certificate has not been tampered with,passed
Checking certificate has not expired,passed
Checking not revoked by issuer,passed
Checking authenticity,passed
Validation,passed
[{'name': 'Checking certificate has not been tampered with', 'status': 'passed'}, {'name': 'Checking certificate has not expired', 'status': 'passed'}, {'name': 'Checking not revoked by issuer', 'status': 'passe
d'}, {'name': 'Checking authenticity', 'status': 'passed'}, {'name': 'Validation', 'status': 'passed'}]
siddu@LAPTOP-T777GQUT:~/cert-verifier/cert_verifier$ _
```

n. We can push all confidential information which is kept in the certificate(ex: Merkle root and transaction ID) to IPFS and let the verifier fetch the details from IPFS securely.

o. We will be sharing verifier.py (the only changed file and need to be replaced with actual verifier.py from the repository.

Reference:

- https://xiaoxing.us/2018/01/30/utilizing-blockcert-blockchain-based-educational-certificates/