NSC_Assignment_4

Project - 2

Submitted by - Ekansh (2021044) and Lakshay Bansal (2021059)

Introduction to Assignment -

In this assignment, we are required to -

- Develop an On-the-go verification of Driver's License protocol
- We had multiple RTOs and One National TO -
    1. Clients (Traffic Police Officer) can verify Validity of a License using digital signal verification
    2. Digital Signature ensures Non-Repudiation, Message Integrity and Authenticity.

Working Schema -

RSA -

We have created a RSA class, that will be used to -

Generating Key Pairs -
In this function, we will generate the keys by taking p and q as random prime numbers then generating, n, ϕ and d.

```
def generate_prime(self, n_bits=1024):
    while True:
        prime_candidate = random.getrandbits(n_bits)
        if isprime(prime_candidate): return prime_candidate

def generate_rsa_keys(self, n_bits=1024):
    p = self.generate_prime(n_bits)
    q = self.generate_prime(n_bits)
    n = p * q
    phi = (p-1) * (q-1)
    e = 65537
    d = mod_inverse(e, phi)
    return ((e, n), (d, n))
```

Encrypting the message -
In this function, we will take the message and the public key as parameters and then convert the message first into integer and then used it to calculate the *encrypted_int* = *(block_int^e)%n.*

```
def rsa_encrypt(self, message, public_key):
    e, n = public_key
    encrypted_blocks = []
    block_size = 245  # Adjust block size based on padding scheme
    for i in range(0, len(message), block_size):
        block = message[i:i+block_size]
        block_int = int.from_bytes(block.encode(), 'big')
        encrypted_int = pow(block_int, e, n)
        encrypted_blocks.append(encrypted_int)
    return encrypted_blocks
```

Decrypting the message -
In this function, we will take the private_key and encrypted message as input, then we will find the *decrypted_int* = *(encrypted_int^d)%n* and the convert the integer into text format.

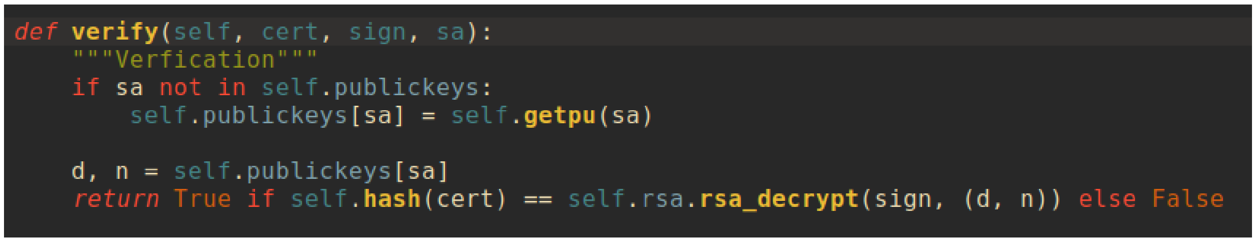
```
def rsa_decrypt(self, encrypted_blocks, private_key):
    d, n = private_key
    decrypted_blocks = []
    for encrypted_int in eval(encrypted_blocks):
        decrypted_int = pow(int(encrypted_int), d, n)
        decrypted_block = 
decrypted_int.to_bytes((decrypted_int.bit_length() + 7) // 8, 
'big').decode()
        decrypted_blocks.append(decrypted_block)
    return ''.join(decrypted_blocks)
```

TransportAuthority -

We have created a TransportAuthority class which has two functions that are mainly used for the overall functionality of it.

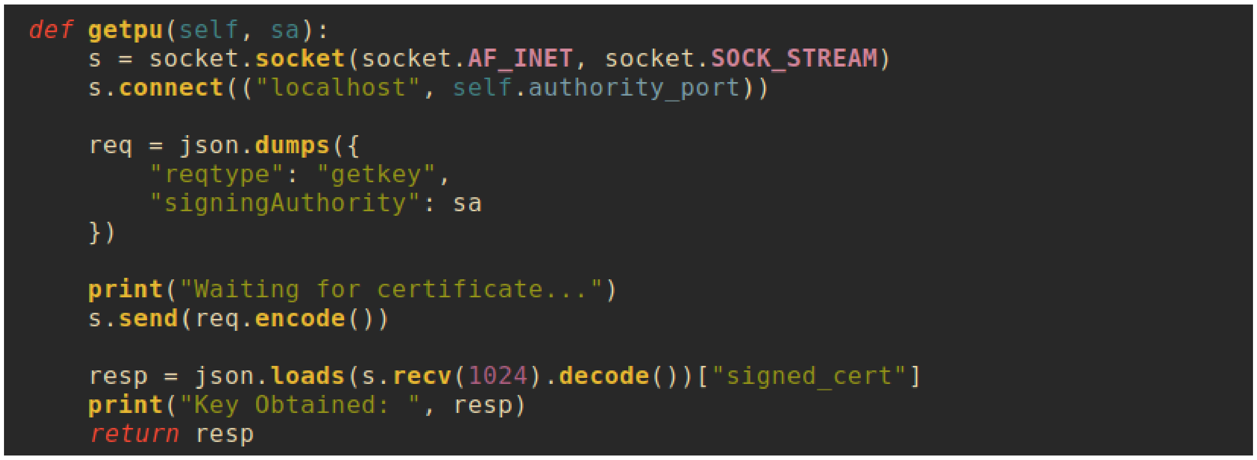
Verify License -

It is used to connect to the client using the port and the host ID, then it will verify the license obtained by the client.



```python
def verify(self, cert, sign, sa):
    """Verfication"""
    if sa not in self.publickeys:
        self.publickeys[sa] = self.getpu(sa)

    d, n = self.publickeys[sa]
    return True if self.hash(cert) == self.rsa.rsa_decrypt(sign, (d, n)) else False
```

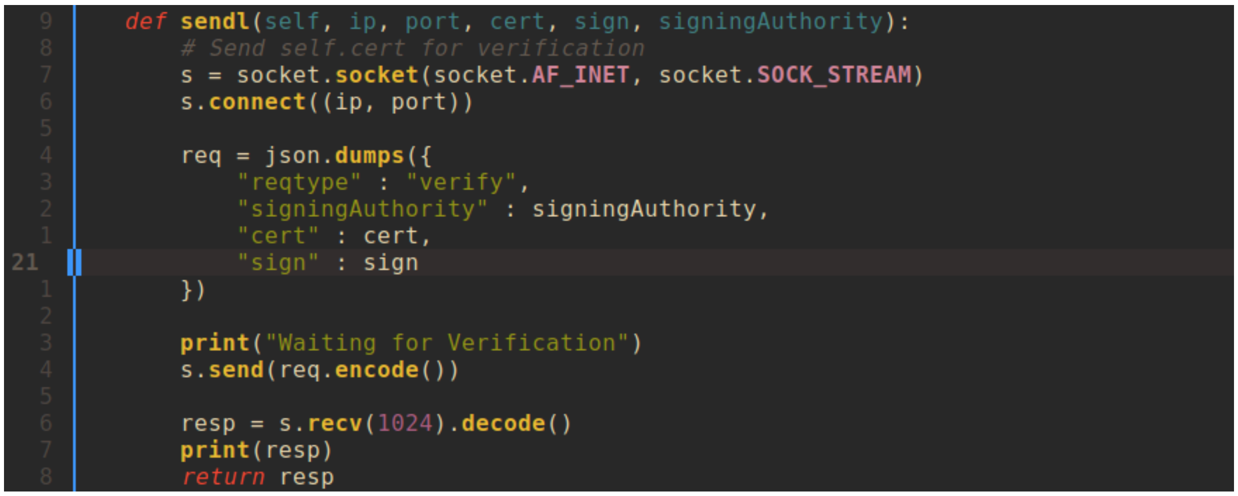
Obtain PublicKey of Regional RTO -
In case the license obtained by the RTO is not signed by itself it will ask National TO to provide its public key of the RTO which signed the cert to verify it .



```python
def getpu(self, sa):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect(("localhost", self.authority_port))

    req = json.dumps({
        "reqtype": "getkey",
        "signingAuthority": sa
    })

    print("Waiting for certificate...")
    s.send(req.encode())

    resp = json.loads(s.recv(1024).decode())["signed_cert"]
    print("Key Obtained: ", resp)
    return resp
```

Client -

We have created a Client class which have two functionalities -
Get Document Verified - Client can request any of the RTO to verify the license it received.



```python
    def sendl(self, ip, port, cert, sign, signingAuthority):
        # Send self.cert for verification
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect((ip, port))

        req = json.dumps({
            "reqtype" : "verify",
            "signingAuthority" : signingAuthority,
            "cert" : cert,
            "sign" : sign
        })

        print("Waiting for Verification")
        s.send(req.encode())

        resp = s.recv(1024).decode()
        print(resp)
        return resp
```

Running the code -

To run the code, you need to keep all the files in a folder, then run ***Client.py*** using *python* **TransportAuthority.py** in at least two different terminals (one for National TO and others for Regional TO) and ***SignCert.py*** using *python* another termina to get signed certificates.

Solution to Problems-

What is the information to be supplied by the driver to the police officer? And what information is sought and obtained by the police officer from the server in the transport authority?
The driver needs to show his/her license which was signed by an established RTO. The police officer then verifies the license using the public key of the RTO which signed that document to verify the identity. It does not matter even if the Driver and the Police are from different regions.

Would you need a central server that has the correct and complete information on all drivers and the licenses issued to them?
No, that information is not required on the central server as regional RTOs can themselves verify the license even if it is not signed by the same RTO. The central server job is to provide Certificates of the requested RTOs only.

In what way are digital signatures relevant?
Digital signature helps us in verifying the license provided by the driver. Digital signature ensures three properties that are Non-repudiation, Message Integrity and Authenticity all of which are required to verify the license. Confidentiality is not our concern in any case.

Does one need to ensure that information is kept confidential? Or not altered during 2-way communication?
No one need not care about confidentiality of messages in such scenarios as confidentiality is not our concern. We only care about the validity of the document provided for which Message Integrity, and Authenticity are sufficient. However information need not to be altered.

Which of these, viz. confidentiality, authentication, integrity and non-repudiation is/are relevant?
Among these confidentiality is not relevant. However Authenticity, Integrity and non-repudiation are relevant. Here digital signatures help to achieve the same.