MT20062 Shivani Mishra
MT20066 Ravi Rathee

Network Security CSE 350/550
Project Report (On-the-go verification of Driver’s License)

Server Side Code :

```
import socket
from Crypto.PublicKey import RSA
from Crypto import Random
from Crypto.Cipher import PKCS1_OAEP
from licensing.models import *
from licensing.methods import Key, Helpers, Message, Product, Customer
import hashlib

#Generate private and public keys
random_generator = Random.new().read
keys = RSA.generate(1024, random_generator)
decryptor = PKCS1_OAEP.new(keys)
public_key = keys.publickey()

# Our 1 variable database
message = "Ramesh 2344567890 GJ05-1234" #Data stored in database
result = hashlib.sha256(message.encode()) #Data stored in database


# encryptor = PKCS1_OAEP.new(public_key)

# message = "This is my license"
# encrypted = encryptor.encrypt(message.encode('utf-8'))
# decrypted = decryptor.decrypt(encrypted)
# print(str(decrypted, encoding='utf8'))

#Declartion
mysocket = socket.socket()
host = socket.gethostbyname(socket.getfqdn())
port = 8080
encrypt_str = "encrypted_message="

print("host = " + host)

#Prevent socket.error: [Errno 98] Address already in use
mysocket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

mysocket.bind((host, port))

mysocket.listen(5)

c, addr = mysocket.accept()
#Wait until data is received.
################ 1st Recv ###################
temp = c.recv(1024)
data = str(temp, encoding='utf8')
data = data.replace("\r\n", '') #remove new line character
c.send(public_key.exportKey())
print("Public key sent to client.")

################ 2nd Recv ##################
temp = c.recv(1024)
decrypted = decryptor.decrypt(temp)
hash1 =  str(decrypted, encoding='utf8')
hash2 =  result.hexdigest()

if hash1 == hash2:
    c.send(bytes("Server: This person has valid license", encoding='utf8'))
else:
    c.send(bytes("Server: Invalid : Catch him/her", encoding='utf8'))


################ 3rd Recv ###################
temp = c.recv(1024)

# while True:

#     #Wait until data is received.
#     temp = c.recv(1024)
#     data = str(temp, encoding='utf8')
#     data = data.replace("\r\n", '') #remove new line character

#     if data == "Client: OK":
#         #c.send(bytes("public_key=" + str(public_key.exportKey(), encoding='utf8') + "\n", encoding='utf8'))
#         # c.send(bytes("public_key=" + str(public_key, encoding='utf8') + "\n", encoding='utf8'))
#         c.send(public_key.exportKey())
#         print(public_key.export_key())
#         print("Public key sent to client.")

#     elif encrypt_str in data: #Reveive encrypted message and decrypt it.
#         #data = data.replace(encrypt_str, '')
#         #print("Received:\nEncrypted message = "+str(data))
#         print(data)
#         decrypted = decryptor.decrypt(temp)
#         c.send(bytes("Server: OK", encoding='utf8'))
#         print("Decrypted message = " + decrypted)

#     elif data == "Quit": break

#Server to stop
c.send(bytes("Server stopped\n", encoding='utf8'))
print("Server stopped")
c.close()
```

Client Side Code

```
import socket
from Crypto.PublicKey import RSA
import Crypto
from Crypto.Cipher import PKCS1_OAEP
from licensing.models import *
from licensing.methods import Key, Helpers, Message, Product, Customer
import hashlib

server = socket.socket()
host = "127.0.1.1"
port = 8080

server.connect((host, port))

#Tell server that connection is OK
s = bytes('Client: OK', encoding = 'utf8')
server.sendall(s)

#Receive public key string from server
temp = server.recv(1024)
key = RSA.import_key(temp)
encryptor = PKCS1_OAEP.new(key)

#Encrypt message and send to server
message = "Ramesh 2344567890 GJ05-1234" #Data taken from scanner
result = hashlib.sha256(message.encode())
encrypted = encryptor.encrypt(result.hexdigest().encode('utf-8'))
server.sendall(encrypted)
```

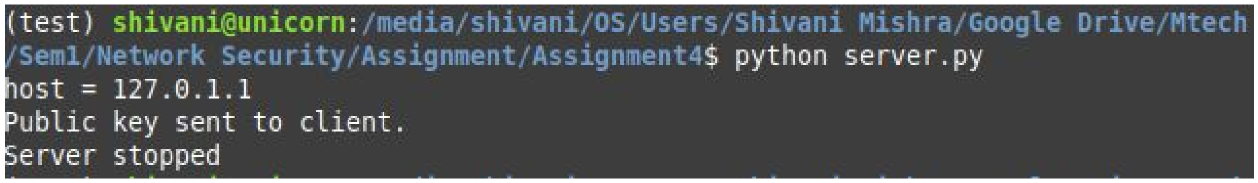
```
#Server's response
temp = server.recv(1024)
server_response = str(temp, encoding='utf8')

server_response = server_response.replace("\r\n", '')
print(server_response)

#Tell server to finish connection
server.sendall(bytes("Quit", encoding='utf8'))
lastwords = server.recv(1024) #Quit server response
print(str(lastwords, encoding='utf8'))
server.close()
```

Explanation :

We have used RSA Algorithm in our client and server side code. We have used hash function for finding the hash value of the message (SHA hash function) then server send the public key to client and client will send the data containing name, phone number and license number to the server. Server side, compare hash value with the hash value stored in the database if that matches server send the message that “User will have valid license” else “Invalid”.

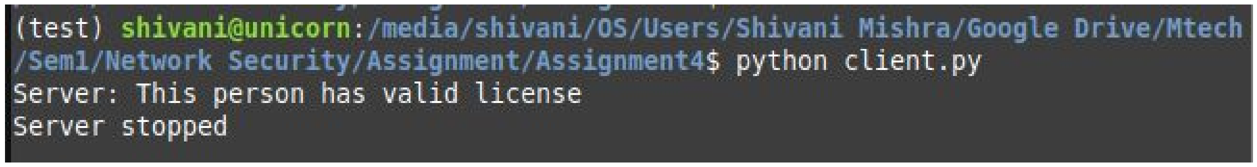
Outputs :

Server (Message stored in the database : "Ramesh 2344567890 GJ05-1234")



```
(test) shivani@unicorn:/media/shivani/OS/Users/Shivani Mishra/Google Drive/Mtech
/Sem1/Network Security/Assignment/Assignment4$ python server.py
host = 127.0.1.1
Public key sent to client.
Server stopped
```

Client (Message sent to server : "Ramesh 2344567890 GJ05-1234")



```
(test) shivani@unicorn:/media/shivani/OS/Users/Shivani Mishra/Google Drive/Mtech
/Sem1/Network Security/Assignment/Assignment4$ python client.py
Server: This person has valid license
Server stopped
```

Question / Answer

1. 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 transport authority?

Ans : We will send the pdf file containing the license number of the owner of vehicle. He will pass on his license card to the police officer. Police officer will type the number manually and file will be converted to pdf. We are just sending a **pdf** file from client to server.

2. Would you need a central server that has the correct and complete information on all drivers and the licenses issued to them?

Ans : A centralized server or many small server located in different locations would be more beneficial as most people buy cars from the same city in which they plan to use the vehicle. So a distributed system containing all license numbers would make the job efficient, but for backup we can keep data in a centralized server. If any data is lost it can be fetched from the centralized server/servers.

3. Is date and time of communication important?

Ans : Yes, date and time of communication is important as drivers license may get expired by the time police enquires about his license number. So the date and time of communication becomes of extreme importance as many can be falsely challaned or many can get use this loop hole to forge fake licenses.

4. In what way are digital signatures relevant?

Ans : Digital signatures have a very wide range of applications. It makes sure authenticity is always maintained on our servers as well as it makes sure that the party which is in charge of the system is not able to forge fake licenses not only for themselves but also for other people. Having a robust and reliable system, helps in keeping trust of users on the system and scales well for a lot of people in use.

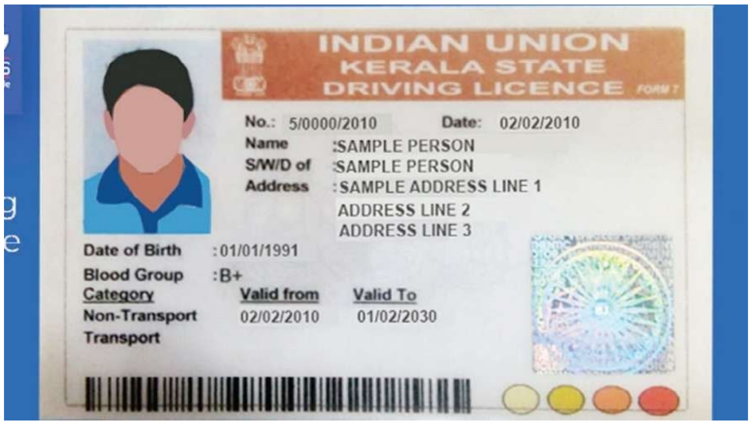
5. Does one need to ensure that information is kept confidential? Or not altered during 2-way communication?

Ans : Information may or may not be kept confidential that is an altogether different issue. Here we are concerned with keeping the authenticity of our message. We just want to check whether the license holder is having a genuine license number and if that is true our target is fulfilled. But definitely information security is also very important but for our project its out of scope.

6. Which of these, viz. confidentiality, authentication, integrity and non-repudiation are relevant?

Ans : For our project authenticity and integrity is of more importance here. But definitely all of them are important when we to design a system which is going to be used by millions of people on a national scale.

Sample of a license number show to a police officer.



INDIAN UNION
KERALA STATE
DRIVING LICENCE FORM 7

No.: 5/0000/2010 Date: 02/02/2010
Name :SAMPLE PERSON
S/W/D of :SAMPLE PERSON
Address :SAMPLE ADDRESS LINE 1
ADDRESS LINE 2
ADDRESS LINE 3

Date of Birth :01/01/1991
Blood Group :B+

Category	Valid from	Valid To
Non-Transport	02/02/2010	01/02/2030
Transport		