Project Report:
On-the-go verification of Driver’s License

Abhishek 2020014
Rishabh Jain 2020399

1. Introduction:

This project focuses on enhancing the verification process of driver's licenses through technology. By employing cryptographic techniques such as RSA encryption and Diffie-Hellman key exchange, we aim to secure the exchange of messages between police devices(PDs) and a centralized transport authority database(TA), ensuring authenticity and integrity on the go.

2. Functions:

We made 4 Python files named “database.py(for TA database)” and “device.py(for PDs)” and two helper files, “rfid_generators.py” and “rsa.py”(we implemented the whole RSA algorithm). Now, the important functions are:

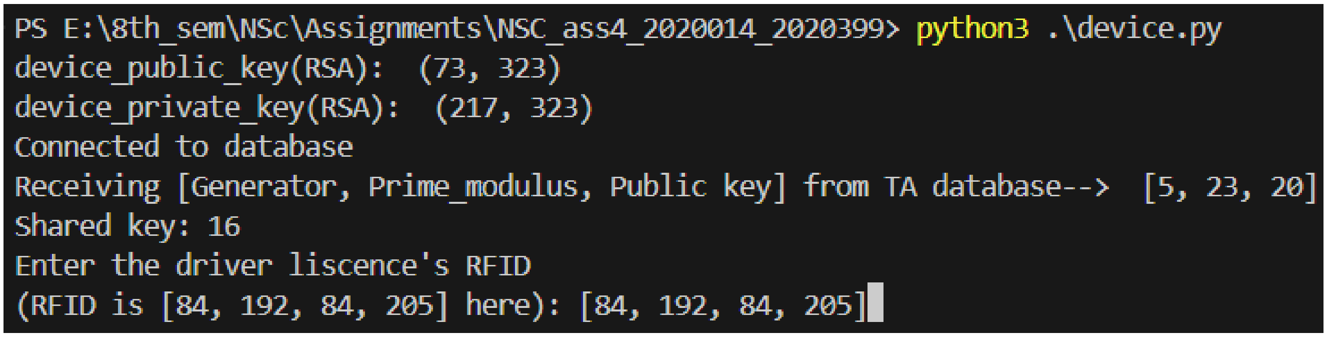
- **Encrypt function-** Inputs are plaintext messages and the private key for RSA encryption. It iterates over each character in the message and computes its modular exponentiation with the encryption exponent (e) and modulus (n). Finally, the function returns the list of encrypted characters representing the ciphertext.
- **Decrypt function-** Inputs are encrypted messages and the public key for RSA decryption. It iterates over each element (character) in the encrypted message list. For each element, it computes the modular exponentiation with the decryption exponent (d) and modulus (n).
  Finally, the function returns the decrypted plaintext.
- **Power(modular exponentiation)-** This function uses a binary exponentiation algorithm and efficiently computes the result of raising a number “a” to the power of “b” modulo “c.”
- **Deffie_Hellman Public Key(generate_public_key)-** Inputs are private key, prime modulus, and generator. It calls the power function to raise the generator to the power of the private key and modulo the prime modulus. As a result, it generates the public key for the Deffie-Hellman key exchange protocol.
- **Shared Session Key(compute_shared_key)-** Inputs are public key, private key, and prime modulus. It calls the power function to raise the received public key to the power of the private key and modulo the prime modulus. As a result, it returns the computed Shared key for the Deffie-Hellman key exchange protocol.

- **Send_message-** Encrypts the message using the RSA algorithm with the provided private key and sends it over the socket connection. Facilitates secure transmission of messages over the network by encrypting them before sending.
- **Recieve_message-** Receives encrypted data over the socket connection, decrypts it using RSA decryption with the provided public key, and returns the decrypted message. Ensures secure reception of messages by decrypting them before processing
- **Connect_to_port-** Establishes a socket connection to a specified host and port, binding to a specified port on the local machine. Enables communication between client and server components of the system, ensuring connectivity for message exchange.

3. Questions addressed:

1) Driver provides basic details along with Driver's license, and the important information is on the license itself, i.e. Rfid. The Police Officer obtained the Validation of the license from the server database.
2) Yes, a centralized transport authority database is required to store and retrieve information about all drivers and their licenses.
3) Digital signatures are relevant for ensuring the authenticity and integrity of the exchanged messages, which are currently implemented using RSA encryption.
4) Yes, we need confidentiality, and not alteration during 2-way communication, and hence, our code ensures encryption of sensitive data during transmission.
5) Confidentiality, authentication, integrity, and non-repudiation are relevant, and we ensured this using the RSA algorithm and the Deffie-Hellman algorithm (and Digital signatures for non-repudiation); overall, we implemented the El-Gamal cryptosystem to address these.
6) Bonus- Yes, as it's important for auditing, logging and so as to check whether the license is valid or not now at current time or got expired.

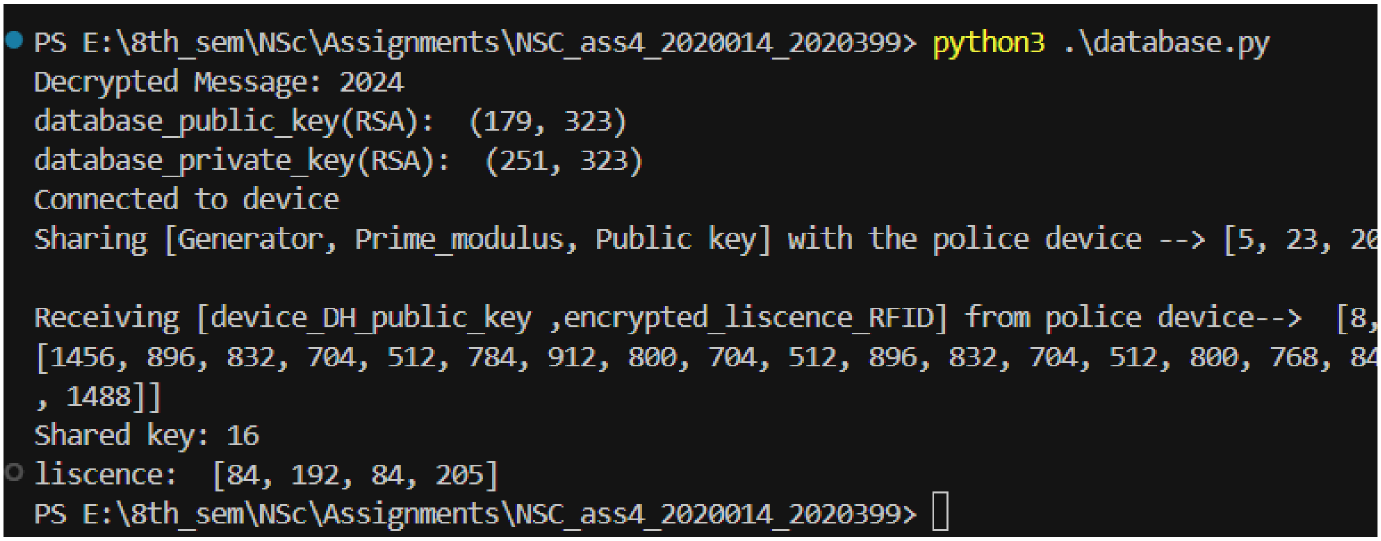
4. **Input:**



```
PS E:\8th_sem\NSc\Assignments\NSC_ass4_2020014_2020399> python3 .\device.py
device_public_key(RSA):  (73, 323)
device_private_key(RSA):  (217, 323)
Connected to database
Receiving [Generator, Prime_modulus, Public key] from TA database-->  [5, 23, 20]
Shared key: 16
Enter the driver liscence's RFID
(RFID is [84, 192, 84, 205] here): [84, 192, 84, 205]
```

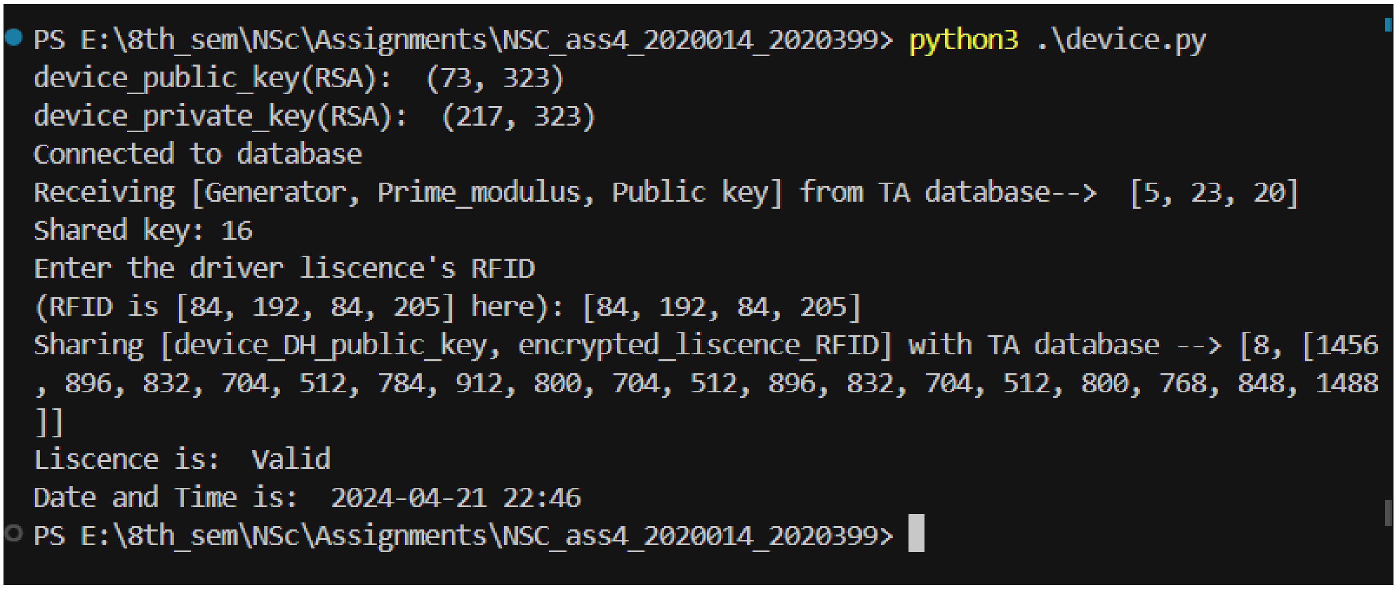
This is where the officer can enter the Rifd(we have entered the example one).

5. **Output:** The whole working of our system and output is shown below



```
PS E:\8th_sem\NSc\Assignments\NSC_ass4_2020014_2020399> python3 .\database.py
Decrypted Message: 2024
database_public_key(RSA):  (179, 323)
database_private_key(RSA):  (251, 323)
Connected to device
Sharing [Generator, Prime_modulus, Public key] with the police device --> [5, 23, 20

Receiving [device_DH_public_key ,encrypted_liscence_RFID] from police device-->  [8,
[1456, 896, 832, 704, 512, 784, 912, 800, 704, 512, 896, 832, 704, 512, 800, 768, 84
, 1488]]
Shared key: 16
liscence:  [84, 192, 84, 205]
PS E:\8th_sem\NSc\Assignments\NSC_ass4_2020014_2020399>
```



```
PS E:\8th_sem\NSc\Assignments\NSC_ass4_2020014_2020399> python3 .\device.py
device_public_key(RSA):  (73, 323)
device_private_key(RSA):  (217, 323)
Connected to database
Receiving [Generator, Prime_modulus, Public key] from TA database-->  [5, 23, 20]
Shared key: 16
Enter the driver liscence's RFID
(RFID is [84, 192, 84, 205] here): [84, 192, 84, 205]
Sharing [device_DH_public_key, encrypted_liscence_RFID] with TA database --> [8, [1456
, 896, 832, 704, 512, 784, 912, 800, 704, 512, 896, 832, 704, 512, 800, 768, 848, 1488
]]
Liscence is:  Valid
Date and Time is:  2024-04-21 22:46
PS E:\8th_sem\NSc\Assignments\NSC_ass4_2020014_2020399>
```

6. References:

a. Lecture Slides
b. GeeksforGeeks
c. https://www.geeksforgeeks.org/implementation-diffie-hellman-algorithm/
d. https://www.javatpoint.com/rsa-encryption-algorithm