# CSE350: Network Security Assignment-4

## 1. Information Supplied by the Driver vs. Information from the Server

- **Driver's Information:** The driver presents their physical or digital license, which includes personal details such as their name, photograph, license number, address, date of birth, the vehicle categories allowed, issue date, and expiration date. The license will also have a **QR code** that encapsulates this information in a machine-readable format.

- **Information from the Server:** When the officer scans the QR code, the officer retrieves the current status of the license from the central transport authority server. This information can include:

    - **Validity status**: Whether the license is active or suspended.

    - **Revocation status**: Any recent revocation or legal issues.

    - **Recent updates**: Fines, suspensions, or other modifications since the license was issued. This is important since the QR code may not be updated in real-time (e.g., suspensions or fines post-issuance).

---

## 2. Need for a Central Server and Its Role

Yes, a **central server** is essential in this system. The key roles of the central server include:

- **Data Storage:** It holds comprehensive records of all driver licenses, including personal data, license categories, issue and expiry dates, and any updates (e.g., suspensions or fines). The central server is the source of truth for the license information.

- **Real-time Verification:** The server provides real-time license validation during a traffic stop. The officer can query the server to verify the validity of the license, check if there are any updates (such as suspensions), and ensure that the license has not been revoked.

- **Public Key Distribution:** The server manages and distributes the **public keys** used to verify the digital signatures of QR codes on licenses. This ensures that only authorized authorities can issue valid, digitally signed licenses.

- **System Updates:** The central server is updated with any new driver information, modifications, or corrections, ensuring the information available to officers is up-to-date.

---

## 3. Role of Digital Signatures

- **Ensuring Authenticity and Integrity:** A **digital signature** is used to sign the license data (including key personal information and license details). The transport authority uses its **private key** to sign a **hash** (a unique digital fingerprint) of the data. This signed hash is then embedded in the QR code on the license.

- **Verification Process:** When the police officer scans the QR code, the officer's app retrieves the license data and the signature. The app then uses the **public key** of the transport authority (stored in the app or fetched from the server) to verify the signature. If the signature is valid, it proves that:
    - The data is **authentic**, i.e., it was issued by the legitimate transport authority.
    - The data has not been **altered** since it was signed, ensuring its **integrity**.

- **Forgery Prevention:** If the QR code data is altered (e.g., someone tries to tamper with the expiration date), the signature verification will fail, alerting the officer to a possible counterfeit license. Digital signatures act as a safeguard against fake or altered licenses.

---

## 4. Confidentiality and Tampering Protection

Yes, it is essential to ensure **confidentiality** and **protection against tampering** during communication between the officer's device and the transport authority's server:

- **Confidentiality:** Ensuring **privacy** is critical, especially since the driver's personal information (name, address, etc.) is transmitted during the verification process. **Encryption** via protocols like **HTTPS (SSL/TLS)** ensures that the data transmitted between the officer's device and the central server is protected from eavesdropping and unauthorized access. This prevents anyone from intercepting and reading the sensitive data being exchanged.

- **Tampering Protection:** Digital signatures embedded in the QR code prevent the license data from being modified after issuance. If any alteration occurs, the signature verification will fail, indicating potential tampering. This ensures that the license data the officer retrieves from the QR code is **authentic and unaltered**.

- **Secure Communication:** When the officer's device communicates with the server to verify license details, secure communication protocols ensure that the data remains intact during transmission. This includes using **encrypted connections** to prevent unauthorized parties from modifying or injecting fake data into the communication.

---

## 5. Relevant Security Concepts

The following concepts are all crucial for the proper functioning of the driver's license verification system:

- **Authentication:** Authentication ensures that the license presented by the driver is valid and was indeed issued by the transport authority. This is achieved through the use of **digital signatures** on the QR code. The officer's device verifies the authenticity of the license data using the **public key** of the issuing authority. Without successful authentication, the license cannot be considered genuine.

- **Integrity:** Integrity is guaranteed by **hashing** the license data before signing it. This ensures that the data has not been altered since the signature was applied. If the data is modified, the digital signature will not match, signaling a potential forgery. Integrity ensures that the data has not been tampered with during transit or storage.

- **Confidentiality:** Confidentiality protects the personal data of the driver (such as name, address, and license details) from unauthorized access during the transmission process. This is achieved through **encryption** (via HTTPS or SSL/TLS) when communicating between the officer's device and the central server. While the license data is not secret (the driver voluntarily shares it with the officer), the communication and sensitive data (like the query results) need to remain confidential.

- **Non-repudiation:** Non-repudiation ensures that the transport authority cannot deny issuing the license. The **digital signature** provides a proof of the license's issuance and cannot be refuted by the authority. If there is a dispute, the digital signature serves as irrefutable evidence that the authority issued the license. This is particularly important in legal or administrative scenarios, where the authenticity of a document may need to be legally verified.

## Implementation:

Driver's License: The driver's license data contains the following fields:

- DL_No – Driver's License Number (e.g., KA01AB1234)
- DOI – Date of Issue (e.g., 2020-01-01)
- Validity – Expiry Date of the License (e.g., 2030-01-01)
- Name – Full Name of the License Holder (e.g., Rahul Sharma)
- DOB – Date of Birth (e.g., 1990-05-15)
- Address – Residential Address (e.g., 123 MG Road, Bengaluru)
- Categories – Vehicle Categories Authorized (e.g., LMV, MCWG)
- Restrictions – Driving Restrictions, if any (e.g., Corrective Lenses)
- data_signature – Placeholder for the digital signature of the data

Generate the QR code after computing the hash and signing the concatenated (actual data + hash) data. Verify the QR Code after decoding it and comparing of hashed data with the counterpart

## QR Generation:

**Generate.py:**

- Loads driver license data from driver_data.json, which includes:
  - Name, DOB, DL number, categories, restrictions, etc.
- Generates a new RSA key pair (2048-bit)
  - Saves private.pem (for signing)
  - Saves public.pem (for verification)
- Hashes the license data using SHA256.
- Signs the hash using the **private key** via PKCS#1 v1.5.
- Combines original data and base64-encoded signature into a single payload.
- Converts this payload into a QR code and saves it as driver_license_qr.png.

**Purpose:**
Ensure the license information is **authentic** and **tamper-evident**.

## Tampering Simulation:

**make_tampered_qr.py:**

- Manually creates a **fake driver profile**:
  - E.g., name: "Evil Hacker", DL_No: "FAKE0000", etc.
- Generates a **different** RSA key pair (not from the authority).
- Signs the fake data with this unauthorized private key.
- Bundles the fake data and its signature into a QR payload.
- Saves the tampered QR code as tampered_qr.png.

**Purpose:**
To demonstrate how **unauthorized signing** can be detected at verification time.

## QR Verification:

**QR_verify.py:**

- Loads the **public key** from public.pem (originally created during QR generation).
- Reads and decodes the QR code from an image (driver_license_qr.png or tampered_qr.png).
- Extracts JSON payload: separates data and signature.

- Hashes the extracted data using SHA256.
- Uses the **public key** to verify the signature.
    - If valid → Prints full license info.
    - If invalid → Displays "Signature invalid – data may be tampered."

**Features:**

- Handles binarization if QR reading fails due to poor contrast.
- Gracefully handles malformed payloads or unreadable QR codes.

**Purpose:**
Authenticate license data and **detect tampering or forgery**.

## Instructions for running:

1. Run the **generate.py** to first generate the RSA public and private key and save them as **private.pem** and **public.pem**, it will also generate the QR code and save it in the same directory as **driver_license_qr.png.**
2. Run **QR_verify.py** to verify the generated QR code. If it gets verified, it will display full license info; else, it displays "Signature invalid – data may be tampered."
3. If you want to check the verification for tampered data, run **make_tampered_qr.py** and then again run the **QR_verify.py.**

## Sample Input:

**Driver's license data:**

```
{
   "DL_No": "KA01AB1234",
   "DOI": "2020-01-01",
   "Validity": "2030-01-01",
   "Name": "Rahul Sharma",
   "DOB": "1990-05-15",
   "Address": "123 MG Road, Bengaluru",
   "data_signature": "",
   "categories": ["LMV", "MCWG"],
   "restrictions": ["Corrective Lenses"]
 }
```

## Generated QR:

## In Case of Valid data:

```
(venv) akash@akash-Modern-14-B11MOU:~/Downloads/Nsc_A4$ python3 QR_verify.py
 Signature valid. Driver info:
{
  "Address": "123 MG Road, Bengaluru",
  "DL_No": "KA01AB1234",
  "DOB": "1990-05-15",
  "DOI": "2020-01-01",
  "Name": "Rahul Sharma",
  "Validity": "2030-01-01",
  "categories": [
    "LMV",
    "MCWG"
  ],
  "data_signature": "",
  "restrictions": [
    "Corrective Lenses"
  ]
}
```

## In case of tampered data:

```
(venv) akash@akash-Modern-14-B11MOU:~/Downloads/Nsc_A4$ python3 QR_verify.py
Signature invalid — data may be tampered.
(venv) akash@akash-Modern-14-B11MOU:~/Downloads/Nsc_A4$
```