Overview

The system comprises several components:

Certificate: Represents a driver license, including issuer information, a unique ID, date of issuance, validity duration, public key, and a digital signature.
Transport Office: Manages regional offices and issues certificates.
Regional Office: Issues and verifies certificates for drivers.
Driver: Represents an individual with a unique ID and associated RSA keys.
Officer: Responsible for verifying the validity and authenticity of certificates.

Key Components and Their Functions

1. Certificate Class

Attributes:
issuer: The ID of the issuing office.
id: The unique ID of the certificate (e.g., driver's license number).
date_issued: The time of issuance, formatted as HH:MM:SS.
validity: Duration for which the certificate is valid, in seconds.
publickey: Public key associated with the driver.
signature: Digital signature of the certificate.
Purpose: Represents a license or certificate, including cryptographic elements for security.

2. Transport Office Class

Methods:
register_office(Regional_office): Registers a regional office and stores its public key.
get_certificate(id): Issues a new certificate with a digital signature.
Purpose: Central management and certificate issuance authority.

3. Regional Office Class

Methods:
get_certificate(id): Retrieves or issues a certificate, verifying its validity and authenticity.
make_certificate(Driver): Issues a certificate specifically for a driver.
Purpose: Handles local certificate management and interacts with the central office.

4. Driver Class

Attributes:
Certificate: Stores the driver's certificate.
Purpose: Represents an entity that receives and uses certificates (licenses).

5. Officer Class

Methods:
verify(certificate, time): Verifies the authenticity and validity of a given certificate.
Purpose: Enforcement and verification of certificates.

Security Implementation

RSA Encryption: Used for digital signatures, ensuring that certificates are authentic and tamper-proof.
Hashing: SHA-256 is used to hash the certificate data before signing, providing integrity.
Functions
create_hash(certificate): Generates a SHA-256 hash of the certificate's concatenated data.
verify_certificate(certificate, public_key): Verifies the digital signature using RSA.
verify_time(certificate, secpassed): Checks if the certificate is still valid based on its issuance time and validity period.

System Workflow

Initialization: Transport and regional offices are initialized with RSA keys. Drivers are registered with unique IDs and keys.
Certificate Issuance: Regional offices request certificates from the transport office, which signs them. These certificates are then given to drivers.
Verification Process: When verification is needed, an officer checks both the digital signature and the temporal validity of the certificate.

Example Scenario

Setup: Transport office registers regional offices and generates their keys. Drivers are initialized.
Certificate Issuance: A driver requests a license, and a regional office issues it after verifying with the transport office.
Verification: An officer verifies a driver’s certificate for both signature authenticity and time validity.

Assumptions:

The regional transport office always knows the public key of head transport office.
Head transport Office always knows the public key of all the Regional transport Office.
Here Certificate represents Driver’s License
Instead of date only time parameter is used as both are interconvertible

Answers to Question given in Assignment:
1)
Information supplied is License in form of certificate here. Officer just need the public key of the Regional transport office that made that certificate/license and then verify by decrypting and then hashing.
2)No, we don't need a central server. We can do this by making a hierarchical server. In the system provided the Head Transport Office only knows about the 2 regional Office not the Driver

only Regional Office knows about the drivers. This reduces the load on the Head Transport Office. In the central system this will put too much load on one server.

3)Digital signatures can be kept by individuals to be verified by anyone and can be given as identity proof , as they are created by Certified Authorities. Here it has the non signature part and then decrypts using the public key of the trusted Certification authority who made that certificate and then hashes the decrypted part. And then compare the two if it matches the Document is Valid.

4)
In the context of the system even if the message is changed on the go the system can stil detect it as then the hash will not match. It is also assumed that the decrypting key of the head transport office is already available with the regional office and when 1 regional office gets a key/certificate for other regional office it gets it from the head office which it can verify using its key. And since the goal is Authentication not Confidentiality.

5)Authentication and non-repudiation is relevant as the regional Transport Office cannot deny that a particular certificate is not issued by them as it is encrypted by its private key. Same for the Head transport Office. Authentication is one of the main goals to verify if the document is valid.

Bonus:

The system comprises time based certificates which have their own validity to check if a certificate at each level is expired or not