Overview

This Python script facilitates interactions with a smart card via a compatible card reader. It demonstrates how to:

- Connect to the smart card.
- Send Application Protocol Data Units (APDUs) to perform various operations such as:
  - Selecting an applet (AID).
  - Setting PIN, PUK, and SO keys.
  - Generating RSA and ECDSA key pairs.
  - Generating and importing certificates.
  - Renaming certificates.
  - Changing and unblocking PINs.

Dependencies

The script requires the following Python libraries:

1. **pySmartCard** : A library for interacting with smart cards through a reader.

- Install via `pip install pyscard`.

1. **logging** : For logging messages related to script execution and errors.

Hardware Requirements

1. **Smart Card Reader** : A compatible reader connected to your machine.

2. **Smart Card** : The smart card should support the required operations (e.g., RSA/ECDSA key generation, certificate management).

Script Breakdown

1. **Logging Setup** : The script uses the `logging` library to log key actions, successes, and errors to a log file (`cosmo_x_interaction.log`).
2. **AID (Applet Identifier)** : The script selects a specific applet by sending an APDU with a predefined AID (`AID_APPLET`). The AID represents a specific smart card application.
3. **Sending APDU Commands** : The `send_apdu_command` function sends an APDU command to the smart card. It logs the APDU request, response, and status words (SW1, SW2). If the status is not successful, it logs a warning.
4. **Smart Card Operations** :

- **Select AID** : The `select_aid` function selects the applet on the smart card by sending the `SELECT AID` APDU.
- **Initialize Keys (PIN, PUK, SO)** : The `initialize_keys` function sets the PIN, PUK, and SO keys on the smart card.
- **Generate RSA/ECDSA Keys** : These functions generate RSA and ECDSA key pairs on the card.
- **Generate CSR** : Generates a Certificate Signing Request (CSR) using a key reference and a template.
- **Import Certificate** : Imports a certificate onto the smart card.
- **Rename Certificate** : Renames a certificate based on its key ID and alias.
- **Change and Unblock PIN** : The `change_pin` and `unblock_and_set_pin` functions allow you to change and unblock the PIN on the smart card.

1. **Main Function** :

- It checks for available card readers and selects the first one.
- It initializes keys (PIN, PUK, SO) and performs operations like generating keys, importing certificates, and managing PINs.

How to Run the Script

1. **Install Dependencies** :

- Ensure that you have Python installed on your system (preferably Python 3.x).
- Install the required libraries using `pip`

```
pip install pyscard
```

- **Connect the Smart Card Reader** :
- Ensure your smart card reader is properly connected to your computer.
- Insert the smart card into the reader.
- **Modify the Script for Your Setup** :
- Modify the `AID_APPLET` variable if you are working with a different applet on your smart card.
- Replace the PIN, PUK, and SO keys with the actual values for your card.
- If you're importing a certificate, make sure the `dummy_cert` is a valid certificate in the correct format.
- **Run the Script** :
- After ensuring your setup is correct, run the script from the terminal:

```
python card_reader.py
```

1. **Check Logs** :

- The script will generate logs in a file called `cosmo_x_interaction.log` in the same directory.
- The logs will contain detailed information about the APDU commands sent, responses received, and the success or failure of each operation.

1. **Troubleshooting** :

- If the script encounters an error, it will log the error in the log file. Check the logs for detailed information about what went wrong.
- Ensure that the smart card reader is compatible and correctly connected.
- The script will check for available readers and select the first one; if no readers are found, an error will be logged.

Example of a Log Entry

The log file will look something like this:

```
2024-11-17 12:45:30,134 - INFO - Selecting AID: Sending APDU: 00 A4 04 00 10 A0 00
00 77 03 0C 60 00 00 00 00 FE 00 00 05 00 00
```

```
2024-11-17 12:45:30,189 - INFO - Selecting AID: Response: 90 00, SW1: 90, SW2: 00
2024-11-17 12:45:30,200 - INFO - Setting PIN: Sending APDU: 00 20 00 00 02 12 34
2024-11-17 12:45:30,250 - INFO - Setting PIN: Response: 90 00, SW1: 90, SW2: 00
...
```

Function Details

- **send_apdu_command(reader, apdu_command, description)** :
- Sends an APDU command to the smart card.
- Logs the APDU command and response.
- Returns the response and status words.
- **select_aid(reader)** :
- Sends a `SELECT AID` command to select the applet on the smart card.
- Returns the response and status.
- **initialize_keys(reader, pin, puk, so_key)** :
- Sets the PIN, PUK, and SO key on the smart card.
- Sends appropriate APDU commands to initialize the keys.
- **generate_rsa_key(reader)** and **generate_ecdsa_key(reader)** :
  - Generates RSA and ECDSA key pairs on the smart card.
  - Logs success or failure.
- **import_certificate(reader, cert_data)** :
- Imports a certificate onto the smart card.
- Takes `cert_data` as input, which is the certificate to be imported.
- **rename_certificate(reader, key_id, alias)** :
- Renames a certificate by its key ID and alias.
- **unblock_and_set_pin(reader, puk, new_pin)** :
- Unblocks the PIN and sets a new PIN using the PUK.
- **change_pin(reader, old_pin, new_pin)** :
- Changes the PIN on the smart card.

Security Considerations

- Be cautious when logging sensitive information like PINs, keys, or certificates in a production environment.
- Mask or redact sensitive data in the logs if necessary.