Aplikasi Pengiriman Data Pembayaran Menggunakan Enkripsi AES & DES

Demo aplikasi: https://youtu.be/xoUMgiOU03c

Mochamad Zidan Hadipratama - 5027221052
Marselinus Krisnawan Riandika - 5027221056

Client.py

```
from flask import Flask, request, render_template, redirect, url_for
import requests

app = Flask(__name__)

@app.route('/')
def landing():
    return render_template('landing.html')

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        full_name = request.form['full_name']
        credit_card = request.form['credit_card']
        cvv = request.form['cvv']
        expiration_year = request.form['expiration_year']
        key = request.form['key']

        data = {
            'full_name': full_name,
            'credit_card': credit_card,
            'cvv': cvv,
            'expiration_year': expiration_year,
            'key': key
        }

        response = requests.post('http://127.0.0.1:5001/encrypt',
data=data)

        return redirect(url_for('success'))
```

```
    return render_template('client.html')

@app.route('/success')
def success():
    return render_template('success.html')

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        full_name = request.form['full_name']
        key = request.form['key']

        data = {'full_name': full_name, 'key': key}
        response = requests.post('http://127.0.0.1:5001/decrypt',
data=data)

        if response.status_code == 404:
            return render_template('retrieve.html', error="User not found")


        if response.status_code == 403:
            return render_template('retrieve.html', error="Invalid key,
please try again")

        decrypted_data = response.json()
        return render_template('retrieve.html',
credit_card=decrypted_data['credit_card'], cvv=decrypted_data['cvv'],
expiration_year=decrypted_data['expiration_year'])

    return render_template('retrieve.html')

if __name__ == '__main__':
    app.run(debug=True, port=5000)
```

PENJELASAN KODE

```
app = Flask(__name__)
```

Digunakan untuk membuat instance aplikasi Flask.

```
@app.route('/')
def landing():
   return render_template('landing.html')
```

Digunakan untuk mengarahkan ke landing page, dan “landing()” Untuk merender template landing.html.

```
@app.route('/register', methods=['GET', 'POST'])
def register():
   if request.method == 'POST':
       full_name = request.form['full_name']
       credit_card = request.form['credit_card']
       cvv = request.form['cvv']
       expiration_year = request.form['expiration_year']
       key = request.form['key']

       data = {
           'full_name': full_name,
           'credit_card': credit_card,
           'cvv': cvv,
           'expiration_year': expiration_year,
           'key': key
       }

       response = requests.post('http://127.0.0.1:5001/encrypt',
data=data)

       return redirect(url_for('success'))

   return render_template('client.html')
```

Digunakan untuk Mengarahkan ke Register, memiliki 2 metode HTTP yaitu get dan post. Pada metode post, terdapat data nama, credit card, cvv, cvv, expiration year, dan key (password). Dan Post digunakan untuk Mengirimkan formulir yang telah diisi tadi menuju ./encrypt untuk di enrypt. Apabila sudah maka menuju Halaman success dan menuju “client.html” kembali

```
@app.route('/success')
def success():
   return render_template('success.html')
```

Digunakan untuk mengarahkan ke ./success dan merender template success.html

```
@app.route('/login', methods=['GET', 'POST'])
def login():
   if request.method == 'POST':
       full_name = request.form['full_name']
       key = request.form['key']

       data = {'full_name': full_name, 'key': key}
       response = requests.post('http://127.0.0.1:5001/decrypt',
data=data)

       if response.status_code == 404:
           return render_template('retrieve.html', error="User not found")


       if response.status_code == 403:
           return render_template('retrieve.html', error="Invalid key,
please try again")

       decrypted_data = response.json()
       return render_template('retrieve.html',
credit_card=decrypted_data['credit_card'], cvv=decrypted_data['cvv'],
expiration_year=decrypted_data['expiration_year'])

   return render_template('retrieve.html')
```

Digunakan untuk merutekan ke ./login dan memiliki metode GET dan POST. Pada metode post digunakan untuk login dengan menggunakan Full name dan Keynya. Kemudian data tadi dikirimkan ke ./decrypt untuk mendecrypt data yang kemudian dicek apabila data tersebut benar atau salah. Apabila benar maka akan merender retrieve.html yang akan menampilkan data data user yang sudah diisi pada saat register

Server.py:

```
from flask import Flask, request, jsonify, render_template
from Crypto.Cipher import DES, AES
import base64
import hashlib
import os

app = Flask(__name__)

def generate_16_char_key(key):
    hashed_key = hashlib.sha256(key.encode('utf-8')).hexdigest()

    return hashed_key[:16]

def encrypt_data(data, key):
    des_key = key[:8]
    aes_key = generate_16_char_key(key)

    des_cipher = DES.new(des_key.encode('utf-8'), DES.MODE_ECB)
    padded_data = data + (8 - len(data) % 8) * ' '
    des_encrypted = des_cipher.encrypt(padded_data.encode('utf-8'))

    aes_cipher = AES.new(aes_key.encode('utf-8'), AES.MODE_ECB)
    aes_encrypted = aes_cipher.encrypt(des_encrypted.ljust(16, b' '))

    return base64.b64encode(aes_encrypted).decode('utf-8')

def decrypt_data(encrypted_data, key):
    des_key = key[:8]
    aes_key = generate_16_char_key(key)

    try:

        aes_cipher = AES.new(aes_key.encode('utf-8'), AES.MODE_ECB)
        aes_decrypted =
aes_cipher.decrypt(base64.b64decode(encrypted_data))

        des_cipher = DES.new(des_key.encode('utf-8'), DES.MODE_ECB)
        des_decrypted =
des_cipher.decrypt(aes_decrypted).decode('utf-8').strip()
```

```
        return des_decrypted.strip()
    except Exception as e:

        return None


def save_to_file(username, encrypted_data):
    with open("penting.txt", "a") as file:
        file.write(f"{username},{encrypted_data}\n")


def read_all_data_from_file():
    if os.path.exists("penting.txt"):
        with open("penting.txt", "r") as file:
            data = [line.strip().split(',') for line in file.readlines()]
        return data
    return []


def read_from_file(username):
    if os.path.exists("penting.txt"):
        with open("penting.txt", "r") as file:
            for line in file:
                saved_username, saved_encrypted_data =
line.strip().split(',')
                if saved_username == username:
                    return saved_encrypted_data
    return None

@app.route('/encrypt', methods=['POST'])
def encrypt():

    full_name = request.form['full_name']
    credit_card = request.form['credit_card']
    cvv = request.form['cvv']
    expiration_year = request.form['expiration_year']
    key = request.form['key']
```

```
    payment_data = f"Card: {credit_card}, CVV: {cvv}, Expiry:
{expiration_year}"


    encrypted_data = encrypt_data(payment_data, key)


    save_to_file(full_name, encrypted_data)

    return encrypted_data

@app.route('/decrypt', methods=['POST'])
def decrypt():

    full_name = request.form['full_name']
    key = request.form['key']


    encrypted_data = read_from_file(full_name)
    if not encrypted_data:
        return "User not found", 404


    decrypted_data = decrypt_data(encrypted_data, key)

    if not decrypted_data:
        return "Invalid key", 403


    parts = decrypted_data.split(", ")
    credit_card = parts[0].split(": ")[1]
    cvv = parts[1].split(": ")[1]
    expiration_year = parts[2].split(": ")[1]

    return jsonify({
        'credit_card': credit_card,
        'cvv': cvv,
        'expiration_year': expiration_year
    })
```

```python
@app.route('/')
def users():

    user_data = read_all_data_from_file()


    return render_template('server.html', user_data=user_data)

if __name__ == '__main__':
    app.run(debug=True, port=5001)
```

PENJELASAN KODE

```
def generate_16_char_key(key):
    hashed_key = hashlib.sha256(key.encode('utf-8')).hexdigest()
    return hashed_key[:16]
```

Digunakan untuk menghasilkan kunci 16 karakter dari kunci yang dimasukkan pengguna

```
def encrypt_data(data, key):
    des_key = key[:8]
    aes_key = generate_16_char_key(key)

    des_cipher = DES.new(des_key.encode('utf-8'), DES.MODE_ECB)
    padded_data = data + (8 - len(data) % 8) * ' '
    des_encrypted = des_cipher.encrypt(padded_data.encode('utf-8'))

    aes_cipher = AES.new(aes_key.encode('utf-8'), AES.MODE_ECB)
    aes_encrypted = aes_cipher.encrypt(des_encrypted.ljust(16, b' '))

    return base64.b64encode(aes_encrypted).decode('utf-8')
```

Digunakan untuk menggabungkan dua algoritma enkripsi, yaitu DES (Data Encryption Standard) dan AES (Advanced Encryption Standard), untuk mengenkripsi data sensitif.

```
def decrypt_data(encrypted_data, key):
    des_key = key[:8]
    aes_key = generate_16_char_key(key)

    try:
        aes_cipher = AES.new(aes_key.encode('utf-8'), AES.MODE_ECB)
        aes_decrypted =
aes_cipher.decrypt(base64.b64decode(encrypted_data))

        des_cipher = DES.new(des_key.encode('utf-8'), DES.MODE_ECB)
        des_decrypted =
des_cipher.decrypt(aes_decrypted).decode('utf-8').strip()

        return des_decrypted.strip()
    except Exception as e:
        return None
```

Digunakan melakukan dekripsi gabungan dari data yang telah dienkripsi menggunakan DES dan AES.

```
def save_to_file(username, encrypted_data):
```

```
    with open("penting.txt", "a") as file:
        file.write(f"{username},{encrypted_data}\n")
```

Fungsi ini menyimpan data pengguna dan data terenkripsi ke file penting.txt. Data disimpan dalam format username,encrypted_data di setiap baris

```
def read_all_data_from_file():
    if os.path.exists("penting.txt"):
        with open("penting.txt", "r") as file:
            data = [line.strip().split(',') for line in file.readlines()]
        return data
    return []
```

Fungsi ini membaca semua data dari file penting.txt dan mengembalikan data dalam bentuk list, di mana setiap elemen list berisi username dan data terenkripsi.

```
def read_from_file(username):
    if os.path.exists("penting.txt"):
        with open("penting.txt", "r") as file:
            for line in file:
                saved_username, saved_encrypted_data =
line.strip().split(',')
                if saved_username == username:
                    return saved_encrypted_data
    return None
```

Fungsi ini mencari data pengguna berdasarkan username di dalam file penting.txt. Jika ditemukan, fungsi ini mengembalikan data terenkripsi milik pengguna tersebut.

```
@app.route('/encrypt', methods=['POST'])
def encrypt():

    full_name = request.form['full_name']
    credit_card = request.form['credit_card']
    cvv = request.form['cvv']
    expiration_year = request.form['expiration_year']
    key = request.form['key']


    payment_data = f"Card: {credit_card}, CVV: {cvv}, Expiry:
{expiration_year}"


    encrypted_data = encrypt_data(payment_data, key)
```

```
    save_to_file(full_name, encrypted_data)

    return encrypted_data
```

Kode ini akan menerima request post dengan data data register. Data data tadi akan menjadi string lalu akan dienkripsi dengan fungsi encypt data dan hasilnya akan disimpan di penting.txt

```
@app.route('/decrypt', methods=['POST'])
def decrypt():

    full_name = request.form['full_name']
    key = request.form['key']


    encrypted_data = read_from_file(full_name)
    if not encrypted_data:
        return "User not found", 404

    decrypted_data = decrypt_data(encrypted_data, key)

    if not decrypted_data:
        return "Invalid key", 403

    parts = decrypted_data.split(", ")
    credit_card = parts[0].split(": ")[1]
    cvv = parts[1].split(": ")[1]
    expiration_year = parts[2].split(": ")[1]
    return jsonify({
        'credit_card': credit_card,
        'cvv': cvv,
        'expiration_year': expiration_year
    })
```

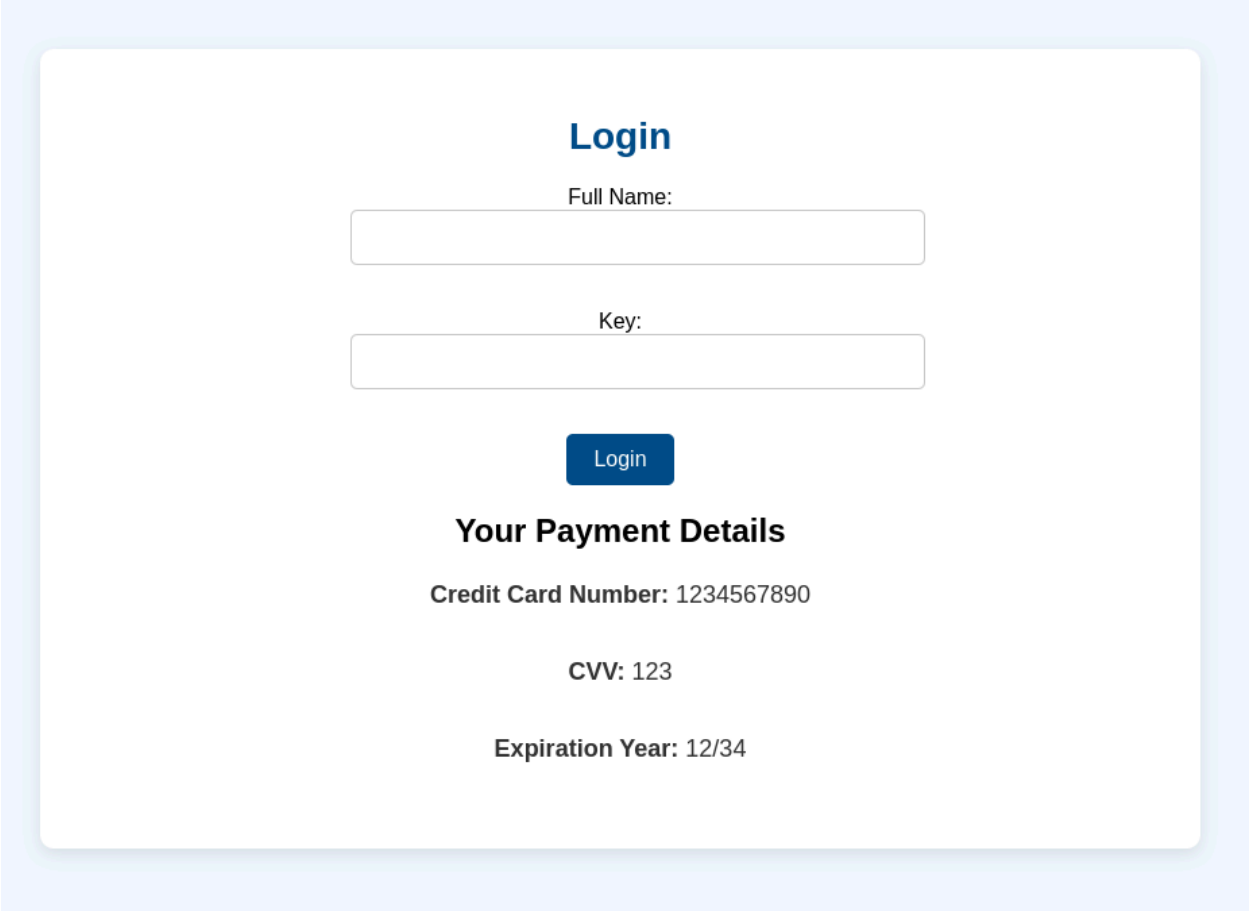
Rute ini akan menerima request post dengan full name dan key dari pengguna. Kemudian Data terenkripsi milik pengguna dicari di file menggunakan fungsi read_from_file(). ika ditemukan, data tersebut didekripsi menggunakan fungsi decrypt_data().Hasil dekripsi dikembalikan dalam format JSON yang berisi nomor kartu kredit, CVV, dan tahun kedaluwarsa. Jika pengguna tidak ditemukan atau kunci salah, pesan error 404 atau 403 dikembalikan.

```
@app.route('/')
def users():
```

```
    user_data = read_all_data_from_file()

    return render_template('server.html', user_data=user_data)
```

Rute ini menampilkan daftar semua pengguna dan data mereka yang terenkripsi dalam bentuk tabel di template server.html. Data diambil dari file penting.txt menggunakan fungsi read_all_data_from_file().



Register Your Payment Details

Full Name:

Credit Card Number:

CVV:

Expiration Year:

Key (8 characters):

Register