Penjelasan Kode Aplikasi Catatan Terenkripsi dengan Enkripsi ElGamal

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

Mochamad Zidan Hadipratama - 5027221052
Marselinus Krisnawan Riandika - 5027221056

---

1. Inisialisasi Aplikasi Flask dan Mengimpor Modul yang Diperlukan

```
import os
import hashlib
from flask import Flask, render_template, request, redirect, url_for,
flash, session
from Crypto.PublicKey import ElGamal
from Crypto.Random import random, get_random_bytes
from Crypto.Cipher import AES
from Crypto.Hash import SHA256
from base64 import b64encode, b64decode
from Crypto.Util.Padding import pad, unpad

app = Flask(__name__)
app.secret_key = 'your_secret_key'
```

Penjelasan:
Bagian ini mengimpor berbagai modul yang diperlukan:

- `os` untuk pengelolaan file dan direktori,
- `hashlib` untuk hashing kata sandi,
- `flask` untuk membuat aplikasi web dan mengelola sesi,
- `Crypto` untuk menyediakan algoritma ElGamal, AES, dan SHA-256,
- `base64` untuk encoding/decoding base64, dan
- `app.secret_key` untuk mengamankan sesi aplikasi.

---

2. Mendefinisikan Direktori dan File yang Digunakan

```
KEYS_DIRECTORY = 'data/keys'
```

```
NOTES_DIRECTORY = 'data/notes'
CREDS_FILE = 'data/creds.txt'
```

Penjelasan:

- **KEYS_DIRECTORY** untuk menyimpan kunci privat dan publik per pengguna,
- **NOTES_DIRECTORY** untuk menyimpan catatan terenkripsi,
- **CREDS_FILE** untuk menyimpan kredensial pengguna berupa nama pengguna dan kata sandi yang sudah di-hash.

---

3. Fungsi generate_elgamal_keys()

```
def generate_elgamal_keys():
    key = ElGamal.generate(2048, random.get_random_bytes)
    private_key = key.export_key()
    public_key = key.publickey().export_key()
    return private_key, public_key
```

Penjelasan:
Fungsi ini menghasilkan pasangan kunci ElGamal (privat dan publik) dengan panjang kunci 2048 bit:

1. **key = ElGamal.generate(2048, random.get_random_bytes)**: Menghasilkan kunci ElGamal.
2. **private_key dan public_key**: Mengekspor kunci privat dan publik ke dalam format yang bisa disimpan di file.

---

4. Fungsi save_user_keys(username, private_key, public_key)

```
def save_user_keys(username, private_key, public_key):
    username = username.lower()
    user_dir = os.path.join(KEYS_DIRECTORY, username)

    if not os.path.exists(user_dir):
        os.makedirs(user_dir)

    private_key_file = os.path.join(user_dir, 'private_key.pem')
    with open(private_key_file, 'wb') as f:
        f.write(private_key)
```

```
    public_key_file = os.path.join(user_dir, 'public_key.pem')
    with open(public_key_file, 'wb') as f:
        f.write(public_key)
```

Penjelasan:
Menyimpan kunci privat dan publik pengguna:

- `user_dir` adalah direktori unik untuk pengguna,
- Membuat file `private_key.pem` dan `public_key.pem` di dalamnya untuk menyimpan kunci privat dan publik pengguna.

---

5. **Fungsi load_user_keys(username)**

```
def load_user_keys(username):
    username = username.lower()
    user_dir = os.path.join(KEYS_DIRECTORY, username)
    private_key_file = os.path.join(user_dir, 'private_key.pem')
    public_key_file = os.path.join(user_dir, 'public_key.pem')

    if os.path.exists(private_key_file) and
os.path.exists(public_key_file):
        with open(private_key_file, 'rb') as f:
            private_key = f.read()
        with open(public_key_file, 'rb') as f:
            public_key = f.read()
        return private_key, public_key
    else:
        return None, None
```

Penjelasan:
Memuat kunci privat dan publik yang disimpan di direktori pengguna. Jika tidak ada, mengembalikan `None`.

---

6. **Fungsi hash_password(password)**

```
def hash_password(password):
    return hashlib.sha256(password.encode()).hexdigest()
```

Penjelasan:

Meng-hash kata sandi menggunakan SHA-256 untuk keamanan, sehingga kata sandi tidak disimpan dalam bentuk asli.

---

7. **Fungsi save_user_credentials(username, hashed_password)**

```
def save_user_credentials(username, hashed_password):
    with open(CREDS_FILE, 'a') as f:
        f.write(f"{username},{hashed_password}\n")
```

Penjelasan:
Menyimpan kredensial pengguna (nama pengguna dan kata sandi yang di-hash) dalam file `CREDS_FILE`.

---

8. **Fungsi check_credentials(username, password)**

```
def check_credentials(username, password):
    hashed_input_password = hash_password(password)
    if os.path.exists(CREDS_FILE):
        with open(CREDS_FILE, 'r') as f:
            for line in f.readlines():
                stored_username, stored_hashed_password =
line.strip().split(',')
                if stored_username == username and stored_hashed_password
== hashed_input_password:
                    return True
    return False
```

Penjelasan:
Memverifikasi kredensial pengguna saat login. Mengembalikan `True` jika kredensial cocok, `False` jika tidak.

---

9. **Rute Flask untuk Autentikasi dan Registrasi**

```
@app.route('/')
```

```
def home():
    return render_template('home.html')
```

Penjelasan:
Rute ini menampilkan halaman utama aplikasi, `home.html`.

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

        if check_credentials(username, password):
            session['username'] = username
            flash(f"Welcome, {username}!", "success")
            return redirect(url_for('dashboard'))
        else:
            flash("Invalid username or password. Please try again.",
"error")
            return redirect(url_for('login'))

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

Penjelasan:
Rute `/login` menangani proses login:

1. **Jika** `POST`: Memeriksa kredensial. Jika valid, menyimpan username di session dan mengarahkan pengguna ke dashboard.
2. Jika tidak valid, mengarahkan kembali ke halaman login.

```
@app.route('/logout')
def logout():
    session.pop('username', None)
    flash("You have been logged out.", "success")
    return redirect(url_for('login'))
```

Penjelasan:
Rute /logout menghapus username dari sesi dan mengarahkan pengguna ke halaman login.

```
@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username'].lower()
        password = request.form['password']
```

```
        private_key, public_key = load_user_keys(username)

        if private_key is None or public_key is None:
            hashed_password = hash_password(password)
            private_key, public_key = generate_elgamal_keys()
            save_user_keys(username, private_key, public_key)
            save_user_credentials(username, hashed_password)

            flash(f"User {username} registered successfully!", "success")
            return redirect(url_for('login'))
        else:
            flash(f"User {username} already exists!", "error")
            return redirect(url_for('register'))

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

Penjelasan:
Rute /register menangani registrasi pengguna baru. Jika pengguna belum ada, maka kunci ElGamal akan dibuat dan disimpan, lalu kredensial pengguna akan disimpan.

---

10. **Fungsi encrypt_message(public_key_str, message)**

```
def encrypt_message(public_key_str, message):
    public_key = ElGamal.import_key(public_key_str)
    session_key = get_random_bytes(16)
    cipher_aes = AES.new(session_key, AES.MODE_CBC)
    ciphertext = cipher_aes.encrypt(pad(message.encode('utf-8'),
AES.block_size))
    iv = cipher_aes.iv

    # Encrypt the session key with ElGamal public key
    k = random.StrongRandom().randint(1, public_key.p - 1)
    cipher_text_elgamal = public_key.encrypt(session_key, k)

    encrypted_message = {
        'elgamal': cipher_text_elgamal,
        'aes': b64encode(ciphertext).decode('utf-8'),
        'iv': b64encode(iv).decode('utf-8')
    }

    return encrypted_message
```

Penjelasan:

Fungsi `encrypt_message()` mengenkripsi pesan:

1. **Generate `session_key`:** Menghasilkan kunci sesi AES.
2. **`cipher_aes` dan `ciphertext`:**

`Mengenkripsi pesan dengan AES.`

1. **Enkripsi kunci sesi dengan ElGamal:** Kunci sesi dienkripsi dengan kunci publik ElGamal.
2. **`encrypted_message`:** Mengembalikan pesan terenkripsi yang terdiri dari kunci AES terenkripsi dan data AES.

---

11. Fungsi decrypt_message(private_key_str, encrypted_message)

```
def decrypt_message(private_key_str, encrypted_message):
    private_key = ElGamal.import_key(private_key_str)

    elgamal_encrypted = encrypted_message['elgamal']
    aes_encrypted = b64decode(encrypted_message['aes'])
    iv = b64decode(encrypted_message['iv'])

    # Decrypt the session key with ElGamal private key
    session_key = private_key.decrypt(elgamal_encrypted)
    cipher_aes = AES.new(session_key, AES.MODE_CBC, iv)
    decrypted_message = unpad(cipher_aes.decrypt(aes_encrypted),
AES.block_size)

    return decrypted_message.decode('utf-8')
```

Penjelasan:
Mendekripsi pesan yang terenkripsi dalam `encrypted_message`:

1. **Dekripsi kunci sesi dengan ElGamal**.
2. **Dekripsi AES menggunakan kunci sesi** untuk mendapatkan pesan asli.

---

12. Fungsi save_note dan load_notes

```
def save_note(username, judul, encrypted_note):
    username = username.lower()
    user_notes_file = os.path.join(NOTES_DIRECTORY,
```

```
f"{username}_notes.txt")

    with open(user_notes_file, 'a') as f:
        f.write(f"{judul},{encrypted_note}\n")
```

Penjelasan:
Fungsi `save_note()` menyimpan catatan terenkripsi pengguna ke dalam file teks.

```
def load_notes(username):
    username = username.lower()
    user_notes_file = os.path.join(NOTES_DIRECTORY,
f"{username}_notes.txt")
    notes = []

    if os.path.exists(user_notes_file):
        with open(user_notes_file, 'r') as f:
            for line in f.readlines():
                judul, encrypted_note = line.strip().split(',', 1)
                notes.append({
                    'judul': judul,
                    'content': eval(encrypted_note)  # Convert string back
to dict
                })
    return notes
```

Penjelasan:
`load_notes()` membaca catatan yang telah terenkripsi dari file dan mengonversi kembali string catatan terenkripsi menjadi dictionary.

Menjalankan Aplikasi

```
if __name__ == '__main__':
    os.makedirs(KEYS_DIRECTORY, exist_ok=True)
    os.makedirs(NOTES_DIRECTORY, exist_ok=True)
    app.run(debug=True, port=5005)
```

Penjelasan:
Pada bagian ini, aplikasi Flask diinisialisasi dan dijalankan di port `5005`.