Implementation of DES Encryption Algorithm

Supervisor

Dr. Narendran Rajagopalan,

Associate Professor,

NITPY.

Assignment by,

Mutyala Shiva,

(CS22B1034).

Introduction:

In today's world, information security plays a critical role in protecting data from unauthorized access and tampering. One of the earliest and most well-known encryption techniques is the Data Encryption Standard (DES), a symmetric-key algorithm for encrypting and decrypting information.

This project presents a simple web application that demonstrates the DES encryption and decryption process using Python (Flask framework) for backend logic and HTML/CSS for the frontend interface.

Users can input a message, provide an 8-character key, and choose whether to encrypt or decrypt their message, helping them understand how classical encryption algorithms work in practice.

Implementation:

app.py

```python

from flask import Flask, render_template, request
import pyDes

app = Flask(__name__)

def des_encrypt(data, key):
    des = pyDes.des(key, pyDes.ECB, padmode=pyDes.PAD_PKCS5)
    encrypted_data = des.encrypt(data)
    return encrypted_data.hex()  # Return as hex string for display

def des_decrypt(data, key):
    des = pyDes.des(key, pyDes.ECB, padmode=pyDes.PAD_PKCS5)
    decrypted_data = des.decrypt(bytes.fromhex(data))
    return decrypted_data.decode()

@app.route('/', methods=['GET'])
def home():
    return render_template('index.html')

@app.route('/process', methods=['POST'])
def process():
    message = request.form['message']
    key = request.form['key']
    operation = request.form['operation']
    result = ''

    if len(key) != 8:
        result = "Key must be exactly 8 characters long."
    else:
        try:
            if operation == 'encrypt':
                result = des_encrypt(message, key)
            else:
                result = des_decrypt(message, key)
        except Exception as e:
            result = f"Error: {str(e)}"

    return render_template('index.html', result=result)

if __name__ == '__main__':
    app.run(debug=True)

```

app.py index.html

templates > index.html > ...

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>DES Encryption App</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <div class="container">
        <h1>DES Encryption/Decryption</h1>
        <form method="POST" action="/process">
            <textarea name="message" placeholder="Enter your message here" required></textarea><br>
            <input type="text" name="key" placeholder="Enter 8-character key" maxlength="8" required><br>
            <select name="operation" required>
                <option value="encrypt">Encrypt</option>
                <option value="decrypt">Decrypt</option>
            </select><br>
            <button type="submit">Submit</button>
        </form>

        {% if result %}
            <div class="result">
                <h2>Result:</h2>
                <p>{{ result }}</p>
            </div>
        {% endif %}
    </div>
</body>
</html>
```

app.py | index.html | style.css

static > # style.css > body

```css
body {
    font-family: Arial, sans-serif;
    background-color: #f5f5f5;
    text-align: center;
    padding-top: 50px;
}

.container {
    background-color: white;
    padding: 30px;
    border-radius: 10px;
    width: 400px;
    margin: auto;
    box-shadow: 0px 0px 10px gray;
}

textarea {
    width: 90%;
    height: 100px;
    margin-bottom: 10px;
}

input, select, button {
    width: 90%;
    margin-bottom: 10px;
    padding: 10px;
}

.result {
    margin-top: 20px;
    background-color: #e0ffe0;
    padding: 10px;
    border-radius: 8px;
}

```

Output:

Explanation:

This project is built with three main components:

1. Frontend (HTML/CSS)

- A clean, simple webpage is designed to accept:
    - The **message** to encrypt or decrypt
    - An **8-character key** (as required by DES)
    - The operation: **Encrypt** or **Decrypt**
- It provides a user-friendly interface styled with CSS to make the interaction smooth.

2. Backend (Python Flask)

- **Flask**, a lightweight Python web framework, handles the form submission.
- After receiving the input, it uses the **pyDes** library to perform DES encryption or decryption:
    - **Encryption**: Converts plain text into a secure hex string.
    - **Decryption**: Converts the hex string back into readable plain text.
- The backend then sends the result back to the frontend for display.

3. Encryption Logic

- The DES algorithm requires a **fixed 8-character key**.
- Messages are padded automatically to match DES block size requirements.
- Outputs are presented in a readable **hexadecimal format** after encryption, making it easier for users to copy and store.

Conclusion:

This project successfully demonstrates the implementation of a classic DES encryption system inside a web application, combining web development (HTML, CSS) with cybersecurity techniques (encryption algorithms).

It helps users interactively understand how messages can be encrypted and decrypted using symmetric-key cryptography.

While DES is no longer recommended for high-security systems due to its shorter key length and vulnerability to brute-force attacks, it remains a fundamental learning tool for cryptography basics.

This project serves as a foundation for further explorations into more advanced encryption algorithms like AES or RSA and introduces how web applications can be used for real-time security operations.

DES Encryption/ Decryption

DES PROJECT

NSAS_123

Encrypt

Submit

Result:

b6880bcd8f82c5a9d3005ee327482e49

DES Encryption/ Decryption

b6880bcd8f82c5a9d3005ee327482e49

NSAS_123

Decrypt

Submit

Result:

DES PROJECT