Implementation of AES Encryption Algorithm

Supervisor

Dr. Narendran Rajagopalan,

Associate Professor,

NITPY.

Assignment by,

Megavath Eshwar,

(CS22B1030).

Introduction:

In today's digital era, securing sensitive information is a top priority. The Advanced Encryption Standard (AES) is a widely adopted encryption algorithm used to secure data in various applications. This project focuses on implementing AES encryption and decryption through a simple and interactive web interface using Python (Flask), HTML, and CSS. It allows users to encrypt and decrypt textual data securely by providing a secret key, showcasing the real-world utility of cryptographic techniques in a user-friendly environment.

Implementation:

//NS_ASSIGNMENT//app.py

```python
from flask import Flask, render_template, request
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes
import base64
import hashlib

app = Flask(__name__)

def pad(text):
    while len(text) % 16 != 0:
        text += ' '
    return text

@app.route('/', methods=['GET', 'POST'])
def index():
    result = ""
    mode = ""
    if request.method == 'POST':
        key = hashlib.sha256(request.form['key'].encode()).digest()
        text = request.form['text']
        mode = request.form['mode']

        if mode == 'encrypt':
            cipher = AES.new(key, AES.MODE_ECB)
            ct_bytes = cipher.encrypt(pad(text).encode('utf-8'))
            result = base64.b64encode(ct_bytes).decode('utf-8')
        elif mode == 'decrypt':
            cipher = AES.new(key, AES.MODE_ECB)
            ct = base64.b64decode(text)
            result = cipher.decrypt(ct).decode('utf-8').strip()

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

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

//NS_ASSIGNMENT//templates//index.html

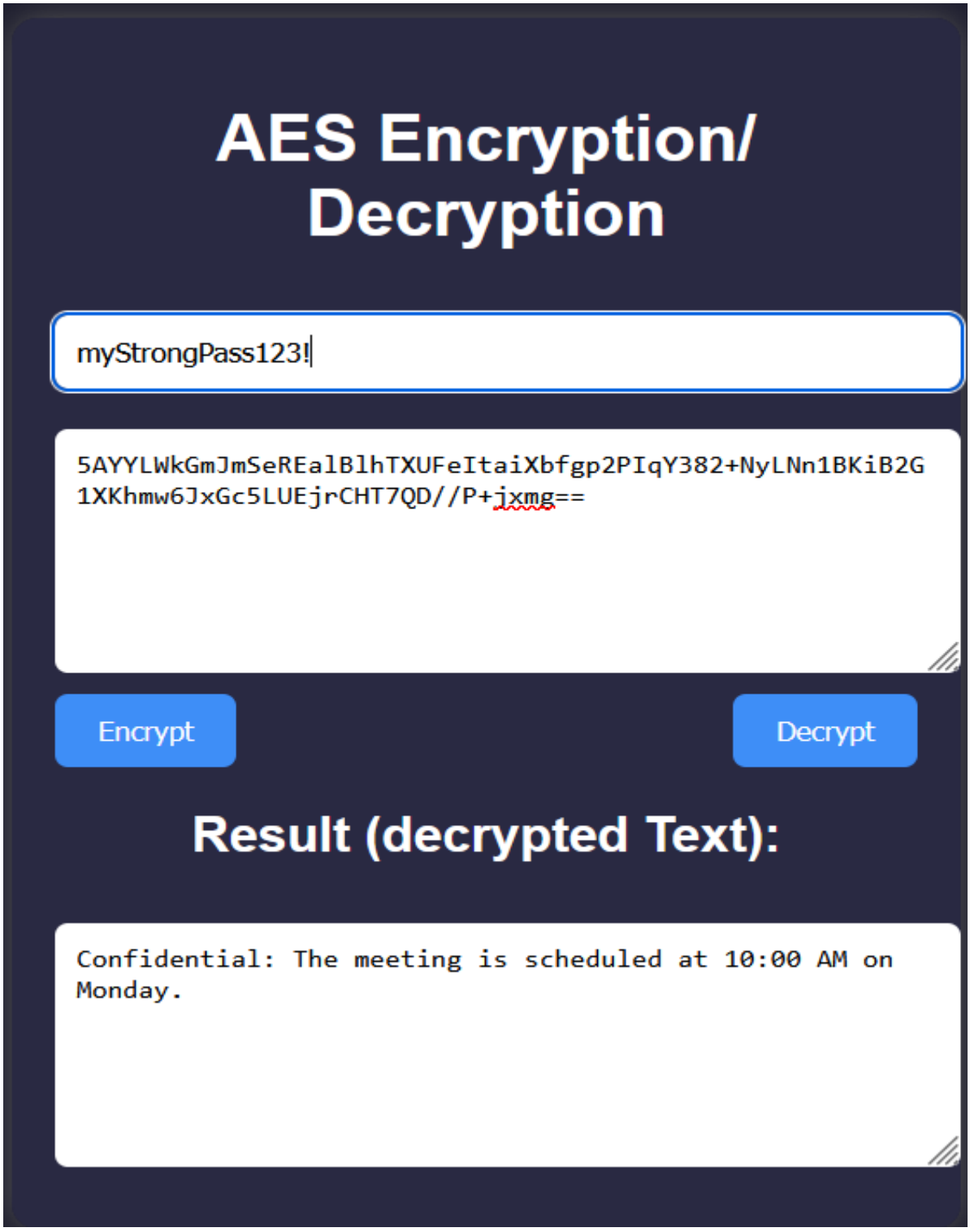
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>AES Encryption Web App</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <div class="container">
        <h1>AES Encryption/Decryption</h1>
        <form method="POST">
            <input type="text" name="key" placeholder="Enter Secret Key" required>
            <textarea name="text" placeholder="Enter text..." required></textarea>
            <div class="buttons">
                <button type="submit" name="mode" value="encrypt">Encrypt</button>
                <button type="submit" name="mode" value="decrypt">Decrypt</button>
            </div>
        </form>

        {% if result %}
            <h2>Result ({{ mode }}ed Text):</h2>
            <textarea readonly>{{ result }}</textarea>
        {% endif %}
    </div>
</body>
</html>
```

//NS_ASSIGNMENT//static//style.css

```css
body {
    font-family: Arial, sans-serif;
    background: #1e1e2f;
    color: white;
    display: flex;
    justify-content: center;
    padding: 50px;
}
.container {
    width: 400px;
    background: #292942;
    padding: 20px;
    border-radius: 12px;
    box-shadow: 0px 0px 15px #555;
}
h1, h2 {
    text-align: center;
}
input, textarea {
    width: 100%;
    margin: 10px 0;
    padding: 10px;
    border-radius: 6px;
    border: none;
}
textarea {
    height: 100px;
}
.buttons {
    display: flex;
    justify-content: space-between;
}
button {
    padding: 10px 20px;
    background-color: #3e8ef7;
    color: white;
    border: none;
    border-radius: 6px;
    cursor: pointer;
}
```

Output:


AES Encryption/ Decryption
myStrongPass123!
Confidential: The meeting is scheduled at 10:00 AM on Monday.
Encrypt
Decrypt
Result (encrypted Text):
5AYYLWkGmJmSeREalBlhTXUFeItaiXbfgp2PIqY382+NyLNn1BKiB2G1XKhmw6JxGc5LUEjrCHT7QD//P+jxmg==



Explanation:

This project is a lightweight web-based application that allows users to input plain text and a secret key to perform AES encryption or decryption. The frontend is developed using HTML and styled with CSS to provide a clean and responsive interface. The backend is powered by Python using the Flask framework, and the pycryptodome library is used for implementing AES encryption logic.

Here's how the app works:

- Users enter a text message and a secret key.
- The backend hashes the secret key using SHA-256 to generate a fixed-size key suitable for AES.
- The text is padded (to match AES block size requirements) and encrypted using AES in ECB mode.
- Encrypted data is returned in Base64 format for readability and compatibility.
- For decryption, the process is reversed: the encrypted Base64 text is decoded, decrypted, and stripped of padding.

This project demonstrates how cryptographic algorithms like AES can be effectively integrated into modern web applications to ensure data confidentiality.

Conclusion:

This AES encryption-decryption web app successfully integrates cybersecurity concepts with modern web development technologies. By combining Python for server-side processing with a responsive frontend, the project offers a practical and secure tool for encrypting and decrypting text. It not only strengthens the understanding of how AES works but also showcases the importance of data security in web applications. This implementation can serve as a foundational step toward more complex secure communication systems in real-world scenarios.