**Substitution Cipher Encryption and Decryption**

This code implements a substitution cipher algorithm for encrypting and decrypting text using a customized substitution table. The encryption process involves converting the input text into a base64-encoded string and then applying a substitution cipher using a dynamically generated substitution table. The decryption process reverses the encryption steps to retrieve the original plaintext.

**Code Overview**

The code is implemented as a Flask web application, allowing users to encrypt and decrypt text through a user interface.

**Dependencies**

The code requires the following dependencies:

- Flask: A micro web framework used for building the web application.

- hashlib: A library providing various hashing algorithms, used for generating a hash of the encryption key.

- base64: A library providing functions for base64 encoding and decoding.

Make sure to install these dependencies before running the code.

**Encryption and Decryption Functions**

The code provides two main functions for encryption and decryption:

**1. `encrypt(string)`:** This function takes a plaintext string as input and performs the encryption process. It first base64-encodes the string, converts it to a list of characters, and then applies a substitution cipher using a dynamically generated substitution table. The resulting ciphertext is returned.

**2. `decrypt(string)`:** This function takes a ciphertext string as input and performs the decryption process. It first converts the string to a list of characters, applies the reverse substitution cipher using the same substitution table, and then base64-decodes the resulting string. The original plaintext is returned.

**Substitution Table Generation**

The substitution table used in the encryption and decryption process is dynamically generated based on a provided encryption key. The key is hashed using the SHA-512 algorithm to create a hash value. This hash value is used to determine the order of characters in the substitution table.

The substitution table is initially populated with a set of characters representing the base64 encoding characters. The order of these characters is then modified based on the hash value. The algorithm performs a series of operations on the substitution table to shuffle its contents. The specific operations include inserting and popping elements at specific positions and reversing the table.

The resulting substitution table is stored in a dictionary called `sbox`. For encryption, `sbox` maps each base64 character to its substituted counterpart. For decryption, the mapping is reversed to map the substituted characters back to their original base64 characters.

**Flask Routes**

The Flask web application provides the following routes:

**1. `/`:** The home route that renders the `index.html` template, which contains the user interface for encryption and decryption.

**2. `/encrypt` (POST):** This route is triggered when the user submits the encryption form. It retrieves the plaintext from the form data, calls the `encrypt` function to encrypt the plaintext, and returns the resulting ciphertext.

**3. `/decrypt` (POST):** This route is triggered when the user submits the decryption form. It retrieves the ciphertext from the form data, calls the `decrypt` function to decrypt the ciphertext, and returns the resulting plaintext.

**User Interface**

The user interface is implemented using HTML and Flask's template engine. The `index.html` template contains an input form for both encryption and decryption. Upon submission, the form data is sent to the appropriate Flask route for encryption or decryption.

The encrypted or decrypted text is displayed in the web interface for the user to view.

**Usage**

To use the code:

1. Install the required dependencies (Flask, hashlib, base64).
2. Save the code in a Python file (e.g., `substitution\_cipher.py`).
3. Run the Python file (`python substitution\_cipher.py`).
4. Access the application in a web browser at `http://localhost:5000`.
5. Use the provided interface to encrypt and decrypt text.

**Security Considerations**

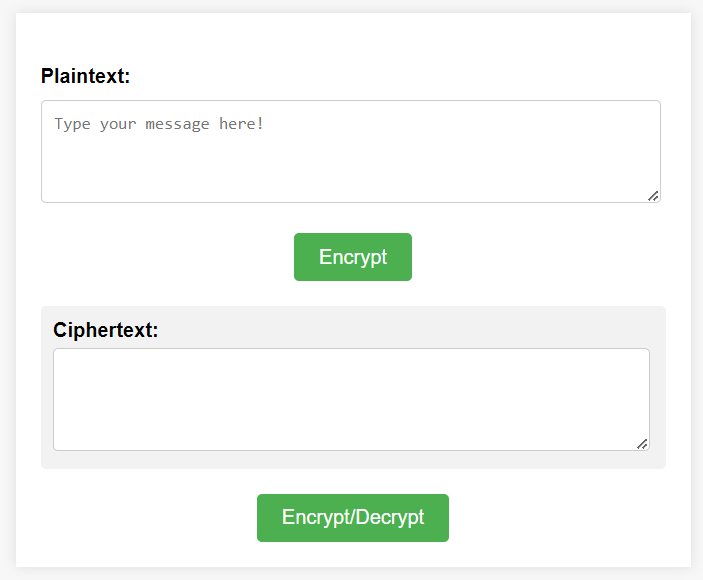
- The substitution cipher used in this code is a simple form of encryption and may not provide strong security. It is primarily meant for educational purposes and should not be used for sensitive data.

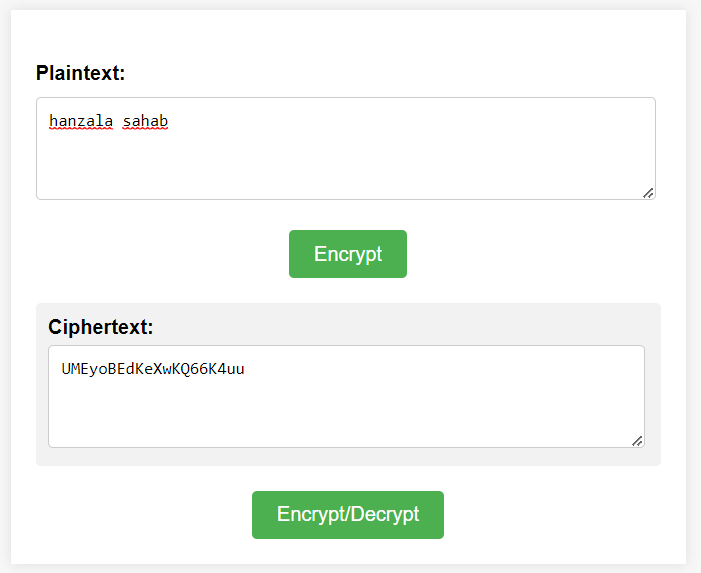
- The security of the encryption relies on the strength of the encryption key. It's important to use a strong and sufficiently complex key to ensure the security of the encrypted data.

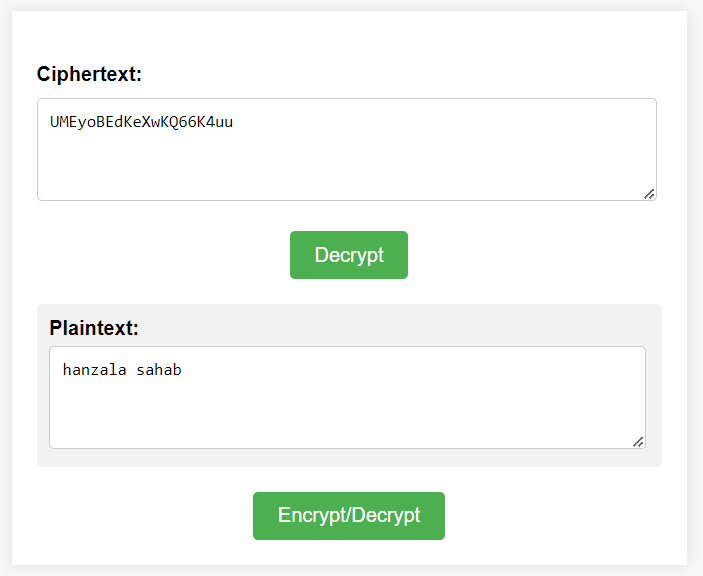
- As with any encryption algorithm, it's essential to follow best practices for key management, secure communication channels, and protection against various attack vectors.

- This code does not implement additional security measures such as authentication, access control, or input validation. Consider adding these measures if deploying the code in a production environment.

**Graphical User Interface**







**Python Code**

from flask import Flask, request, render\_template  
import base64  
import hashlib  
  
app = Flask(\_\_name\_\_)  
  
KEY = b'My5up3rC0mpl3xK3y!@#$%^'  
base64\_chars = [c for c in 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890+/=']  
  
  
def convert(string, type):  
 hash = hashlib.sha512(KEY).hexdigest()  
  
 cipher = base64\_chars[:]  
  
 for c in hash:  
 char\_int = int(c, 16)  
 pos = 65 \* (char\_int / 15)  
  
 cipher.insert(0, cipher.pop(int(pos) - 1))  
 cipher = cipher[::-1]  
  
 sbox = {}  
  
 for i, c in enumerate(base64\_chars):  
 sbox[c] = cipher[i]  
  
 if type == 'd':  
 sbox = dict((v, k) for k, v in sbox.items())  
  
 for i, c in enumerate(string):  
 string[i] = sbox[c]  
  
 return ''.join(string)  
  
  
def encrypt(string):  
 string = [c for c in base64.b64encode(string.encode()).decode()]  
  
 return convert(string, 'e')  
  
  
def decrypt(string):  
 string = [c for c in string.strip()]  
  
 return base64.b64decode(convert(string, 'd')).decode()  
  
  
@app.route('/')  
def index():  
 return render\_template('index.html')  
  
  
@app.route('/encrypt', methods=['POST'])  
def encrypt\_route():  
 plaintext = request.form['plaintext']  
 ciphertext = encrypt(plaintext)  
 return ciphertext  
  
  
@app.route('/decrypt', methods=['POST'])  
def decrypt\_route():  
 ciphertext = request.form['ciphertext']  
 plaintext = decrypt(ciphertext)  
 return plaintext  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run()

**HTML Code**

<!DOCTYPE html>  
<html>  
<head>  
 <title>Substitution Cipher</title>  
 <link rel="stylesheet" type="text/css" href="{{ url\_for('static', filename='style.css') }}">  
</head>  
<body>  
 <div class="container">  
 <h1></h1>  
   
 <div id="encrypt-section">  
 <form id="encrypt-form" onsubmit="event.preventDefault(); encrypt();" method="POST">  
 <label for="plaintext">Plaintext:</label>  
 <textarea id="plaintext" name="plaintext" rows="4" class="plain-textarea" placeholder="Type your message here!"></textarea>  
   
 <div class="button-container">  
 <button class="button" type="submit">Encrypt</button>  
 </div>  
   
 <div class="result-container">  
 <label class="result-label" for="ciphertext">Ciphertext:</label>  
 <textarea id="ciphertext" name="ciphertext" rows="4" readonly class="result-textarea"></textarea>  
 </div>  
 </form>  
 </div>  
   
 <div id="decrypt-section" style="display: none;">  
 <form id="decrypt-form" onsubmit="event.preventDefault(); decrypt();" method="POST">  
 <label for="ciphertext-dec">Ciphertext:</label>  
 <textarea id="ciphertext-dec" name="ciphertext" rows="4" class="plain-textarea" placeholder="Please enter your ciphertext here!"></textarea>  
   
 <div class="button-container">  
 <button class="button" type="submit">Decrypt</button>  
 </div>  
   
 <div class="result-container">  
 <label class="result-label" for="plaintext-dec">Plaintext:</label>  
 <textarea id="plaintext-dec" name="plaintext" rows="4" readonly class="result-textarea"></textarea>  
 </div>  
 </form>  
 </div>  
   
 <div class="toggle-container">  
 <button class="button" onclick="switchSection()">Encrypt/Decrypt</button>  
 </div>  
 </div>  
   
 <script src="{{ url\_for('static', filename='script.js') }}"></script>  
</body>  
</html>

**CSS Code**

body {  
 font-family: Arial, sans-serif;  
 background-color: #f7f7f7;  
}  
  
.container {  
 max-width: 500px;  
 margin: 80px auto;  
 padding: 20px;  
 background-color: #fff;  
 box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);  
}  
  
h1 {  
 text-align: center;  
 color: #333;  
}  
  
label {  
 display: block;  
 margin-bottom: 10px;  
 font-weight: bold;  
}  
  
input[type="text"], textarea {  
 width: 100%;  
 padding: 10px;  
 border: 1px solid #ccc;  
 border-radius: 4px;  
 resize: vertical;  
}  
  
.button-container {  
 text-align: center;  
 margin-top: 20px;  
}  
  
.button {  
 display: inline-block;  
 padding: 10px 20px;  
 background-color: #4CAF50;  
 color: #fff;  
 border: none;  
 border-radius: 4px;  
 cursor: pointer;  
 font-size: 16px;  
 transition: background-color 0.3s ease;  
}  
  
.button:hover {  
 background-color: #45a049;  
}  
  
.result-container {  
 margin-top: 20px;  
 background-color: #f2f2f2;  
 padding: 10px;  
 border-radius: 4px;  
}  
  
.result-label {  
 display: block;  
 font-weight: bold;  
 margin-bottom: 5px;  
}  
.plain-textarea{  
 width: 95%;  
}  
.result-textarea {  
 width: 95%;  
 padding: 10px;  
 border: 1px solid #ccc;  
 border-radius: 4px;  
 resize: vertical;  
}  
  
.toggle-container {  
 text-align: center;  
 margin-top: 20px;  
}

**JavaScript Code**

function switchSection() {  
 var encryptSection = document.getElementById('encrypt-section');  
 var decryptSection = document.getElementById('decrypt-section');  
   
 if (encryptSection.style.display === 'none') {  
 encryptSection.style.display = 'block';  
 decryptSection.style.display = 'none';  
 document.getElementById('ciphertext').value = '';  
 document.getElementById('plaintext').value = '';  
 } else {  
 encryptSection.style.display = 'none';  
 decryptSection.style.display = 'block';  
 document.getElementById('ciphertext-dec').value = '';  
 document.getElementById('plaintext-dec').value = '';  
 }  
}  
  
function encrypt() {  
 var plaintext = document.getElementById('plaintext').value;  
   
 fetch('/encrypt', {  
 method: 'POST',  
 headers: {  
 'Content-Type': 'application/x-www-form-urlencoded'  
 },  
 body: 'plaintext=' + encodeURIComponent(plaintext)  
 })  
 .then(response => response.text())  
 .then(ciphertext => {  
 document.getElementById('ciphertext').value = ciphertext;  
 })  
 .catch(error => console.error('Error:', error));  
}  
  
function decrypt() {  
 var ciphertext = document.getElementById('ciphertext-dec').value;  
   
 fetch('/decrypt', {  
 method: 'POST',  
 headers: {  
 'Content-Type': 'application/x-www-form-urlencoded'  
 },  
 body: 'ciphertext=' + encodeURIComponent(ciphertext)  
 })  
 .then(response => response.text())  
 .then(plaintext => {  
 document.getElementById('plaintext-dec').value = plaintext;  
 })  
 .catch(error => console.error('Error:', error));  
}