Text

Description automatically generated

**Activity based**

**Lab Cie Report on**

**Application Security**

**Submitted to Vishwakarma University, Pune**

**Under the Initiative of**

**Contemporary Curriculum, Pedagogy, and Practice (C2P2)**

**By**

**Student Name : Alfiya Sayyed**

**Student Name : Smeet Shah**

**Student Name : Avani Joshi**

**SRN No : 202202232**

**SRN No : 202202260**

**SRN No : 202202273**

**Roll No : 58**

**Roll No : 68**

**Roll No : 72**

**Div : F**

**Second Year Engineering**

**Department of Computer Engineering**

**Faculty of Science and Technology**

**Academic Year**

**2023-2024**

**Project Statement:**

Develop a web-based cryptography application that allows users to encrypt and decrypt text using various cipher techniques. The application should provide a user-friendly interface for selecting the encryption or decryption operation, choosing the cipher type, and entering the input text along with relevant keys. It should support multiple cipher techniques, including the Vigenère Cipher, Substitution Cipher, Transposition Cipher, and Caesar Cipher. Security considerations such as key management, input validation, and protection against cryptographic attacks should be addressed. The goal is to create a robust and efficient system that enables users to securely encrypt and decrypt sensitive information for communication and data protection purposes.

**Introduction:**

In today's digital era, ensuring the confidentiality and integrity of sensitive information is paramount. Cryptography serves as a cornerstone in achieving this objective by providing techniques for secure communication and data protection. By transforming plaintext into ciphertext using cryptographic algorithms, sensitive data can be shielded from unauthorized access and tampering.

To leverage the power of cryptography, we introduce a web-based cryptography application. This application empowers users to encrypt and decrypt text using various cipher techniques, including the Vigenère Cipher, Substitution Cipher, Transposition Cipher, and Caesar Cipher. With a user-friendly interface, users can select the desired operation, cipher type, and input text, enabling seamless cryptographic operations.

Through this application, users can harness the power of cryptography to safeguard their communications and sensitive information, ensuring confidentiality and integrity in an increasingly interconnected digital world.

**Cipher Techniques:**

* Vigenère Cipher: Uses a keyword to shift letters in the plaintext.
* Substitution Cipher: Replaces each letter in the plaintext with another letter based on a substitution key.
* Transposition Cipher: Rearranges characters in the plaintext according to a specific key.
* Caesar Cipher: Shifts each letter in the plaintext a fixed number of positions down or up the alphabet.

**Code Structure:**

* Cipher.py: Contains functions for each cipher technique.
* Main Function: Handles user input for encryption or decryption.

**Web Application with Flask:**

* app.py: Implements Flask routes for rendering pages and handling form submissions.

**User Interface:**

* index.html: Provides a form for selecting the operation, cipher type, and input text.
* result.html: Displays the result of the encryption or decryption operation.

**Security Considerations:**

* Key management, input validation, and protection against cryptographic attacks are essential security aspects.

**Methodology:**

1. **Cipher Implementation:**
   * The project implements the following classical ciphers:
     + Caesar Cipher: Shifts each letter by a fixed number of positions in the alphabet.
     + Vigenère Cipher: Uses a keyword to encrypt alphabetic text by shifting letters based on a Vigenère table.
     + Substitution Cipher: Replaces each plaintext character with another character based on a substitution key.
     + Transposition Cipher: Rearranges the plaintext characters based on a given key.
2. **Web Application Development:**
   * The Flask framework is employed to develop a web application.
   * The application provides an intuitive interface for users to select the operation (encryption or decryption) and the desired cipher.
   * Users input the plaintext or ciphertext, along with any required parameters (e.g., keyword, shift value).
   * Upon submission, the application performs the encryption or decryption based on the chosen cipher and displays the result to the user.

**Implementation:**

**Cipher Implementation:**

**Caesar Cipher:**

* Involves shifting each letter in the plaintext by a fixed number of positions in the alphabet.
* Two functions are implemented: one for encryption and one for decryption.

**Vigenère Cipher:**

* Utilizes a keyword to determine the amount of shifting applied to each letter in the plaintext.
* Encryption and decryption functions are implemented, both relying on the keyword for processing.

**Substitution Cipher:**

* Replaces each letter in the plaintext with a corresponding letter from a substitution key.
* Separate functions for encryption and decryption are implemented, utilizing the substitution key.

**Transposition Cipher:**

* Rearranges the characters in the plaintext based on a given key.
* Encryption and decryption functions are provided, both operating according to the transposition key.

**2. Web Application Development:**

**Flask Integration:**

* Utilizes the Flask framework to develop a web-based application.
* Routes are defined to handle user requests for encryption and decryption.

**User Interface (HTML Templates):**

* Provides a simple HTML form where users can select the operation (encryption or decryption), choose a cipher, and input the plaintext or ciphertext.
* Additional input fields are provided for cipher-specific parameters such as keyword, substitution key, and transposition key.

**Processing Requests:**

* Upon form submission, the Flask backend processes the request, calls the appropriate cipher function based on user input, and returns the result to the user interface for display.

**Code:**

**1)Cipher.py**

# Function to generate Vigenère table def generate\_vigenere\_table():

table = [[chr((i + j) % 26 + ord('A')) for j in range(26)] for i in range(26)] return table

# Function to encrypt plaintext using Vigenère Cipher def encrypt\_vigenere\_cipher(plaintext, keyword):

table = generate\_vigenere\_table() encrypted\_text = "" keyword = keyword.upper() keyword\_index = 0

for char in plaintext: if char.isalpha():

row = ord(keyword[keyword\_index % len(keyword)]) - ord('A') col = ord(char.upper()) - ord('A') encrypted\_text += table[row][col] keyword\_index += 1 else:

encrypted\_text += char return encrypted\_text

# Function to decrypt ciphertext using Vigenère Cipher def decrypt\_vigenere\_cipher(ciphertext, keyword):

table = generate\_vigenere\_table() decrypted\_text = "" keyword = keyword.upper() keyword\_index = 0 for char in ciphertext: if char.isalpha():

row = ord(keyword[keyword\_index % len(keyword)]) - ord('A') col = table[row].index(char.upper()) decrypted\_text += chr(col + ord('A')) keyword\_index += 1 else:

decrypted\_text += char return decrypted\_text

# Function to encrypt plaintext using Substitution Cipher def encrypt\_substitution\_cipher(plaintext, substitution\_key):

substitution\_key = substitution\_key.upper()

substitution\_dict = {chr(ord('A') + i): substitution\_key[i] for i in range(26)} encrypted\_text = "".join(substitution\_dict.get(char.upper(), char) for char in plaintext) return encrypted\_text

# Function to decrypt ciphertext using Substitution Cipher def decrypt\_substitution\_cipher(ciphertext, substitution\_key):

substitution\_key = substitution\_key.upper()

substitution\_dict = {substitution\_key[i]: chr(ord('A') + i) for i in range(26)} decrypted\_text = "".join(substitution\_dict.get(char.upper(), char) for char in ciphertext) return decrypted\_text

# Function to encrypt plaintext using Transposition Cipher def encrypt\_transposition\_cipher(plaintext, key):

key = int(key) encrypted\_text = "" for i in range(key): encrypted\_text += plaintext[i::key] return encrypted\_text

# Function to decrypt ciphertext using Transposition Cipher def decrypt\_transposition\_cipher(ciphertext, key):

key = int(key) decrypted\_text = [''] \* key for i in range(len(ciphertext)): decrypted\_text[i % key] += ciphertext[i] return ''.join(decrypted\_text)

# Function to encrypt plaintext using Caesar Cipher def encrypt\_caesar\_cipher(plaintext, shift):

ciphertext = "" for char in plaintext: if char.isalpha():

shifted\_ascii = ord(char) + shift if char.islower(): if shifted\_ascii > ord('z'): shifted\_ascii -= 26 elif char.isupper(): if shifted\_ascii > ord('Z'): shifted\_ascii -= 26 ciphertext += chr(shifted\_ascii) else:

ciphertext += char return ciphertext

# Function to decrypt ciphertext using Caesar Cipher def decrypt\_caesar\_cipher(ciphertext, shift):

decrypted\_text = "" for char in ciphertext: if char.isalpha():

shifted\_ascii = ord(char) - shift if char.islower():

if shifted\_ascii < ord('a'): shifted\_ascii += 26 elif char.isupper(): if shifted\_ascii < ord('A'): shifted\_ascii += 26 decrypted\_text += chr(shifted\_ascii) else:

decrypted\_text += char return decrypted\_text

# Main function def main():

choice = input("Enter 'E' for encryption or 'D' for decryption: ").upper() if choice == 'E':

cipher\_choice = input("Enter 'V' for Vigenère Cipher, 'S' for Substitution Cipher, 'T' for

Transposition Cipher, or 'C' for Caesar Cipher: ").upper() plaintext = input("Enter the plaintext: ") if cipher\_choice == 'V':

keyword = input("Enter the keyword: ")

encrypted\_text = encrypt\_vigenere\_cipher(plaintext, keyword) print("Encrypted text:", encrypted\_text) elif cipher\_choice == 'S':

substitution\_key = input("Enter the substitution key: ")

encrypted\_text = encrypt\_substitution\_cipher(plaintext, substitution\_key) print("Encrypted text:", encrypted\_text) elif cipher\_choice == 'T':

key = input("Enter the transposition key (integer): ") encrypted\_text = encrypt\_transposition\_cipher(plaintext, key) print("Encrypted text:", encrypted\_text) elif cipher\_choice == 'C':

shift = int(input("Enter the shift value: "))

encrypted\_text = encrypt\_caesar\_cipher(plaintext, shift) print("Encrypted text:", encrypted\_text) else:

print("Invalid choice.") elif choice == 'D':

cipher\_choice = input("Enter 'V' for Vigenère Cipher, 'S' for Substitution Cipher, 'T' for

Transposition Cipher, or 'C' for Caesar Cipher: ").upper() ciphertext = input("Enter the ciphertext: ") if cipher\_choice == 'V':

keyword = input("Enter the keyword: ")

decrypted\_text = decrypt\_vigenere\_cipher(ciphertext, keyword) print("Decrypted text:", decrypted\_text) elif cipher\_choice == 'S':

substitution\_key = input("Enter the substitution key: ")

decrypted\_text = decrypt\_substitution\_cipher(ciphertext, substitution\_key) print("Decrypted text:", decrypted\_text) elif cipher\_choice == 'T':

key = input("Enter the transposition key (integer): ") decrypted\_text = decrypt\_transposition\_cipher(ciphertext, key) print("Decrypted text:", decrypted\_text) elif cipher\_choice == 'C':

shift = int(input("Enter the shift value: "))

decrypted\_text = decrypt\_caesar\_cipher(ciphertext, shift) print("Decrypted text:", decrypted\_text) else: print("Invalid choice.") else:

print("Invalid choice.")

if \_\_name\_\_ == "\_main\_": main()

**2)app.py**

from flask import Flask, render\_template, request from cipher import \*

app = Flask(\_\_name\_\_)

@app.route('/') def index():

return render\_template('index.html')

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

operation = request.form['operation'] cipher = request.form['cipher'] input\_text = request.form['plaintext'] result\_text = ""

if cipher == 'caesar':

shift = int(request.form['shift']) if operation == 'encrypt':

result\_text = encrypt\_caesar\_cipher(input\_text, shift) elif operation == 'decrypt':

result\_text = decrypt\_caesar\_cipher(input\_text, shift) elif cipher == 'vigenere':

keyword = request.form['keyword'] if operation == 'encrypt':

result\_text = encrypt\_vigenere\_cipher(input\_text, keyword) elif operation == 'decrypt':

result\_text = decrypt\_vigenere\_cipher(input\_text, keyword) elif cipher == 'substitution':

substitution\_key = request.form['substitution\_key'] if operation == 'encrypt':

result\_text = encrypt\_substitution\_cipher(input\_text, substitution\_key) elif operation == 'decrypt':

result\_text = decrypt\_substitution\_cipher(input\_text, substitution\_key) elif cipher == 'transposition':

transposition\_key = request.form['transposition\_key'] if operation == 'encrypt':

result\_text = encrypt\_transposition\_cipher(input\_text, transposition\_key) elif operation == 'decrypt':

result\_text = decrypt\_transposition\_cipher(input\_text, transposition\_key)

return render\_template('result.html', operation=operation, cipher=cipher, input\_text=input\_text, result\_text=result\_text)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

**3)index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Cryptography Tool</title>

<link rel="stylesheet" type="text/css" href="static\style.css">

</head>

<body>

<center><h1>Cryptography Tool</h1></center>

<form action="/encrypt\_decrypt" method="post">

<label for="operation">Select operation:</label><br>

<select id="operation" name="operation">

<option value="encrypt">Encrypt</option>

<option value="decrypt">Decrypt</option>

</select><br><br>

<label for="cipher">Select cipher:</label><br>

<select id="cipher" name="cipher">

<option value="caesar">Caesar Cipher</option>

<option value="vigenere">Vigenère Cipher</option>

<option value="substitution">Substitution Cipher</option>

<option value="transposition">Transposition Cipher</option>

</select><br><br>

<label for="plaintext">Enter plaintext:</label><br>

<textarea id="plaintext" name="plaintext" rows="4" cols="50"></textarea><br><br>

<label for="keyword">Enter keyword (for Vigenère Cipher):</label><br>

<input type="text" id="keyword" name="keyword"><br><br>

<label for="substitution\_key">Enter substitution key (for Substitution Cipher):</label><br>

<input type="text" id="substitution\_key" name="substitution\_key"><br><br>

<label for="transposition\_key">Enter transposition key (for Transposition Cipher):</label><br>

<input type="number" id="transposition\_key" name="transposition\_key"><br><br>

<label for="shift">Enter shift value (for Caesar Cipher):</label><br>

<input type="number" id="shift" name="shift"><br><br>

<input type="submit" value="Submit">

</form>

</body>

</html>

**4)results.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Cryptography Result</title>

<link rel="stylesheet" type="text/css" href="static\style1.css">

</head>

<body>

<center>

<h1>Cryptography Result</h1>

<h2>Operation: {{ operation }}</h2>

<h2>Cipher: {{ cipher }}</h2>

<h3>Plaintext/Ciphertext:</h3>

<p>{{ input\_text }}</p>

<h3>Result:</h3>

<p>{{ result\_text }}</p>

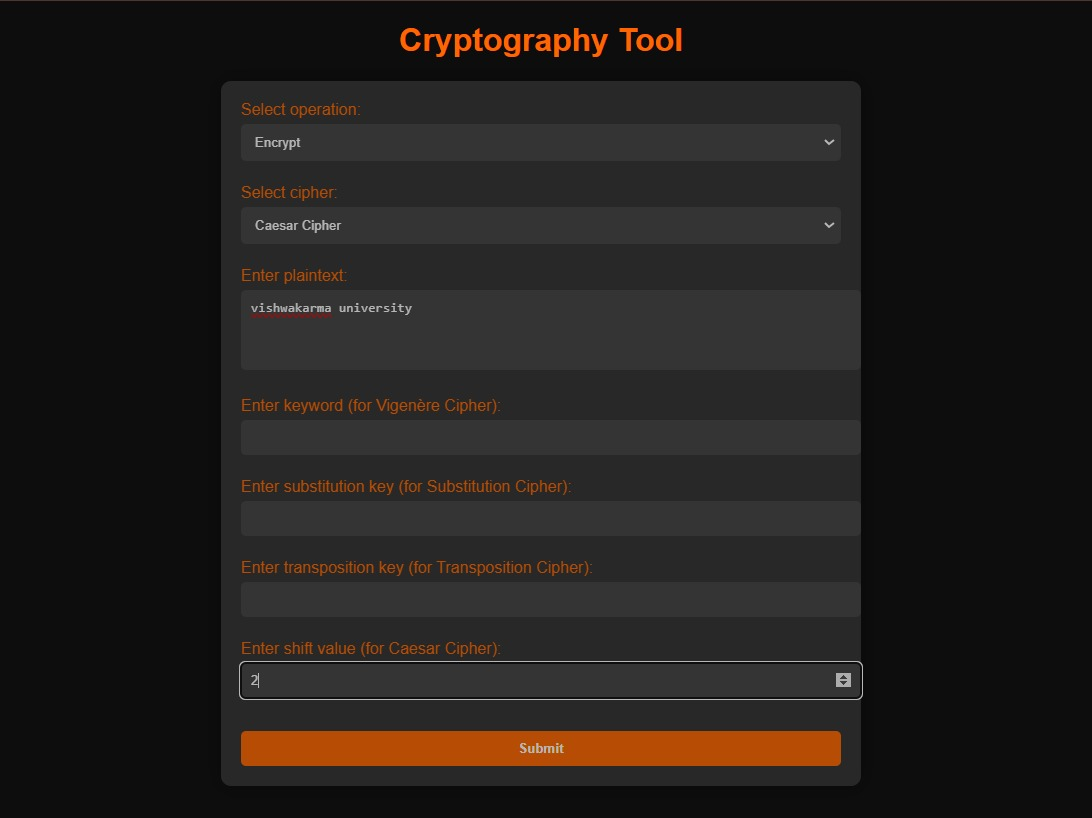
<a href="/">Back to Home</a>

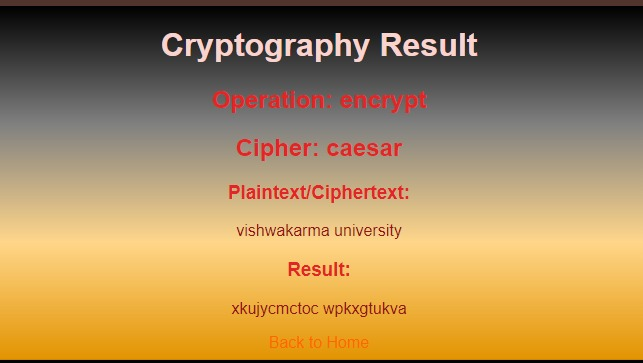
</center>

</body>

</html>

**Output:**

****

****

**Conclusion :**

The implementation of classical cryptographic ciphers along with the development of a user-friendly web interface enables users to encrypt and decrypt messages with ease. This tool enhances awareness and understanding of cryptographic techniques while providing practical means for secure communication. Further enhancements and additional features can be incorporated to extend the functionality of the tool in the future.