**IS LAB\_5**

**~ Prof. Dr. Aashka Raval**

✅ **Title 4: Implementation of Vigenère Cipher (Standard)**

🎯 **Objective:**  
• To implement the classical Vigenère Cipher encryption and decryption technique.  
• To understand how polyalphabetic substitution enhances the security compared to monoalphabetic ciphers (like Caesar Cipher).

📘 **Introduction:**  
The **Vigenère Cipher** is a classical polyalphabetic substitution cipher, which uses a keyword to determine the shift of each letter in the plaintext.  
Unlike the Caesar Cipher that applies a single fixed shift, the Vigenère Cipher applies different shifts based on repeating characters of the key. This makes it more secure against frequency analysis.

For example, using key = "LEMON" to encrypt plaintext = "ATTACKATDAWN", each letter is shifted by the alphabetical index of the corresponding key character (A=0, B=1, ..., Z=25).

📚 **Concepts Used:**  
• Symmetric Key Cryptography  
• Polyalphabetic Substitution Cipher  
• Modular Arithmetic (mod 26)  
• Repeating Keyword Mechanism  
• Case Insensitivity Handling

🧠 **Logic:**

1. **Encryption:**  
   • Extend the keyword to match the length of the plaintext.  
   • For each character in plaintext:  
   – Convert both plaintext and key character to numerical form (A=0, B=1, ..., Z=25).  
   – Add their values modulo 26.  
   – Convert back to a character.  
   • Concatenate all encrypted characters to form the ciphertext.
2. **Decryption:**  
   • Extend the keyword to match the length of the ciphertext.  
   • For each character in ciphertext:  
   – Convert both ciphertext and key character to numerical form.  
   – Subtract key value from ciphertext value, add 26 if negative, then take modulo 26.  
   – Convert back to a character.  
   • Concatenate all decrypted characters to recover the plaintext.

💻 **Python Code:**

def generate\_key(text, key):

key = list(key)

if len(text) == len(key):

return "".join(key)

else:

for i in range(len(text) - len(key)):

key.append(key[i % len(key)])

return "".join(key)

def encrypt\_vigenere(text, key):

cipher\_text = []

for i in range(len(text)):

if text[i].isalpha():

x = (ord(text[i].upper()) + ord(key[i].upper())) % 26

x += ord('A')

cipher\_text.append(chr(x))

else:

cipher\_text.append(text[i])

return "".join(cipher\_text)

def decrypt\_vigenere(cipher\_text, key):

orig\_text = []

for i in range(len(cipher\_text)):

if cipher\_text[i].isalpha():

x = (ord(cipher\_text[i].upper()) - ord(key[i].upper()) + 26) % 26

x += ord('A')

orig\_text.append(chr(x))

else:

orig\_text.append(cipher\_text[i])

return "".join(orig\_text)

text = input("Enter the message: ").replace(" ", "")

keyword = input("Enter the key: ")

key = generate\_key(text, keyword)

cipher\_text = encrypt\_vigenere(text, key)

print("\nEncrypted Text:", cipher\_text)

decrypted\_text = decrypt\_vigenere(cipher\_text, key)

print("Decrypted Text:", decrypted\_text)

🧪 **Sample Output:**

Enter the message: ATTACKATDAWN

Enter the key: LEMON

Encrypted Text: LXFOPVEFRNHR

Decrypted Text: ATTACKATDAWN

✅ **Conclusion:**  
The Vigenère Cipher demonstrates how polyalphabetic substitution increases security compared to monoalphabetic ciphers.  
By using a keyword to shift letters differently at each position, it resists simple frequency analysis and introduces stronger confusion in ciphertext.  
This cipher forms the basis of more modern symmetric encryption systems.