Academic Year: 2024-25	**Programme: BTECH-Cyber (CSE)**
Year: 2nd	**Semester: IV**
Student Name: Arjun Mehta	**Batch : K1**
Roll No:K036	**Date of experiment: 23-01-25**
Faculty: Rejo Mathew	**Signature with Date:**

Experiment 4: Hill Cipher

Aim: Write a program to implement Hill Cipher.

Learning Outcomes:
After completion of this experiment, student should be able to
1. Describe working of Hill Cipher.
2. Understand application of Hill Cipher along with its advantage and limitations.

Theory:

Hill cipher is a polygraphic substitution cipher based on linear algebra. Each letter is represented by a number modulo 26. To encrypt a message, each block of n letters (considered as an n-component vector) is multiplied by an invertible n × n matrix, against modulus 26. To decrypt the message, each block is multiplied by the inverse of the matrix used for encryption. The matrix used for encryption is the cipher key, and it should be chosen randomly from the set of invertible n × n matrices (modulo 26).

Alphabet Codings

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Inverses in Z_{26}

a	1	3	5	7	11	17	25
a^{-1}	1	9	21	15	19	23	25

Example:

Plaintext = HELP

$K = \begin{bmatrix} 3 & 3 \\ 2 & 5 \end{bmatrix}$

Encryption

C = K * P mod 26

We can take only 2 x 2 matrix so we take first two letters “HE” first

$= \begin{bmatrix} 3 & 3 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} 7 \\ 4 \end{bmatrix} \text{ mod } 26$

$= \begin{bmatrix} 3*7+3*4 \\ 2*7+5*4 \end{bmatrix} \text{ mod } 26$

$= \begin{bmatrix} 33 \\ 34 \end{bmatrix} \text{ mod } 26$

$= \begin{bmatrix} 7 \\ 8 \end{bmatrix}$ = HI

Now we take next two letters “LP”

$= \begin{bmatrix} 3 & 3 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} 11 \\ 15 \end{bmatrix} \text{ mod } 26$

$= \begin{bmatrix} 3*11+3*15 \\ 2*11+5*15 \end{bmatrix} \text{ mod } 26$

$= \begin{bmatrix} 78 \\ 97 \end{bmatrix} \text{ mod } 26$

$= \begin{bmatrix} 0 \\ 19 \end{bmatrix}$ = AT

So for Plaintext **HELP** the Ciphertext is **HIAT**

Decryption

$P = K^{-1} * C \text{ mod } 26$

To find K^{-1} we need to use the formula

$K^{-1} = 1 / |K| * \text{adj } K \quad \text{mod } 26$

$K = \begin{bmatrix} 3 & 3 \\ 2 & 5 \end{bmatrix}$

|K| = [3*5-3*2] = [15 - 6] = 9

Need to find adjoint K

K = $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$

= $\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$

= $\begin{bmatrix} 5 & -3 \\ -2 & 3 \end{bmatrix}$

= $\begin{bmatrix} 5 & 23 \\ 24 & 3 \end{bmatrix}$ \\ Convert negative to positive by adding 26. E.g. -3 +26= 23, -2+26 = 24

= 1/9 * $\begin{bmatrix} 5 & 23 \\ 24 & 3 \end{bmatrix}$ mod26

=3*$\begin{bmatrix} 5 & 23 \\ 24 & 3 \end{bmatrix}$ mod26

= $\begin{bmatrix} 15 & 69 \\ 72 & 9 \end{bmatrix}$ mod26

K^{-1} = $\begin{bmatrix} 15 & 17 \\ 20 & 9 \end{bmatrix}$

P = $\begin{bmatrix} 15 & 17 \\ 20 & 9 \end{bmatrix}$ $\begin{bmatrix} 7 \\ 8 \end{bmatrix}$ mod26

= $\begin{bmatrix} 15*7+17*8 \\ 20*7+9*8 \end{bmatrix}$ mod26

= $\begin{bmatrix} 241 \\ 212 \end{bmatrix}$ mod26

= $\begin{bmatrix} 7 \\ 4 \end{bmatrix}$ = HE

Next two letters

= $\begin{bmatrix} 15 & 17 \\ 20 & 9 \end{bmatrix}$ $\begin{bmatrix} 0 \\ 19 \end{bmatrix}$ mod26

= $\begin{bmatrix} 15*0+17*19 \\ 20*0+9*19 \end{bmatrix}$ mod26

= $\begin{bmatrix} 323 \\ 171 \end{bmatrix}$ mod26

= $\begin{bmatrix} 11 \\ 15 \end{bmatrix}$ = LP

So for Ciphertext is **HIAT**, the Plaintext is **HELP**

Code: ***type or copy your completed working code here (only 2 x 2 matrix to be taken)***

In [8]:

```
import numpy as np

# Function to convert a letter to its corresponding number
def letter_to_number(letter):
    return ord(letter) - ord('A')

# Function to convert a number to its corresponding letter
def number_to_letter(number):
    return chr(number + ord('A'))
```

In [9]:

```
# Function to find the modular inverse of a number modulo 26
def mod_inverse(a, m):
    for x in range(1, m):
        if (a * x) % m == 1:
            return x
    return -1

# Function to find the inverse of a 2x2 matrix modulo 26
def matrix_inverse_2x2(matrix):
    det = int(np.round(np.linalg.det(matrix))) % 26
    det_inv = mod_inverse(det, 26)
    if det_inv == -1:
        raise ValueError("Matrix is not invertible")

    adjugate_matrix = np.array([[matrix[1, 1], -matrix[0, 1]],
                                [-matrix[1, 0], matrix[0, 0]]]) % 26
    inverse_matrix = (det_inv * adjugate_matrix) % 26
    return inverse_matrix

# Function to check if the key matrix is invertible
def is_invertible(matrix):
    try:
        matrix_inverse_2x2(matrix)
        return True
    except ValueError:
        return False
```

In [10]:

```
# Function to encrypt a block using the Hill cipher
def hill_encrypt_block(block, key_matrix):
    block_vector = np.array([letter_to_number(char) for char in block])
    encrypted_vector = np.dot(key_matrix, block_vector) % 26
    return ''.join(number_to_letter(num) for num in encrypted_vector)

# Function to decrypt a block using the Hill cipher
def hill_decrypt_block(block, key_matrix):
    inverse_key_matrix = matrix_inverse_2x2(key_matrix)
    block_vector = np.array([letter_to_number(char) for char in block])
```

```
    decrypted_vector = np.dot(inverse_key_matrix, block_vector) % 26
    return ''.join(number_to_letter(num) for num in decrypted_vector)
```

In [11]:

```
# Function to process text in blocks of 2
def process_text(text, key_matrix, is_encrypt=True):
    result = ""
    # Pad the text if its length is not even
    if len(text) % 2 != 0:
        text += 'X'
    for i in range(0, len(text), 2):
        block = text[i:i+2]
        if is_encrypt:
            result += hill_encrypt_block(block, key_matrix)
        else:
            result += hill_decrypt_block(block, key_matrix)
    return result
```

In [12]:

```
# Function to check if the key matrix is invertible
def is_invertible(matrix):
    try:
        matrix_inverse_2x2(matrix)
        return True
    except ValueError:
        return False

# Taking user input for plaintext
plaintext = input("Enter the plaintext (in uppercase, without spaces):
").replace(" ", "").upper()

# Taking user input for the key matrix
key_matrix = np.zeros((2, 2), dtype=int)
while True:
    print("Enter the key matrix (2x2) values:")
    for i in range(2):
        for j in range(2):
            key_matrix[i, j] = int(input(f"Enter value for element
[{i}][{j}]: "))
    if is_invertible(key_matrix):
        break
    else:
        print("The key matrix is not invertible. Please enter a different key
matrix.")

# Encrypting the plaintext
ciphertext = process_text(plaintext, key_matrix, is_encrypt=True)
print("Ciphertext:", ciphertext)

# Decrypting the ciphertext
decrypted_text = process_text(ciphertext, key_matrix, is_encrypt=False)
print("Decrypted text:", decrypted_text)
```

Note: Code should have proper comments

Output:

```
Enter the plaintext (in uppercase, without spaces): HELP
Enter the key matrix (2x2) values:
Enter value for element [0][0]: 3
Enter value for element [0][1]: 3
Enter value for element [1][0]: 2
Enter value for element [1][1]: 5
Ciphertext: HIAT
Decrypted text: HELP
```

Questions:

1. List advantages and limitations of Hill Cipher.

Advantages of Hill Cipher:

-Strong Security Through Linear Algebra:
Hill cipher uses matrix multiplication which makes it more secure compared to simple substitution ciphers. It provides diffusion, spreading the influence of each plaintext letter over several ciphertext letters.

-Resistance to Frequency Analysis:
Since the Hill cipher uses multiple letters in its algorithm, it is less susceptible to frequency analysis, a common attack method on classical ciphers.

-Mathematical Elegance:
The use of linear algebra provides a mathematically elegant approach to encryption and decryption, leveraging matrix operations and modular arithmetic.

-Efficiency:
With proper implementation, the Hill cipher can be efficient in both software and hardware, making it suitable for environments where computational resources are limited.

Limitations of Hill Cipher:
Key Management:

The key in Hill cipher is a matrix, which can be difficult to manage and distribute securely. The key must be invertible, adding an additional constraint.

Vulnerability to Known Plaintext Attacks:

If an attacker has access to a sufficient amount of plaintext and corresponding ciphertext, they can potentially reconstruct the key matrix using linear algebra techniques.
Fixed Block Size:

The Hill cipher operates on fixed-size blocks of text, which can make it less flexible compared to stream ciphers or ciphers that can handle variable-length data.
Complexity in Key Generation:

Generating a suitable key matrix that is invertible modulo 26 can be complex and requires additional computational steps.

2. Describe in which applications this cipher could be used.

Applications of Hill Cipher:

Educational Use:
The Hill cipher is often used in educational contexts to teach the principles of linear algebra and cryptography. It provides a clear example of how mathematical concepts can be applied to encryption.

Historical Context:
While not commonly used in modern cryptographic applications, the Hill cipher is an important part of the history of cryptography and can be studied to understand the evolution of encryption techniques.

Simple Encryption Needs:
For applications that do not require high levels of security but need a basic encryption method, the Hill cipher can be used. Examples include simple file encryption or obfuscation tasks.

Cryptanalysis Practice:
The Hill cipher can be used to practice cryptanalysis techniques, providing a stepping stone to understanding more complex encryption methods.

3. Read the paper given to you and summarize how is it better than Hill Cipher

The paper proposes a hybrid affine cipher and eigenvector-based encryption method which improves upon the traditional Hill cipher by offering:

Enhanced Security: Multiple encryption matrices can be chosen due to the infinite choices of eigenvectors, making it harder for attackers to break the encryption.

Optimized Process: Efficient encryption and decryption processes using matrix constructed from the eigenvectors of matrix

Flexibility and Robustness: Flexibility in selecting different eigenvectors for the same eigenvalue provides robustness against attacks.

Computational Efficiency: Potentially more efficient than the Hill cipher, especially for matrices with special structures.

This method increases security, optimizes the encryption-decryption process, and provides flexibility, making it more suitable for modern applications than the Hill cipher.

Conclusion: *Have hence successfully understood and implemented Hill Cipher and executed in code and answered question.*