Academic Year: 2024-25	**Programme: BTECH-Cyber (CSE)**
Year: 2nd	**Semester: IV**
Student Name: Mishitha Verma	**Batch : K2**
Roll No: K073	**Date of experiment: 27/1/25**
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)***

Note: Code should have proper comments

Encryption:
Code:

```
#define the key matrix
k = [[3, 3], [2, 5]]

#input even plaintext
while True:
    p = input("Enter plaintext (even length only): ").strip().upper()
    if len(p) % 2 == 0 and len(p) > 0:
        break
    print("Invalid input! Plaintext must have even number of characters.")

#encryption process
c = ""
for i in range(0, len(p), 2):
    #convert character pair to numbers
    a = ord(p[i]) - ord('A')
    b = ord(p[i+1]) - ord('A')

    #matrix multiplication with key
    c1 = (k[0][0] * a + k[0][1] * b) % 26
    c2 = (k[1][0] * a + k[1][1] * b) % 26

    #convert back to letters
    c += chr(c1 + ord('A')) + chr(c2 + ord('A'))

#output result
print("Ciphertext:", c)
```

Output:

```
> python -u "c:\Users\
verma\OneDrive\Documents\clg stuff\semester 4\itc\exp4\encryption.py"
Enter plaintext (even length only): help
Ciphertext: HIAT
PS C:\Users\verma\OneDrive\Documents\clg stuff\semester 4\itc\exp4>
Chat  Add Logs  CyberCoder  Improve Code  Share Code Link  CRLF  {} Python  3.12.3  Go Live  AI Code
```

```
> python -u "c:\Users\
verma\OneDrive\Documents\clg stuff\semester 4\itc\exp4\encryption.py"
Enter plaintext (even length only): mpstme
Ciphertext: DVHBWS
PS C:\Users\verma\OneDrive\Documents\clg stuff\semester 4\itc\exp4>
```

```
> python -u "c:\Users\verma\OneDrive\Documents\clg stuff\semester 4\itc\exp4\encryption.py"
Enter plaintext (even length only): mishitha
Ciphertext: IMXTDHVO
PS C:\Users\verma\OneDrive\Documents\clg stuff\semester 4\itc\exp4>
```

Chat Add Logs CyberCoder Improve Code Share Code Link CRLF {} Python 3.12.3 Go Live AI C

Decryption:
Code:

```python
#original key matrix
k = [[3, 3], [2, 5]]

#calculate determinant
det = k[0][0] * k[1][1] - k[0][1] * k[1][0]

#find modular inverse of determinant
inv_det = None
for x in range(1, 26):
    if (det * x) % 26 == 1:
        inv_det = x
        break


#calculate adjugate matrix
adj = [
    [k[1][1], -k[0][1]],
    [-k[1][0], k[0][0]]
]

#convert adjugate to positive modulo 26
for i in range(2):
    for j in range(2):
        adj[i][j] = adj[i][j] % 26

#calculate inverse key matrix
inv_key = [[0]*2, [0]*2]
for i in range(2):
    for j in range(2):
        inv_key[i][j] = (adj[i][j] * inv_det) % 26

#decryption process
while True:
    c = input("Enter ciphertext (even length): ").upper().strip()
    if len(c) % 2 == 0 and len(c) > 0:
```

```
        break
    print("Invalid input! Must have even length")

p = ""
for i in range(0, len(c), 2):
    #convert to numbers
    c1 = ord(c[i]) - ord('A')
    c2 = ord(c[i+1]) - ord('A')

    #matrix multiplication with inverse key
    a = (inv_key[0][0] * c1 + inv_key[0][1] * c2) % 26
    b = (inv_key[1][0] * c1 + inv_key[1][1] * c2) % 26

    p += chr(a + ord('A')) + chr(b + ord('A'))

print("Decrypted text:", p)
```

Output:

```
> python -u "c:\Users\
verma\OneDrive\Documents\clg stuff\semester 4\itc\exp4\decryption.py"
Enter ciphertext (even length): hiat
Decrypted text: HELP
PS C:\Users\verma\OneDrive\Documents\clg stuff\semester 4\itc\exp4>
```

```
> python -u "c:\Users\
verma\OneDrive\Documents\clg stuff\semester 4\itc\exp4\decryption.py"
Enter ciphertext (even length): dvhbws
Decrypted text: MPSTME
PS C:\Users\verma\OneDrive\Documents\clg stuff\semester 4\itc\exp4>
```

```
> python -u "c:\Users\
verma\OneDrive\Documents\clg stuff\semester 4\itc\exp4\decryption.py"
Enter ciphertext (even length): IMXTDHVO
Decrypted text: MISHITHA
PS C:\Users\verma\OneDrive\Documents\clg stuff\semester 4\itc\exp4>
```

Questions:

1. List advantages and limitations of Hill Cipher.

Ans: Advantages of Hill Cipher

1. High Security – Uses matrix-based encryption, making it more secure against frequency analysis attacks.
2. Fast Computation – Uses simple matrix multiplication and inverse operations, making it efficient.

3. Polygraphic Cipher – Encrypts multiple letters at once (n-gram), making it more resistant to single-character frequency attacks.
4. Mathematical Rigor – Provides a strong theoretical foundation using linear algebra.

Limitations of Hill Cipher

1. Key Requirement – The encryption key (matrix) must be invertible modulo 26, limiting key choices.
2. Known-Plaintext Vulnerability – If an attacker obtains enough plaintext-ciphertext pairs, they can solve for the key matrix.
3. No Diffusion Property – A single error in the ciphertext affects only a limited portion of the decrypted message.
4. Limited Resistance to Modern Attacks – Susceptible to known-plaintext and chosen-plaintext attacks due to its algebraic structure.

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

Ans: Secure Communication – Used in military and confidential messaging before modern cryptographic techniques.
Educational Purposes – Helps students understand linear algebra applications in cryptography.
Digital Watermarking – Can be used for embedding secure marks in digital images and videos.
Basic File Encryption – Can be implemented for small-scale text and file encryption systems.

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

Ans: This research paper focuses on the application of linear algebra in information security and cryptography. It explores how matrix operations and linear techniques enhance the design and analysis of encryption algorithms. Specifically, the paper covers the role of linear algebra in modern cryptographic systems and proposes a new matrix-based encryption scheme to improve security and efficiency.

Key points include:

Role of Linear Algebra: Linear algebra is crucial in various fields such as economics, engineering, and computer science. In encryption, it helps construct robust encryption systems that protect information from unauthorized access.

Encryption Techniques: The paper reviews several existing encryption techniques, such as the Hill cipher, affine cipher, and a method based on quadratic forms, all of which utilize matrix operations to enhance data security.

Matrix-Based Scheme: The paper proposes a new encryption scheme that incorporates linear algebra techniques, including eigenvalue decomposition, to enhance both security and efficiency in the encryption-decryption process.

Illustrative Example: The paper uses an example where a message ("GOOD LUCK") is converted into a matrix and encrypted using eigenvectors-based encryption.

Conclusion: *[Write your own conclusion regarding the lab performed]*

In this lab, we implemented the Hill Cipher using a 2×2 key matrix, applying matrix multiplication and modular arithmetic for encryption and decryption. We observed its strengths, such as encrypting multiple letters at once, resistance to simple frequency analysis, and mathematical efficiency. However, limitations like the requirement for an invertible key matrix, susceptibility to known-plaintext attacks, and lack of diffusion properties were also noted. Comparing it with an improved cryptographic method from the given paper, we saw how non-linear transformations, stronger key management, and hybrid encryption techniques (e.g: AES, RSA) enhance security. This lab reinforced the role of matrix operations and modular arithmetic in cryptography and highlighted the need for advanced encryption techniques in modern cybersecurity. By exploring the application of matrix operations, eigenvalues, and affine transformations, we demonstrate how these mathematical tools strengthen encryption methods to secure digital data. The research highlights the intersection of mathematics and computer science in developing efficient and robust encryption algorithms, with future advancements likely to integrate more complex mathematical principles for improved digital security. As cyber threats grow more sophisticated, the application of linear algebra remains crucial in safeguarding sensitive information and ensuring trust in digital systems.