|  |  |
| --- | --- |
| **Academic Year: 2024-25** | **Programme: BTECH-Cyber (CSE)** |
| **Year: 2nd** | **Semester: IV** |
| **Student Name: Jal Bafana** | **Batch: K1** |
| **Roll No: K005** | **Date of experiment: 23.01.2025** |
| **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 =

**Encryption**

C = K \* P mod 26

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

= mod 26

= mod 26

= mod 26

= = HI

Now we take next two letters “LP”

= mod 26

= mod 26

= mod 26

= = AT

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

**Decryption**

P = K-1 \* C mod 26

To find K-1 we need to use the formula

K-1 =1 / |K| \* adj K mod 26

K =

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

Need to find adjoint K

K =

=

=

= \\ Convert negative to positive by adding 26. E.g. -3 +26= 23, -2+26 = 24

= 1/9 \* mod26

=3\* mod26

= mod26

K-1 =

P = mod26

= mod26

= mod26

= = HE

Next two letters

= mod26

= mod26

= mod26

= = LP

So for Ciphertext is **HIAT**,thePlaintext is **HELP**

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

#include <iostream>

#include <string>

using namespace std;

int main()

{

string message;

int keyMatrix[2][2];

cout << "Enter the plaintext message (4 characters): ";

cin >> message;

if (message.length() != 4)

{

cout << "Please enter a message of exactly 4 characters!" << endl;

return 0;

}

cout << "Enter the 2x2 key matrix (4 numbers, one per line): " << endl;

for (int i = 0; i < 2; i++)

{

for (int j = 0; j < 2; j++)

{

cout << "Enter value for keyMatrix[" << i << "][" << j << "]: ";

cin >> keyMatrix[i][j];

}

}

// Process each 2-character block of the message

string CipherText = "";

for (int block = 0; block < 2; block++)

{

// Initialize the message vector for the current block (2 characters)

int messageVector[2][1];

// Fill the message vector with the numeric values of the current block

for (int i = 0; i < 2; i++)

{

messageVector[i][0] = (message[block \* 2 + i] - 'A') % 26; // Convert message letter to number

}

// Initialize the cipher matrix (encrypted message for the current block)

int cipherMatrix[2][1] = {0};

// Perform matrix multiplication: cipherMatrix = keyMatrix \* messageVector

for (int i = 0; i < 2; i++)

{

for (int j = 0; j < 1; j++)

{

cipherMatrix[i][j] = 0;

for (int x = 0; x < 2; x++)

{

cipherMatrix[i][j] += keyMatrix[i][x] \* messageVector[x][j];

}

cipherMatrix[i][j] = cipherMatrix[i][j] % 26; // Take modulo 26 for encryption

}

}

// Generate the encrypted ciphertext for the current block

for (int i = 0; i < 2; i++)

{

CipherText += (cipherMatrix[i][0] + 'A'); // Convert back to character (A=0, B=1, ..., Z=25)

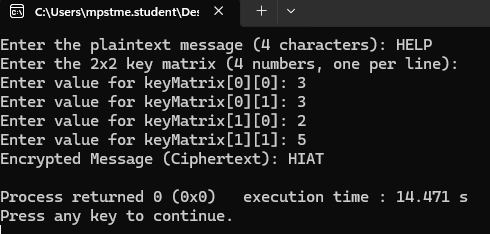
}

}

cout << "Encrypted Message (Ciphertext): " << CipherText << endl;

return 0;

}



#include <iostream>

#include <string>

using namespace std;

int modInverse(int a, int m)

{

a = a % m;

for (int x = 1; x < m; x++)

{

if ((a \* x) % m == 1)

{

return x;

}

}

return -1; // If no modular inverse exists

}

//inverse

bool inverseKeyMatrix(int keyMatrix[2][2], int inverseMatrix[2][2])

{

int determinant = keyMatrix[0][0] \* keyMatrix[1][1] - keyMatrix[0][1] \* keyMatrix[1][0];

determinant = determinant % 26;

if (determinant == 0 || modInverse(determinant, 26) == -1)

{

cout<<"Matrix is not invertible modulo 26!"<<endl;

return false;

}

int invDet = modInverse(determinant, 26);

inverseMatrix[0][0] = (keyMatrix[1][1] \* invDet) % 26;

inverseMatrix[0][1] = (-keyMatrix[0][1] \* invDet) % 26;

inverseMatrix[1][0] = (-keyMatrix[1][0] \* invDet) % 26;

inverseMatrix[1][1] = (keyMatrix[0][0] \* invDet) % 26;

// Ensuring the values are positive by taking mod 26

for (int i = 0; i < 2; i++)

{

for (int j = 0; j < 2; j++)

{

if (inverseMatrix[i][j] < 0) {

inverseMatrix[i][j] += 26;

}

}

}

return true;

}

int main()

{

// Declare variables

string cipherText;

int keyMatrix[2][2];

int inverseMatrix[2][2];

// Get the ciphertext from the user

cout << "Enter the ciphertext message (4 characters): ";

cin >> cipherText;

// Ensure the ciphertext length is 4

if (cipherText.length() != 4)

{

cout << "Please enter a ciphertext of exactly 4 characters!" << endl;

return 0;

}

// Get the key matrix directly from the user (4 values for 2x2 matrix)

cout << "Enter the 2x2 key matrix (4 numbers, one per line): " << endl;

for (int i = 0; i < 2; i++)

{

for (int j = 0; j < 2; j++)

{

cout << "Enter value for keyMatrix[" << i << "][" << j << "]: ";

cin >> keyMatrix[i][j];

}

}

// Find the inverse of the key matrix

if (!inverseKeyMatrix(keyMatrix, inverseMatrix))

{

return 0;

}

// Process each 2-character block of the ciphertext

string decryptedMessage = "";

for (int block = 0; block < 2; block++)

{

// Initialize the cipher vector for the current block (2 characters)

int cipherVector[2][1];

// Fill the cipher vector with the numeric values of the current block

for (int i = 0; i < 2; i++)

{

cipherVector[i][0] = (cipherText[block \* 2 + i] - 'A') % 26; // Convert ciphertext letter to number

}

// Initialize the message matrix (decrypted message for the current block)

int messageMatrix[2][1] = {0};

// multiplication

for (int i = 0; i < 2; i++)

{

for (int j = 0; j < 1; j++)

{

messageMatrix[i][j] = 0;

for (int x = 0; x < 2; x++)

{

messageMatrix[i][j] += inverseMatrix[i][x] \* cipherVector[x][j];

}

messageMatrix[i][j] = messageMatrix[i][j] % 26; // Take modulo 26 for decryption

}

}

// decrypted message generator

for (int i = 0; i < 2; i++)

{

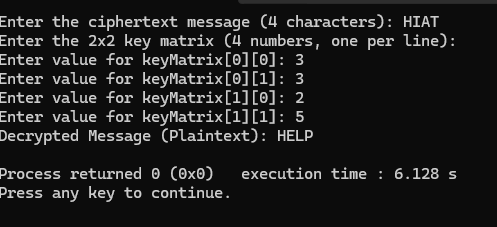
decryptedMessage += (messageMatrix[i][0] + 'A');

}

}

cout << "Decrypted Message (Plaintext): " << decryptedMessage << endl;

return 0;

}

*Note: Code should have proper comments*

**Questions:**

1. List advantages and limitations of Hill Cipher.

**Advantages of Hill Cipher:**

* Efficient Block Encryption: Hill Cipher encrypts multiple characters at once, making it faster than simpler ciphers like Caesar Cipher, especially for longer texts.
* Mathematical Foundation: It uses matrix operations and modular arithmetic, offering a strong mathematical structure and a good example of linear algebra in cryptography.

**Limitations of Hill Cipher:**

* Key Management Issues: If the key matrix is compromised, the entire system is vulnerable. Secure key distribution is crucial and can be complex.
* Vulnerable to Known-Plaintext Attacks: If both the plaintext and ciphertext are known, the key matrix can be deduced easily, making it insecure for modern use.

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

**Applications of Hill Cipher:**

* Educational Purposes: Hill Cipher is widely used in cryptography education to demonstrate matrix operations, modular arithmetic, and encryption techniques.
* Classical Cryptography: Historically used in military communications and other secure messaging systems, though now outdated for practical use.

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

This paper presents two methods: the hybrid affine cipher, and the eigenvector-based encryption, both of which, it is hoped, can overcome the limitations of the Hill cipher, and the benefits provided by them are as follows.

Improved Security: With infinite eigenvectors to select from, numerous encryption matrices can be selected, making it nearly impossible for an adversary to break the encryption.

Improved Lengthy Process: The encryption and decryption processes are done efficiently using the matrix constructed from the eigenvectors of matrix

Mobile & Robustness Flexibility of choosing different eigenvector for same eigenvalue provides robustness against attack.

Computational Efficience: Allen more

**Conclusion:** *In this lab, we implemented and coded the Hill Cipher to understand its encryption and decryption process using matrix operations.*