**Name : Akash Babu Misal**

**PRN: 21520005**

**Class:** Final Year (Computer Science and Engineering)

**Course Name: Cryptography and Network Security**  **Lab**

**Assignment No – 2**

**Aim - Implementation of the Columnar Cipher**

Theory –

**Columnar Cipher Overview:**

The Columnar Cipher is a transposition cipher that rearranges the letters of a plaintext message to create ciphertext. It operates by arranging the plaintext in a grid and then reading it out column by column, typically following a specific key.

**Part-A: Encryption**

In this part of the experiment, we will implement the Columnar Cipher encryption process. The encryption process involves the following steps:

**Key Setup:** The user provides a keyword, which is used to determine the order in which columns are read.

**Input:** The user provides a plaintext message to be encrypted.

**Text Preparation:** The plaintext message is processed to remove spaces and special characters, if needed. It is then converted to uppercase for consistency.

**Grid Formation:** The plaintext is arranged in a grid, with the number of columns determined by the length of the keyword.

**Encryption:** For encryption, we read the grid column by column according to the alphabetical order of the characters in the keyword. The characters in each column are then concatenated to form the ciphertext.

**Output**: The encrypted message is produced, which is the ciphertext created by reading the grid according to the keyword.

**Part-B: Decryption**

In this part of the experiment, we will implement the Columnar Cipher decryption process. The decryption process is the reverse of encryption and involves the following steps:

**Input**: The user provides the ciphertext and the keyword used for encryption.

**Grid Formation:** The number of columns in the grid is determined by the length of the keyword.

**Decryption:** For decryption, we read the grid column by column according to the alphabetical order of the characters in the keyword (the same order used for encryption). The characters in each column are then concatenated to form the plaintext.

**Output:** The decrypted message is produced, which should match the original plaintext message.

**Code -**

#include <bits/stdc++.h>

using namespace std;

string format(string &str)

{

    stringstream res;

    for (auto ch : str)

    {

        if (ch != ' ')

        {

            res << (char)tolower(ch);

        }

    }

    return res.str();

}

string encrypt(string &plain, string &key)

{

    map<char, vector<char>> mp;

    int counter = 0;

    for (int i = 0; i < plain.size(); i++)

    {

        mp[key[counter++]].push\_back(plain[i]);

        if (counter == key.size())

            counter = 0;

    }

    stringstream cipher;

    for (auto it : mp)

    {

        for (int i = 0; i < it.second.size(); i++)

        {

            cipher << it.second[i];

        }

    }

    return cipher.str();

}

string decrypt(string &cipher, string &key)

{

    map<int, int> map1;

    int common = cipher.size() / key.size();

    int extra = cipher.size() % key.size();

    for (int i = 0; i < key.size(); i++)

    {

        if (i < extra)

            map1[i] = common + 1;

        else

            map1[i] = common;

    }

    map<int, vector<char>> map2;

    int start = 0;

    string sortedKey(key);

    sort(sortedKey.begin(), sortedKey.end());

    for (int i = 0; i < sortedKey.size(); i++)

    {

        for (int j = 0; j < key.size(); j++)

        {

            if (sortedKey[i] == key[j])

            {

                for (int k = 0; k < map1[j]; k++)

                {

                    map2[key[j]].push\_back(cipher[start++]);

                }

            }

        }

    }

    string plain;

    vector<int> counters(key.size(), 0);

    int i = 0;

    while (plain.size() < cipher.size())

    {

        for (int i = 0; i < key.size(); i++)

        {

            if (counters[i] < map1[i])

                plain += map2[key[i]][counters[i]++];

        }

    }

    return plain;

}

int main()

{

    int choice;

    cout << "1. Encrypt\n2. Decrypt\nEnter your choice: ";

    cin >> choice;

    cin.get();

    if (choice == 1)

    {

        string plain, key;

        cout << "\nEnter plain text: ";

        getline(cin, plain);

        plain = format(plain);

        cout << "\nEnter key:  ";

        getline(cin, key);

        format(key);

        string cipher = encrypt(plain, key);

        cout << "\nEncrypted text is : " << cipher << endl;

    }

    else if (choice == 2)

    {

        string cipher, key;

        cout << "\nEnter cipher text: ";

        getline(cin, cipher);

        cipher = format(cipher);

        cout << "\nEnter key: ";

        getline(cin, key);

        format(key);

        string plain = decrypt(cipher, key);

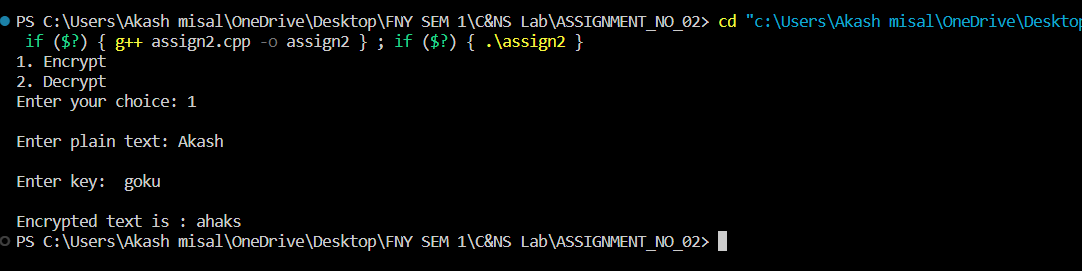
        cout << "\nDecrypted text is : " << plain << endl;

    }

    return 0;

}

Output:



**Conclusion**:

The Columnar Cipher is a relatively simple transposition cipher that rearranges the characters of a plaintext message based on a keyword. This experiment allows us to understand both the encryption and decryption processes of the Columnar Cipher and emphasizes the importance of the keyword in determining the arrangement of columns for encryption and decryption. Proper implementation and key management are essential for successful encryption and decryption using this cipher.