**Experiment-5**

**Aim:** Study and Implement a program for Vigenère Cipher to encrypt and decrypt the message.

**Introduction:**

Vigenère Cipher is a method of encrypting alphabetic text. It uses a simple form of polyalphabetic substitution. A polyalphabetic cipher is any cipher based on substitution, using multiple substitution alphabets. The encryption of the original text is done using the Vigenère square or Vigenère table.

* The table consists of the alphabets written out 26 times in different rows, each alphabet shifted cyclically to the left compared to the previous alphabet, corresponding to the 26 possible Caesar Ciphers.
* At different points in the encryption process, the cipher uses a different alphabet from one of the rows.
* The alphabet used at each point depends on a repeating keyword. For generating key, the given keyword is repeated in a circular manner until it matches the length of the plain text.

**Encryption:**

The first letter of the plaintext is combined with the first letter of the key. Then, the intersection of column of plain text’s character and row of key’s character is seen in the Vigenère table. That is the ciphertext’s character. Similarly, the second letter of the plaintext is combined with the second letter of the key. This process continues continuously until the plaintext is finished.

**Decryption:**

Decryption is done by the row of keys in the Vigenère table. First, select the row of the key letter, find the ciphertext letter's position in that row, and then select the column label of the corresponding ciphertext as the plaintext.

**Another method** of encryption and decryption is when the Vigenère table is not given. The encryption and decryption are done by Vigenère algebraically formula in this method (convert the letters (A-Z) into the numbers (0-25)).

**Formula of encryption is,**

Ei = (Pi + Ki) mod 26

**Formula of decryption is,**

Di = (Ei - Ki) mod 26

If any case (Di) value becomes negative (-ve), in this case, we will add 26 in the negative value.

**Where:**

E denotes the encryption.

D denotes the decryption.

P denotes the plaintext.

K denotes the key.

**Program (Source Code):**

//In this program, spaces are truncated and characters are converted to lowercase

#include <bits/stdc++.h>

using namespace std;

string encrypt(string plainText, string key){

    string encryptedText;

    string keyword;

    int len = plainText.length();

    //make keyword from key (make it the same length as the plainText)

    int repeat = ceil(double(len)/double(key.length()));

    for (int i=0;i<repeat;i++){

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

            if (keyword.length() < len){

                keyword += key[j];

            }

        }

    }

    //making encryptedText (by adding key and plainText)

    for (int i=0;i<len;i++){

        int ch = (((plainText[i] - 'a') + (keyword[i] - 'a'))%26) + 'a';

        encryptedText += char(ch);

    }

    return encryptedText;

}

string decrypt(string encryptedText, string key){

    string decryptedText;

    string keyword;

    int len = encryptedText.length();

    //make keyword from key (make it the same length as the encryptedText)

    int repeat = ceil(double(len)/double(key.length()));

    for (int i=0;i<repeat;i++){

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

            if (keyword.length() < len){

                keyword += key[j];

            }

        }

    }

    //making decryptedText (by subtracting key from encryptedText)

    for (int i=0;i<len;i++){

        int ch = (((encryptedText[i] - 'a') - (keyword[i] - 'a') + 26)%26) + 'a';

        decryptedText += char(ch);

    }

    return decryptedText;

}

int main(){

    string plainText0 = "mynameisravi";

    string key0 = "pdeu";

    string plainText;

    string key;

    //truncate spaces from key and plainText, and convert to lowercase

    for (int i=0;i<plainText0.length();i++){

        if (plainText0[i] != ' '){

            plainText += tolower(plainText0[i]);

        }

    }

    for (int i=0;i<key0.length();i++){

        if (key0[i] != ' '){

            key += tolower(key0[i]);

        }

    }

    cout<<"Plain Text: "<<plainText;

    cout<<"\nKey: "<<key;

    string encryptedText = encrypt(plainText, key);

    cout<<"\nEncrypted Text: "<<encryptedText;

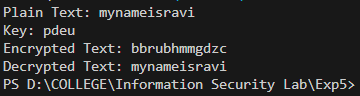
    string decryptedText = decrypt(encryptedText, key);

    cout<<"\nDecrypted Text: "<<decryptedText;

    return 0;

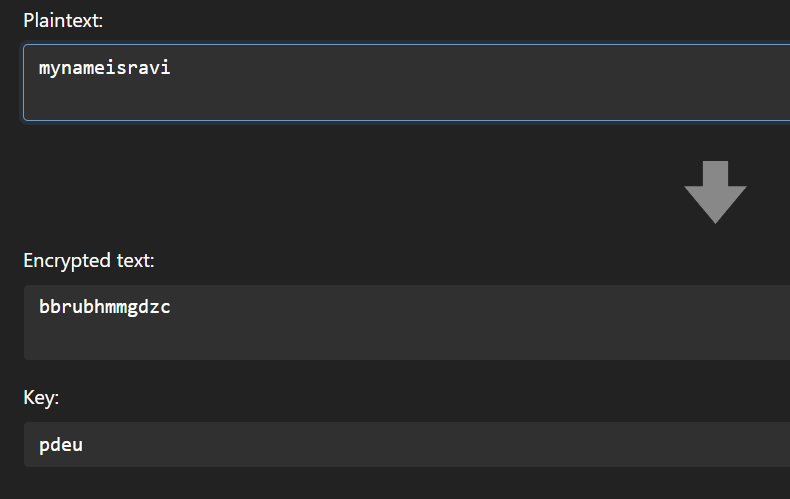
}

**Output (Program):**

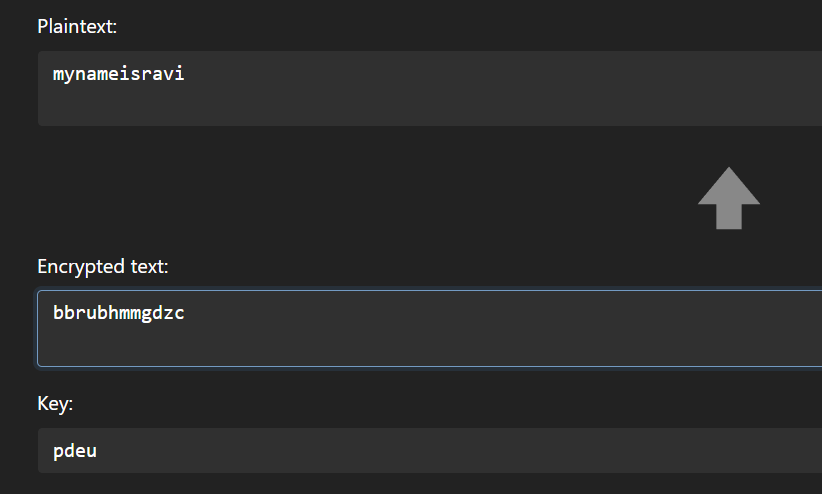
****

**Output (Cryptool):**

**Encryption:**

****

**Decryption:**



**Cryptanalysis:**

There are two main steps involved in breaking a Vigenère cipher. Those two are:

1. **Determine the key length:**

There are 2 ways of finding the key’s length:

**Index of Coincidence:** We start by dividing the message into different groups, each with the same number of characters. Then, we calculate the Index of Coincidence for each group. If the length of your groups matches the length of the keyword, the Index of Coincidence will be higher because the letters in those groups are likely shifted in the same way. In essence, the Index of Coincidence helps us find the right grouping of letters in the encrypted message, which can reveal the length of the keyword and make it easier to decrypt the message.

* **Kasiski Test:** While applying the technique of finding the index of coincidence, you also would have found certain patterns. Using these patterns for finding the length of the key for the message is called the Kasiski test. When we analyse the ciphertext, we get some strong patterns in repetition. This test is based on these observations. We see that two identical segments of the plaintext will be encrypted to the same ciphertext whenever their occurrence in the plaintext is δ positions apart, where δ ≡ 0 (mod m). Conversely, when we observe two identical string segments of the ciphertext, each of length k. There is a good chance that the length of the key may also be k. Each index of the key will act as the key for the corresponding character in the plaintext string.
* **Friedman Test:** It does this by looking at the repeating patterns in the encrypted message. You divide the message into sections of a certain length and then calculate the Index of Coincidence (IC) for each section. If the IC for a particular section matches what you would expect for the language of the message (like English), it suggests that the keyword might be that length. Essentially, it helps you detect when the same part of the keyword is applied to different parts of the message.

1. **Finding the key:**

**Key Elimination:** If the key length is known (or guessed), subtracting the cipher text from itself, offset by the key length, will produce the plain text subtracted from itself, also offset by the key length. If any "probable word" in the plain text is known or can be guessed, its self-subtraction can be recognized, which allows recovery of the key by subtracting the known plaintext from the cipher text.

**Applications:**

The Vigenère cipher is a classical method of encrypting alphabetic text by using a simple form of polyalphabetic substitution. It was originally developed in the 16th century and has been used for various applications throughout history, though it is not considered a secure encryption method today due to its vulnerability to modern cryptanalysis techniques. Here are some historical and modern applications of the Vigenère cipher:

1. Military Communication (Historical): The Vigenère cipher was historically used for secure military communication. It provided a way to encode messages so that only those with the correct key could decipher them.

2. Diplomatic Correspondence (Historical): Diplomats and government officials also used the Vigenère cipher to encode sensitive diplomatic correspondence. It offered a level of security against unauthorized interception of messages.

3. Education: The Vigenère cipher was used as a teaching tool to introduce students to cryptography. Its relatively simple encryption and decryption process made it suitable for educational purposes.

4. Puzzles and Recreational Cryptography: Cryptographers and enthusiasts created puzzles and challenges using the Vigenère cipher as a form of recreational cryptography. This encouraged people to learn about cryptanalysis and decipher encoded messages.

5. Steganography: While not a primary use, the Vigenère cipher can be incorporated into steganography techniques, where the goal is to hide a message within another medium (e.g., an image or audio file) to avoid detection.

It is important to note that the Vigenère cipher is not considered secure for modern cryptographic purposes, as it can be easily broken with modern cryptanalysis techniques.

**References:**

1. GeeksforGeeks
2. www.codingninjas.com
3. en.wikipedia.org