

Experiment No: 5

Aim: To study and implementation of Vigenère Cipher.

Introduction:

The Vigenère Cipher is one type of a substitution cipher. It can be compared to Caesar Cipher. Though it is a interwoven Caesar Cipher. In Caesar Cipher, we used to add a key to every alphabet of a plaintext. But here we have a string as a key. So, we add key and plaintext. The output can also be given by using below matrix.

| | | Plaintext | | | | | | | | | | | | | | | | | | | | | | | | | |
|-----|---|-----------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Key | | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| | A | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| | B | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A |
| | C | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B |
| | D | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C |
| | E | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D |
| | F | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E |
| | G | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F |
| | H | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G |
| | I | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H |
| | J | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I |
| | K | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J |
| | L | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K |
| | M | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L |
| | N | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M |
| | O | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N |
| | P | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| | Q | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P |
| | R | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q |
| | S | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R |
| | T | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S |
| | U | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T |
| | V | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U |
| | W | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V |
| | X | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W |
| | Y | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X |
| | Z | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y |

For example, “Hello world” is a plaintext and “begin” is the key. Then “Tirtb xsxtq” will be the output.

Program (Source Code):

```
#include<iostream>
using namespace std;

int encry(string key, string text)
{
    string output = "";
    for(int i=0,j=0; i<text.length(); i++,j++)
    {
        if(int(text[i]) < 91 && text[i] != ' ')
```

```

        {

            if((int(text[i]) + int(key[j]) - 97) > 90)
            {
                output += (int(text[i]) + int(key[j]) - 97 - 26);
            }
            else
            {
                output += (int(text[i]) + int(key[j]) - 97);
            }
        }

        else if(int(text[i]) > 96)
        {
            if((int(text[i]) + int(key[j]) - 97) < 123)
            {
                output += (int(text[i]) + int(key[j]) - 97);
            }
            else
            {
                output += (int(text[i]) + int(key[j]) - 97 - 26);
            }
        }

        else if(text[i] == ' ')
        {
            output += " ";
            j--;
        }

        if(j == key.length()-1) {j=-1;}
    }

    cout<<output;

    return 0;
}

int decry(string key, string text)
{
    string output = "";
    for(int i=0,j=0; i<text.length(); i++,j++)
    {
        if(int(text[i]) < 91 && text[i] != ' ')
        {
            if((int(text[i]) - int(key[j]) + 97) < 65)
            {
                output += (int(text[i]) - int(key[j]) + 97 + 26);
            }
        }
    }
}

```

```

        else
        {
            output += (int(text[i]) - int(key[j]) + 97);
        }
    }

    else if(int(text[i]) > 96)
    {
        if((int(text[i]) - int(key[j]) + 97) > 96)
        {
            output += (int(text[i]) - int(key[j]) + 97);
        }
        else
        {
            output += (int(text[i]) - int(key[j]) + 97 + 26);
        }
    }

    else if(text[i] == ' ')
    {
        output += " ";
        j--;
    }

    if(j == key.length()-1) {j=-1;}
}

cout<<output;

return 0;
}

int main()
{
    string key,text;
    cout<<"Enter text: "<<endl;
    getline(cin>>ws, text);
    cout<<"Enter key in small letters, excluding space: "<<endl;
    getline(cin>>ws, key);

    int choice;
    cout<<"Press 1 to encipher the text"<<endl;
    cout<<"Press 2 to decipher the text"<<endl;
    cin>>choice;

    switch(choice)
    {
    case 1:
        cout<<"Encrypted output:"<<endl;
        encry(key, text);

```

```

        break;

    case 2:
        cout<<"Decrypted output:"<<endl;
        decry(key, text);
        break;

    default:
        cout<<"Try again next time"<<endl;
    }

    return 0;
}

```

Output (Program):

The screenshot shows the Dev-C++ IDE with the file `D:\BRUJ Study\SEM 5\Information Security\IS_CompiledFiles\Vigenere_Cipher.cpp` open. The program is running, and the output window displays the following text:

```

Enter text:
This cipher was tiny but dangerous
Enter key in small letters, excluding space:
thankgod
Press 1 to encipher the text
Press 2 to decipher the text
1
Encrypted output:
Wolif modkxy wnc zwqr iug ngbjxyohc
-----
Process exited after 6.949 seconds with return value 0
Press any key to continue . . .

```

Below the output window, the compiler status is shown:

```

- Output Size: 1.90497875213623 MB
- Compilation Time: 1.02s

```

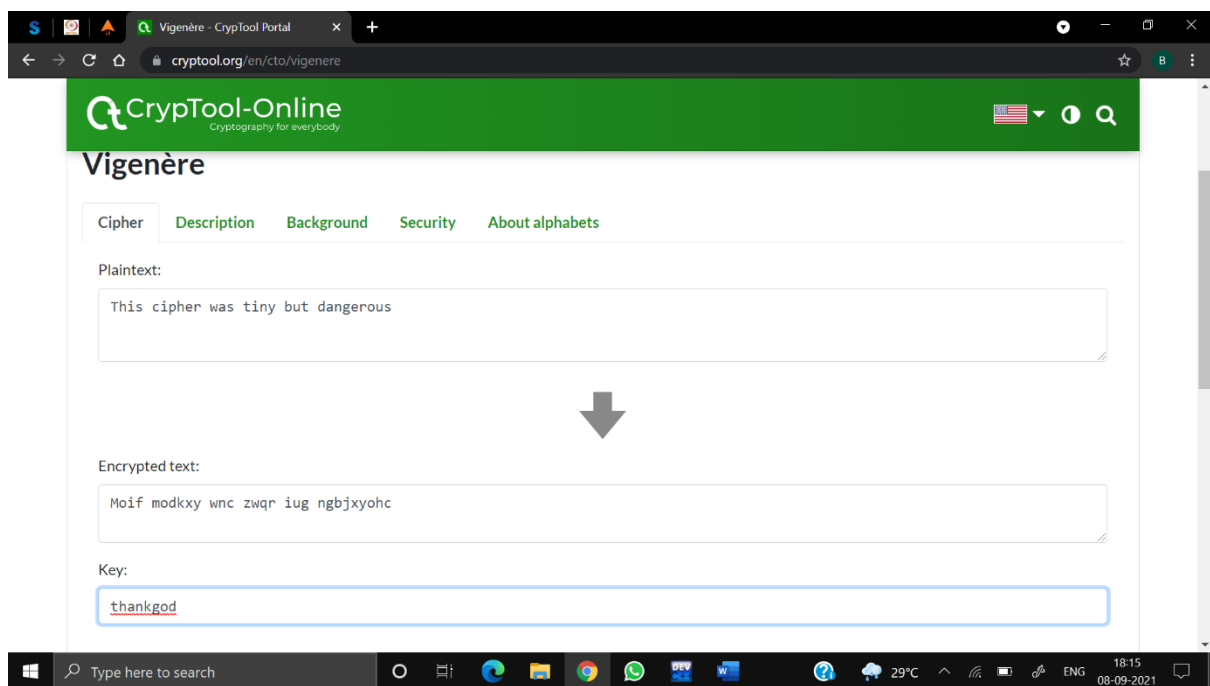
The status bar at the bottom indicates the current line is 110, column is 15, and the file length is 2117 characters. The system tray shows the date and time as 18:16 on 08-09-2021.

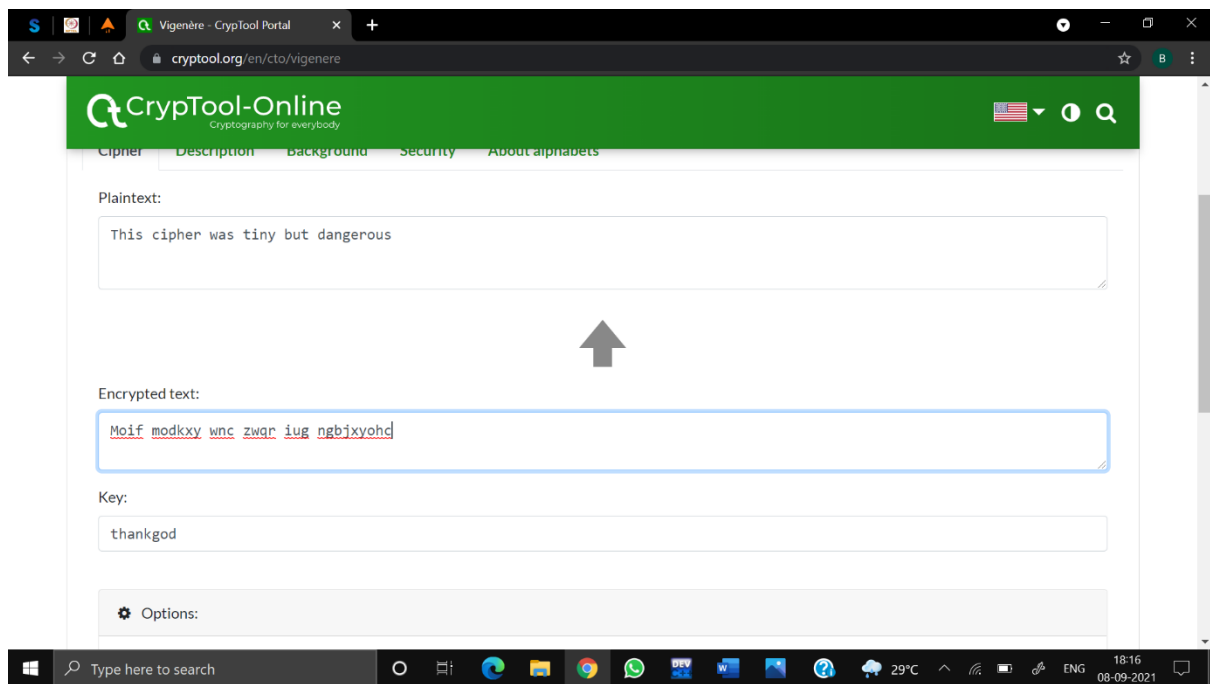
The screenshot shows a Windows IDE (Dev-C++) with a project named "Vigenere_Cipher.cpp". The program is running, and the output window displays the following text:

```
Enter text:
Moif modkxy wnc zwqr iug ngbjxyohc
Enter key in small letters, excluding space:
thankgod
Press 1 to encipher the text
Press 2 to decipher the text
2
Decrypted output:
This cipher was tiny but dangerous
-----
Process exited after 26.43 seconds with return value 0
Press any key to continue . . .
```

The status bar at the bottom indicates the file is at Line 110, Column 15, and the program has finished parsing in 0.015 seconds.

Output (Cryptool):





Cryptanalysis:

The idea behind the Vigenère cipher, like all other polyalphabetic ciphers, is to disguise the plaintext letter frequency to interfere with a straightforward application of frequency analysis. For instance, if P is the most frequent letter in a ciphertext whose plaintext is in English, one might suspect that P corresponds to E since E is the most frequently used letter in English. However, by using the Vigenère cipher, E can be enciphered as different ciphertext letters at different points in the message, which defeats simple frequency analysis.

The primary weakness of the Vigenère cipher is the repeating nature of its key. If a cryptanalyst correctly guesses the key's length, the cipher text can be treated as interwoven Caesar ciphers, which can easily be broken individually.

Applications:

Vigenère cipher, type of substitution cipher used for data encryption in which the original plaintext structure is somewhat concealed in the ciphertext by using several different monoalphabetic substitution ciphers rather than just one; the code key specifies which particular substitution is to be employed for encrypting each plaintext symbol.

1. Such resulting ciphers, known generically as polyalphabetic, have a long history of usage.
2. The systems differ mainly in the way in which the key is used to choose among the collection of monoalphabetic substitution rules.
3. Application of Message Security Application Using Vigenère Cipher

References:

1. <https://www.cryptool.org/en/cto/vigenere>
2. https://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher

3. <https://www.javatpoint.com/vigenere-cipher>
4. <https://static.javatpoint.com/tutorial/pwa/images/vigenere-cipher.png>