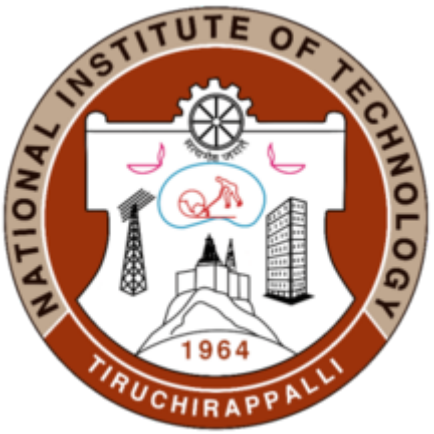
National Institute of Technology, Tiruchirappalli



Department of Computer Applications

Information Security Lab

Lab 7

Submitted by:

Akriti Upadhyay

205120007

MCA – 2nd Year

Exercise - 7

Divide the message in two parts and successively apply 3 different encryption and decryption algorithms on both the parts separately.

```
#include <bits/stdc++.h>
using namespace std;

vector<int> keystream;

// CAESAR CIPHER CODE
string caesar_encrypt(string s, int shift)
{
    string result;
    int arr[26] = {0};
    for (int i = 0; i < s.length(); i++)
    {
        if (s[i] == ' ' || s[i] == '.' || s[i] == ',' || s[i] == '!' ||
s[i] == ';' || s[i] == ':' || s[i] == '-' || s[i] == '_' || s[i] == 34
|| s[i] == 39)
            result.push_back(s[i]);
        else if (((int)s[i] + shift) > 122)
            result.push_back(s[i] + shift - 26);
        else
            result.push_back(s[i] + shift);
    }
    return result;
}

string caesar_decrypt(string s, int shift)
{
    string result;
    int arr[26] = {0};
    for (int i = 0; i < s.length(); i++)
    {
        if (s[i] == ' ' || s[i] == '.' || s[i] == ',' || s[i] == '!' ||
s[i] == ';' || s[i] == ':' || s[i] == '-' || s[i] == '_' || s[i] == 34
|| s[i] == 39)
```

```
            result.push_back(s[i]);
        else if (((int)s[i] - shift) < 97)
            result.push_back(s[i] - shift + 26);
        else
            result.push_back(s[i] - shift);
    }
    return result;
}

// DOUBLE TRANSPOSITION CODE
string double_transp_encrypt(string s, string str, vector<int> row,
vector<int> col)
{
    string result, res;
    int x = 0;
    char arr[row.size()][col.size()];
    for (int i = 0; i < row.size(); i++)
        for (int j = 0; j < col.size(); j++)
        {
            if (x >= str.length())
                arr[i][j] = '#';
            else
            {
                arr[i][j] = str[x];
                x++;
            }
        }
    x = 0;
    for (int i = 0; i < row.size(); i++)
        for (int j = 0; j < col.size(); j++)
            res.push_back(arr[row[i] - 1][col[j] - 1]);
    for (int i = 0; i < s.length(); i++)
    {
        if (s[i] > 96 && s[i] < 123)
        {
            result.push_back(res[x]);
            x++;
        }
        else if (s[i] != '.')
            result.push_back(s[i]);
    }
    while (x != res.length())
    {
```

```
        result.push_back(res[x]);
        x++;
    }
    return result;
}

string double_transp_decrypt(string s, string str, vector<int> row,
vector<int> col)
{
    string result;
    char arr[row.size()][col.size()];
    int x = 0;
    for (int i = 0; i < row.size(); i++)
        for (int j = 0; j < col.size(); j++)
        {
            arr[row[i] - 1][col[j] - 1] = str[x];
            x++;
        }
    x = 0;
    int i = 0, j = 0;
    while (x < s.length())
    {
        if (s[x] > 96 && s[x] < 123)
        {
            result.push_back(arr[i][j]);
            j++;
            if (j >= col.size())
            {
                i++;
                j = 0;
            }
        }
        else if (s[x] != '#')
            result.push_back(s[x]);
        x++;
    }
    return result;
}

// MONO - ALPHABETIC SUBSTITUTION CODE
string mono_alpha_encrypt(string s, string key)
{
    string result;
```

```
    char arr[26];
    for (int i = 0; i < key.length(); i++)
        arr[i] = key[i];
    for (int i = 0; i < s.length(); i++)
    {
        if (s[i] > 96 && s[i] < 123)
            result.push_back(arr[s[i] - 97]);
        else
            result.push_back(s[i]);
    }
    return result;
}

string mono_alpha_decrypt(string s, string key)
{
    string result;
    char arr[26];
    char ch = 'a';
    for (int i = 0; i < key.length(); i++)
    {
        arr[key[i] - 97] = ch;
        ch++;
    }
    for (int i = 0; i < s.length(); i++)
    {
        if (s[i] > 96 && s[i] < 123)
            result.push_back(arr[s[i] - 97]);
        else
            result.push_back(s[i]);
    }
    return result;
}

// POLY - ALPHABETIC SUBSTITUTION CODE
string poly_alpha_encrypt(string s, string key)
{
    string result;
    int arr[26][26];
    for (int i = 0; i < 26; i++)
        for (int j = 0; j < 26; j++)
        {
            if (i + j + 97 > 122)
                arr[i][j] = (i + j + 71);
```

```
            else
                arr[i][j] = (i + j + 97);
        }
    vector<pair<char, char>> v;
    int j = 0;
    for (int i = 0; i < s.length(); i++)
    {
        if (j >= key.length())
            j = 0;
        if (s[i] > 96 && s[i] < 123)
        {
            v.push_back(make_pair(s[i], key[j]));
            j++;
        }
        else
            v.push_back(make_pair(s[i], s[i]));
    }
    for (int i = 0; i < v.size(); i++)
    {
        if (v[i].first > 96 && v[i].first < 123)
            result.push_back(arr[v[i].first - 97][v[i].second - 97]);
        else
            result.push_back(v[i].first);
    }
    return result;
}

string poly_alpha_decrypt(string s, string key)
{
    string result;
    int arr[26][26];
    for (int i = 0; i < 26; i++)
        for (int j = 0; j < 26; j++)
        {
            if (i + j + 97 > 122)
                arr[i][j] = (i + j + 71);
            else
                arr[i][j] = (i + j + 97);
        }
    vector<pair<char, char>> v;
    int j = 0;
    for (int i = 0; i < s.length(); i++)
    {
```

```
        if (j >= key.length())
            j = 0;
        if (s[i] > 96 && s[i] < 123)
        {
            v.push_back(make_pair(s[i], key[j]));
            j++;
        }
        else
            v.push_back(make_pair(s[i], s[i]));
    }
    for (int i = 0; i < v.size(); i++)
    {
        for (int j = 0; j < 26; j++)
        {
            if (v[i].first == arr[v[i].second - 97][j])
            {
                result.push_back(j + 97);
                break;
            }
            if (v[i].first < 97 || v[i].first > 122)
            {
                result.push_back(v[i].first);
                break;
            }
        }
    }
    return result;
}

// ONE TIME PAD (OTP) CODE
string otp_encrypt(string s, string key)
{
    string res;
    for (int i = 0; i < s.length(); i++)
    {
        if (s[i] > 96 && s[i] < 123)
            res.push_back(((s[i] + key[i] - 194) % 26) + 97);
        else
            res.push_back(s[i]);
    }
    return res;
}
```

```
string otp_decrypt(string s, string key)
{
    string res;
    for (int i = 0; i < s.length(); i++)
    {
        if (s[i] > 96 && s[i] < 123)
        {
            if (s[i] - key[i] < 0)
                res.push_back(26 - abs(s[i] - key[i]) + 97);
            else
                res.push_back(s[i] - key[i] + 97);
        }
        else
            res.push_back(s[i]);
    }
    return res;
}

// RC4 CODE
string rc4_encrypt(string message, string key)
{
    int s[256], k[256], j = 0;
    for (int i = 0; i < 256; i++)
    {
        s[i] = i;
        if (j >= key.length())
            j = 0;
        k[i] = (int)key[j];
        j++;
    }
    j = 0;
    for (int i = 0; i < 256; i++)
    {
        j = (j + s[i] + k[i]) % 256;
        swap(s[i], s[j]);
    }
    j = 0;
    int t;
    for (int i = 1; i <= message.length(); i++)
    {
        j = (j + s[i]) % 256;
        swap(s[i], s[j]);
        t = (s[i] + s[j]) % 256;
```

```
        keystream.push_back(s[t]);
    }
    string result;
    for (int i = 0; i < message.length(); i++)
        result.push_back((char)(keystream[i] ^ (int)message[i]));
    return result;
}

string rc4_decrypt(string message)
{
    string result;
    for (int i = 0; i < message.length(); i++)
        result.push_back((char)(keystream[i] ^ (int)message[i]));
    return result;
}

// DRIVER CODE
int main()
{
    // ENCRYPTION OF THE INPUT TEXT MESSAGE

    string message1, message2, text;
    cout << "\nEnter the message to be Encrypted: " << endl;
    getline(cin, text);
    // divide message into two parts
    if (text.length() % 2 == 0)
    {
        message1 = text.substr(0, text.length() / 2);
        message2 = text.substr(text.length() / 2);
    }
    else
    {
        message1 = text.substr(0, text.length() / 2 + 1);
        message2 = text.substr(text.length() / 2 + 1);
    }

    // PART - 01 : ENCRYPTION

    cout <<
"\n_____
_____\n";
    cout << "ENCRYPTION:\n";
    // STEP 1 : CAESAR CIPHER
```

```
    int shift;
    string result_step_1;
    cout << "\nEnter the secret SHIFT KEY (between 1 to 26):" << endl;
    cin >> shift;
    result_step_1 = caesar_encrypt(message1, shift);
    cout << "RESULT AFTER STEP 1 - CAESAR CIPHER ENCRYPTION: " <<
result_step_1 << endl;

    // STEP 2 : DOUBLE TRANSPOSITION
    string result_step_2, str;
    int row_no, col_no;
    for (int i = 0; i < result_step_1.length(); i++)
    {
        if (result_step_1[i] == '#')
            str.push_back(result_step_1[i]); //"hello we are here."
        else if (result_step_1[i] > 96 && result_step_1[i] < 123)
            str.push_back(result_step_1[i]); //"hellowearehere"
    }
    if ((int)sqrt(str.length()) == sqrt(str.length()))
    {
        col_no = (int)sqrt(str.length());
        row_no = (int)sqrt(str.length());
    }
    else
    {
        if (str.length() > ((int)sqrt(str.length()) *
((int)sqrt(str.length()) + 1)))
        {
            col_no = (int)sqrt(str.length()) + 1;
            row_no = (int)sqrt(str.length()) + 1;
        }
        else
        {
            col_no = (int)sqrt(str.length()) + 1;
            row_no = (int)sqrt(str.length());
        }
    }
    cout << "\nThe number of rows and columns are " << row_no << " and
" << col_no << endl;
    vector<int> row(row_no), col(col_no);
    cout << "Enter the row permutation:" << endl;
    for (int i = 0; i < row_no; i++)
        cin >> row[i];
```

```
    cout << "Enter the column permutation:" << endl;
    for (int i = 0; i < col_no; i++)
        cin >> col[i];
    result_step_2 = double_transp_encrypt(result_step_1, str, row,
col);
    cout << "RESULT AFTER STEP 2 - DOUBLE TRANSPOSITION CIPHER
ENCRYPTION: " << result_step_2 << endl;

    // STEP 3 : MONO-ALPHABETIC SUBSTITUTION
    string key_mono, result_step_3;
    cout << "\nEnter the 26 alphabet key (in lowercase only)" << endl;
    cin >> key_mono;
    result_step_3 = mono_alpha_encrypt(result_step_2, key_mono);
    cout << "RESULT AFTER STEP 3 - MONO-ALPHABETIC SUBSTITUTION
ENCRYPTION: " << result_step_3 << endl;

    // PART - 02 : ENCRYPTION

    // STEP 4 : POLY - ALPHABETIC SUBSTITUTION
    string key_poly, result_step_4;
    cout << "\nEnter the key (in lowercase only):" << endl;
    cin >> key_poly;
    result_step_4 = poly_alpha_encrypt(message2, key_poly);
    cout << "RESULT AFTER STEP 4 - POLY-ALPHABETIC SUBSTITUTION
ENCRYPTION: " << result_step_4 << endl;

    // STEP 5 : ONE TIME PAD
    string key_otp, result_step_5;
    cout << "\nEnter the key which is equal to the length of the input
string:" << endl;
    cin >> key_otp;
    result_step_5 = otp_encrypt(result_step_4, key_otp);
    cout << "RESULT AFTER STEP 5 - ONE TIME PAD ENCRYPTION: " <<
result_step_5 << endl;

    // STEP 6 : RC4
    string key_rc4, result_step_6;
    cout << "\nEnter the key: " << endl;
    cin >> key_rc4;
    result_step_6 = rc4_encrypt(result_step_5, key_rc4);
    cout << "RESULT AFTER STEP 6 - RC4 ENCRYPTION: " << result_step_6
<< endl;
```

```
    // FINAL RESULT AFTER ENCRYPTION
    cout << "\nRESULT AFTER PART 01 ENCRYPTION: " << result_step_3 <<
endl;
    cout << "RESULT AFTER PART 02 ENCRYPTION: " << result_step_6 <<
endl;
    string result_final_encrypted;
    result_final_encrypted = result_step_3 + result_step_6;
    cout << endl
         << "THE FINAL RESULT AFTER ENCRYPTION IS: " <<
result_final_encrypted << endl;
    cout << endl;

    // DECRYPTION OF THE ENCRYPTED INPUT TEXT

    // PART - 01 : DECRYPTION

    cout <<
"\n_____________________________________________________________________
_____________\n";
    cout << "DECRYPTION:\n";
    // STEP 7 : MONO-ALPHABETIC SUBSTITUTION
    string result_step_7;
    result_step_7 = mono_alpha_decrypt(result_step_3, key_mono);
    cout << "RESULT AFTER STEP 7 - MONO-ALPHABETIC SUBSTITUTION
DECRYPTION: " << result_step_7 << endl;

    // STEP 8 : DOUBLE TRANSPOSITION
    string result_step_8;
    result_step_8 = double_transp_decrypt(result_step_7, result_step_7,
row, col);
    cout << "RESULT AFTER STEP 8 - DOUBLE TRANSPOSITION CIPHER
DECRYPTION: " << result_step_8 << endl;

    // STEP 9 : CAESAR CIPHER
    string result_step_9;
    result_step_9 = caesar_decrypt(result_step_8, shift);
    cout << "RESULT AFTER STEP 9 - CAESAR CIPHER DECRYPTION: " <<
result_step_9 << endl;

    // PART - 02 : DECRYPTION

    // STEP 10 : RC4 CIPHER
    string result_step_10;
```

```
    result_step_10 = rc4_decrypt(result_step_6);
    cout << "RESULT AFTER STEP 10 - RC4 CIPHER DECRYPTION: " <<
result_step_10 << endl;

    // STEP 11 : ONE TIME PAD
    string result_step_11;
    result_step_11 = otp_decrypt(result_step_10, key_otp);
    cout << "RESULT AFTER STEP 11 - ONE TIME PAD DECRYPTION: " <<
result_step_11 << endl;

    // STEP 12 : POLY-ALPHABETIC SUBSTITUTION
    string result_step_12;
    result_step_12 = poly_alpha_decrypt(result_step_11, key_poly);
    cout << "RESULT AFTER STEP 12 - PLOY-ALPHABETIC SUBSTITUTION
DECRYPTION: " << result_step_12 << endl;

    // FINAL RESULT AFTER DECRYPTION
    cout << "\nRESULT AFTER PART 01 DECRYPTION: " << result_step_9 <<
endl;
    cout << "RESULT AFTER PART 02 DECRYPTION: " << result_step_12 <<
endl;
    string result_final_decrypted;
    result_final_decrypted = result_step_9 + result_step_12;
    cout << endl
         << "THE FINAL RESULT AFTER DECRYPTION IS : " <<
result_final_decrypted << endl
         << endl;

    return 0;
}
```

```
Enter the message to be Encrypted:
secretmessage


_____________________________________________________________________________
ENCRYPTION:

Enter the secret SHIFT KEY (between 1 to 26):
5
RESULT AFTER STEP 1 - CAESAR CIPHER ENCRYPTION: xjhwjyr

The number of rows and columns are 3 and 3
Enter the row permutation:
1 2 3
Enter the column permutation:
3 1 2
RESULT AFTER STEP 2 - DOUBLE TRANSPOSITION CIPHER ENCRYPTION: hxjywj#r#

Enter the 26 alphabet key (in lowercase only)
hijklmnopqrstuvwxyzabcdefg
RESULT AFTER STEP 3 - MONO-ALPHABETIC SUBSTITUTION ENCRYPTION: oeqfdq#y#

Enter the key (in lowercase only):
zebra
RESULT AFTER STEP 4 - POLY-ALPHABETIC SUBSTITUTION ENCRYPTION: dwtrgd

Enter the key which is equal to the length of the input string:
informationse
RESULT AFTER STEP 5 - ONE TIME PAD ENCRYPTION: ljyfxp

Enter the key:
horse
RESULT AFTER STEP 6 - RC4 ENCRYPTION: ╣²¡iO≈

RESULT AFTER PART 01 ENCRYPTION: oeqfdq#y#
RESULT AFTER PART 02 ENCRYPTION: ╣²¡iO≈

THE FINAL RESULT AFTER ENCRYPTION IS: oeqfdq#y#╣²¡iO≈
```

```
_____________________________________________________________________________
DECRYPTION:
RESULT AFTER STEP 7 - MONO-ALPHABETIC SUBSTITUTION DECRYPTION: hxjywj#r#
RESULT AFTER STEP 8 - DOUBLE TRANSPOSITION CIPHER DECRYPTION: xjhwjyr
RESULT AFTER STEP 9 - CAESAR CIPHER DECRYPTION: secretm
RESULT AFTER STEP 10 - RC4 CIPHER DECRYPTION: ljyfxp
RESULT AFTER STEP 11 - ONE TIME PAD DECRYPTION: dwtrgd
RESULT AFTER STEP 12 - PLOY-ALPHABETIC SUBSTITUTION DECRYPTION: essage

RESULT AFTER PART 01 DECRYPTION: secretm
RESULT AFTER PART 02 DECRYPTION: essage

THE FINAL RESULT AFTER DECRYPTION IS : secretmessage
```

X-X-X-X