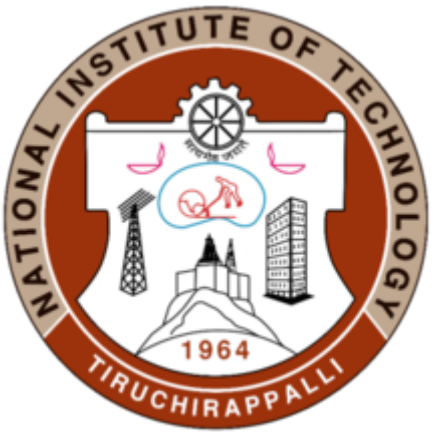
National Institute of Technology, Tiruchirappalli



Department of Computer Applications

Information Security Lab

Lab 1

Submitted to:

Dr. Ghanshyam Bopche

Submitted by:

Akriti Upadhyay

205120007

MCA – 2nd Year

Classical Encryption Algos

1. Caesar Cipher:

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

int main(){
    while(1){
        string msg;
        cout<<"\nEnter the message: \n";
        cin.ignore();
        getline(cin, msg);
        int key;
        cout<<"Enter the key: ";
        cin>>key;
        char choice;
        cout<<"What you want to do with the message?\n1. Encrypt
message\n2. Decrypt message\n";
        cin>>choice;
        // ENCRYPTION
        if(choice == '1'){
            for(int i=0; i<msg.size(); i++){
                //encrypting lower case letters
                if(msg[i]>='a' && msg[i]<='z'){
                    msg[i] = (((msg[i]-'a') + key) % 26) + 'a';
                }
                //encrypting upper case letters
                else if(msg[i]>='A' && msg[i]<='Z'){
                    msg[i] = (((msg[i]-'A') + key) % 26) + 'A';
                }
            }
            cout<<"\nEncrypted Message: "<<msg<<"\n\n";
        }
        // DECRYPTION
        else if(choice == '2'){
            for(int i=0; i<msg.size(); i++){
                //decrypting lower case letters
                if(msg[i]>='a' && msg[i]<='z'){
                    msg[i] = ((((msg[i]-'a') - key) + 26) % 26) + 'a';
                }
```

```
                //encrypting upper case letters
                else if(msg[i]>='A' && msg[i]<='Z'){
                    msg[i] = ((((msg[i]-'A') - key) + 26) % 26) + 'A';
                }
            }
            cout<<"\nDecrypted Message: "<<msg<<"\n\n";
        }
        else{
            cout<<"Invalid Choice, Try again!";
        }
        cout<<"Do you want to continue? (y/n)\n";
        cin>>choice;
        if(choice == 'n')
            break;
    }
    return 0;
}
```

```
Enter the message:
 Hello World I am Akriti Upadhyay
Enter the key: 6
What you want to do with the message?
1. Encrypt message
2. Decrypt message
1

Encrypted Message: Nkrru Cuxrj O gs Gqxozo Avgjnege

Do you want to continue? (y/n)
y

Enter the message:
Nkrru Cuxrj O gs Gqxozo Avgjnege
Enter the key: 6
What you want to do with the message?
1. Encrypt message
2. Decrypt message
2

Decrypted Message: Hello World I am Akriti Upadhyay
```

---

2. Double Transposition Cipher:

```
#include<iostream>
```

```cpp
#include<vector>
#include<cstdlib>
#include<math.h>
using namespace std;

vector<vector<char>> encrypt_step_1(vector<vector<char>> matrix,
vector<int> row_permute, int row){
    vector<vector<char>> temp_matrix(row);
    for(int i=0; i<row; i++){
        temp_matrix[i] = matrix[row_permute[i]-1];
    }
    return temp_matrix;
}

vector<vector<char>> encrypt_step_2(vector<vector<char>> matrix,
vector<int> col_permute, int row, int col){
    vector<vector<char>> temp_matrix(row,vector<char>(col));
    for(int i=0; i<col; i++){
        for(int j=0; j<row; j++){
            temp_matrix[j][i] = matrix[j][col_permute[i]-1];
        }
    }
    return temp_matrix;
}

vector<vector<char>> decrypt_step_1(vector<vector<char>> matrix,
vector<int> col_permute, int row, int col){
    vector<vector<char>> temp_matrix(row,vector<char>(col));
    for(int i=0; i<col; i++){
        for(int j=0; j<row; j++){
            temp_matrix[j][col_permute[i]-1] = matrix[j][i];
        }
    }
    return temp_matrix;
}


vector<vector<char>> decrypt_step_2(vector<vector<char>> matrix,
vector<int> row_permute, int row){
    vector<vector<char>> temp_matrix(row);
    for(int i=0; i<row; i++){
        temp_matrix[row_permute[i]-1] = matrix[i];
    }
```

```
    return temp_matrix;
}

void print_matrix(vector<vector<char>> matrix){
    int r = matrix.size();
    int c = matrix[0].size();
    for(int i=0; i<r; i++){
        for(int j=0; j<c; j++){
            if(matrix[i][j] != '0')
                cout<<matrix[i][j]<<" ";
        }
        cout << "\n";
    }
    cout<<"\n";
}

string print_message(vector<vector<char>> matrix){
    string msg;
    int r = matrix.size();
    int c = matrix[0].size();
    for(int i=0; i<r; i++){
        for(int j=0; j<c; j++){
            if(matrix[i][j] != '0')
                msg.push_back(matrix[i][j]);
        }
    }
    return msg;
}


int main(){
    while(1){
        string msg;
        cout << "\nEnter The Message: \n";
        cin.ignore();
        getline(cin,msg);
        int len = msg.length();

        // Forming a grid of string
        int col = ceil(sqrt(len));
        int row = sqrt(len);
        if(row*col<len)
            row = col;
```

```
        vector<vector<char>> matrix(row,vector<char>(col,'0'));

        for(int i=0, k=0; i<row && k<len; i++){
            for(int j=0; j<col && k<len; j++){
                matrix[i][j] = msg[k++];
            }
        }
        cout << "\nOriginal message in form of a Grid: \n";
        print_matrix(matrix);

        // Generating random permutations for rearranging rows and
columns
        vector<int> row_permute(row);
        for(int i=0; i<row; i++)
            row_permute[i] = i+1;
        for(int i=1; i<row; i++)
            swap(row_permute[i], row_permute[rand()%i]);

        vector<int> col_permute(col);
        for(int i=0; i<col; i++)
            col_permute[i] = i+1;
        for(int i=1; i<col; i++)
            swap(col_permute[i], col_permute[rand()%i]);

        //Rearranging rows
        matrix = encrypt_step_1(matrix, row_permute, row);
        cout<<"Matrix after rearranging rows [Encryption Step 1]: \n";
        print_matrix(matrix);

        //Rearranging Cols
        matrix = encrypt_step_2(matrix,col_permute,row,col);
        cout<<"Matrix after rearranging columns [Encryption Step 2]:
\n";
        print_matrix(matrix);

        // Encrypted Msg
        cout<<"Encrypted Message: "<<print_message(matrix)<<"\n";

        //Getting back the original columns
        matrix = decrypt_step_1(matrix,col_permute,row,col);
        cout << "\nMatrix after getting back original column
arrangement [Decryption Step 1]: \n";
        print_matrix(matrix);
```

```
        //Getting back the original rows
        matrix = decrypt_step_2(matrix,row_permute,row);
        cout << "Matrix after getting back original row arrangement
[Decryption Step 2]: \n";
        print_matrix(matrix);

        // Decrypted Msg
        cout<<"Decrypted Message: "<<print_message(matrix)<<"\n";

        char choice;
        cout<<"\nDo you want to continue? (y/n)\n";
        cin>>choice;
        if(choice == 'n' || choice == 'N')
            break;
    }
    return 0;
}
```

```
Enter The Message:
WE ARE DISCUSSING NEWS

Original message in form of a Grid:
W E   A R
E   D I S
C U S S I
N G   N E
W S

Matrix after rearranging rows [Encryption Step 1]:
W S
W E   A R
E   D I S
C U S S I
N G   N E

Matrix after rearranging columns [Encryption Step 2]:
S W
  R A E W
D S I   E
S I S U C
  E N G N

Encrypted Message: SW RAEWDSI ESISUC ENGN

Matrix after getting back original column arrangement [Decryption Step 1]:
W S
W E   A R
E   D I S
C U S S I
N G   N E

Matrix after getting back original row arrangement [Decryption Step 2]:
W E   A R
E   D I S
C U S S I
N G   N E
W S

Decrypted Message: WE ARE DISCUSSING NEWS
```

---

3. Monoalphabetic Substitutional Cipher:

```cpp
#include<iostream>
#include<string>
#include<unordered_map>
using namespace std;

int main(){
    unordered_map<char,char> map = {
        {'A', 'Q'}, {'B', 'W'}, {'C', 'E'}, {'D', 'R'}, {'E', 'T'},
{'F', 'Y'}, {'G', 'U'}, {'H', 'I'}, {'I', 'O'}, {'J', 'P'}, {'K', 'A'},
```

```
{'L', 'S'}, {'M', 'D'}, {'N', 'F'}, {'O', 'G'}, {'P', 'H'}, {'Q', 'J'},
{'R', 'K'}, {'S', 'L'}, {'T', 'Z'}, {'U', 'X'}, {'V', 'C'}, {'W', 'V'},
{'X', 'B'}, {'Y', 'N'}, {'Z', 'M'},
        {'a', 'q'}, {'b', 'w'}, {'c', 'e'}, {'d', 'r'}, {'e', 't'},
{'f', 'y'}, {'g', 'u'}, {'h', 'i'}, {'i', 'o'}, {'j', 'p'}, {'k', 'a'},
{'l', 's'}, {'m', 'd'}, {'n', 'f'}, {'o', 'g'}, {'p', 'h'}, {'q', 'j'},
{'r', 'k'}, {'s', 'l'}, {'t', 'z'}, {'u', 'x'}, {'v', 'c'}, {'w', 'v'},
{'x', 'b'}, {'y', 'n'}, {'z', 'm'}, {' ', '$'}
    };
    while(1){
        string msg;
        cout<<"\nEnter the message: \n";
        cin.ignore();
        getline(cin, msg);
        char choice;
        cout<<"What you want to do with the message?\n1. Encrypt
message\n2. Decrypt message\n";
        cin>>choice;
        // ENCRYPTION
        if(choice == '1'){
            for(int i=0; i<msg.size(); i++){
                msg[i] = map[msg[i]];
            }
            cout<<"\nEncrypted Message: "<<msg<<"\n\n";
        }
        // DECRYPTION
        else if(choice == '2'){
            for(int i=0; i<msg.size(); i++){
                // find the key in map with value msg[i], and assign it
to msg[i]
                for(auto &it:map){
                    if(it.second == msg[i]){
                        msg[i] = it.first;
                        break;
                    }
                }
            }
            cout<<"\nDecrypted Message: "<<msg<<"\n\n";
        }
        else{
            cout<<"Invalid Choice, Try again!";
        }
        cout<<"Do you want to continue? (y/n)\n";
```

```
        cin>>choice;
        if(choice == 'n')
            break;
    }
    return 0;
}
```

```
Enter the message:
 Hello World I am Akriti Upadhyay
What you want to do with the message?
1. Encrypt message
2. Decrypt message
1

Encrypted Message: Itssg$Vgksr$O$qd$Qakozo$Xhqrinqn

Do you want to continue? (y/n)
y

Enter the message:
Itssg$Vgksr$O$qd$Qakozo$Xhqrinqn
What you want to do with the message?
1. Encrypt message
2. Decrypt message
2

Decrypted Message: Hello World I am Akriti Upadhyay

Do you want to continue? (y/n)
n
```

---

4. Polyalphabetic Substitutional Cipher:

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

string generateKey(string msg, string keyWord){
    int sizeKeyword = keyWord.size();
    int msgSize = msg.size();
    // if keyword is longer than the msg
    if(sizeKeyword > msgSize){
        return keyWord.substr(0, msgSize);
    }
    // msg >= keyword
    for(int i=sizeKeyword; i<msg.size(); i++){
```

```
        keyWord.push_back(keyWord[i%sizeKeyword]);
    }
    return keyWord;
}
string encryption(string msg, string key) {
    string output;
    for(int i=0; i<msg.size(); i++){
        // converting to range 0-25
        char x = (msg[i] + key[i]) % 26;
        // convert to range of ASCII Alphabet (65-90 | A-Z)
        if(msg[i] == ' ')
            x = ' ';
        else
            x = x + 'A';
        output.push_back(x);
    }
    return output;
}
string decryption(string encrypt, string key) {
    string output;
    int j=0;
    for (int i=0; i<encrypt.size(); i++) {
        // converting to range 0-25
        char x = (encrypt[i] - key[i] + 26) % 26;
        // convert to range of ASCII Alphabet (65-90 | A-Z)
        if(encrypt[i] == ' ')
            x = ' ';
        else
            x = x + 'A';
        output.push_back(x);
    }
    return output;
}

int main() {
    while(1){
        string keyWord;
        cout<<"\nEnter the key word (In Capital Letters): \n";
        cin>>keyWord;
        string msg;
        cout<<"\nEnter the Message (In Capital Letters): \n";
        cin.ignore();
        getline(cin,msg);
```

```
        string key = generateKey(msg, keyWord);
        string encrypt = encryption(msg, key);
        string decrypt = decryption(encrypt, key);
        cout<<"\nOriginal Message: "<<msg;
        cout<<"\nEncrypted Message: "<<encrypt;
        cout<<"\nDecrypted Message: "<<decrypt;
        char choice;
        cout<<"\nDo you want to continue? (y/n)\n";
        cin>>choice;
        if(choice == 'n' || choice == 'N')
            break;
    }
    return 0;
}
```

```
Enter the key word (In Capital Letters):
MEGABUCK

Enter the Message (In Capital Letters):
WE ARE DISCUSSING NEWS

Original Message: WE ARE DISCUSSING NEWS
Encrypted Message: II ASY NUWIUTMKXS TEXM
Decrypted Message: WE ARE DISCUSSING NEWS
Do you want to continue? (y/n)
Y

Enter the key word (In Capital Letters):
ZEBRA

Enter the Message (In Capital Letters):
HI I AM AKRITI UPADHYAY

Original Message: HI I AM AKRITI UPADHYAY
Encrypted Message: GM Z ZQ RKQMUZ TTBUHXEZ
Decrypted Message: HI I AM AKRITI UPADHYAY
Do you want to continue? (y/n)
N
```

X-X-X-X