

**AHSANULLAH UNIVERSITY OF SCIENCE AND TECHNOLOGY**

Department of Computer Science and Engineering

Program: Bachelor of Science in Computer Science and Engineering

Course Code: CSE 4174

Course Title: Cyber Security Lab

Academic Semester: Spring 2023

Assignment Topic: Substitution & Transposition Ciphers

Submitted on:13th November 2023

Submitted by

Name: SHUVASHIS SARKER

Student ID: ***20200104116***

Lab Section: C1

1. Devise a code for implementation of Monoalphabetic cipher.

Code:

#include<bits/stdc++.h>

using namespace std;

int main()

{

string plainText;

string key;

string key2="abcdefghijklmnopqrstuvwxyz";

cout << "Enter the plain text: ";

cin >> plainText;

cout << "Enter the key: ";

cin >> key;

int n = plainText.length(), temp=0;

string ency;

for(int i=0;i<n;++i){

temp = plainText[i]-'a';

ency+=key[temp];

}

cout << "Monoalphabetic\_Cipher\_encryption: " << ency << endl;

temp = 0;

string dency;

for(int i=0;i<n;++i){

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

{

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

{

temp = j;

break;

}

}

dency += key2[temp];

}

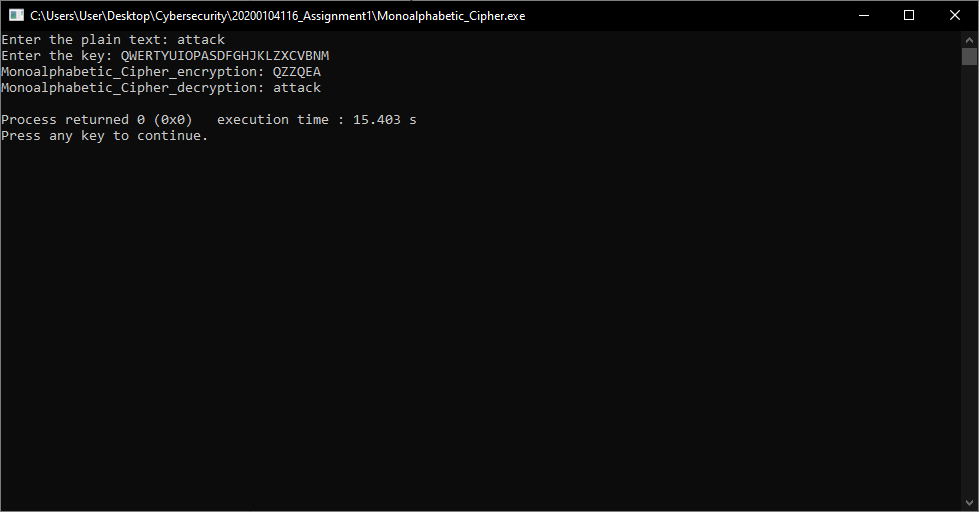
cout << "Monoalphabetic\_Cipher\_decryption: " << dency << endl;

return 0;

}

//key=QWERTYUIOPASDFGHJKLZXCVBNM

//plain\_text=attack



1. Devise a code for implementation of Polyalphabetic cipher.

Code:

#include<bits/stdc++.h>

using namespace std;

int main()

{

string pt, ct, key, rt;

int i,j;

cout << "enter the plain text: ";

cin >> pt;

cout << "enter the key: ";

cin >> key;

//length of plaintext equal to length of key

j=0;

for(i=key.length();i<pt.length();i++)

{

if(j==key.length())

{

j=0;

}

key+=key[j];

j++;

}

cout << "new key is: " << key;

cout << endl;

//converting plain text to cipher text (encryption)

for(i=0;i<pt.length();i++)

{

ct += (((pt[i]-'a')+(key[i]-'a'))%26)+'a';

}

cout << "cipher text is: " << ct << endl;

//converting cipher text to plain text (decryption)

for(i=0;i<ct.length();i++)

{

if(ct[i]<key[i])

{

rt += 26+((ct[i]-'a')-(key[i]-'a'))+'a';

}

else

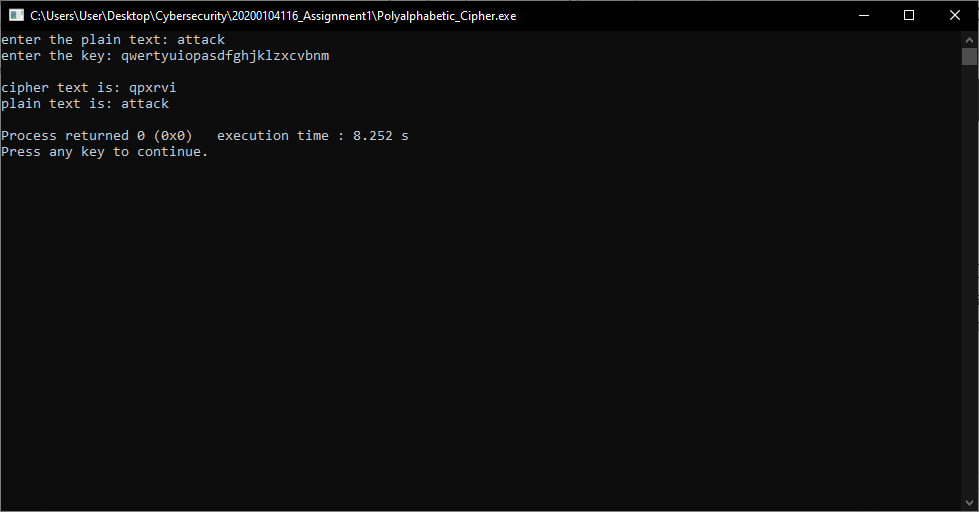
rt += (((ct[i]-'a')-(key[i]-'a'))%26)+'a';

}

cout << "plain text is: " << rt << endl;

}

//qwertyuiopasdfghjklzxcvbnm



1. Devise a code for implementation of Row Transposition cipher.

CODE:

#include<bits/stdc++.h>

using namespace std;

int main()

{

string plaintext, key;

cout << "Enter the plain text: ";

cin >> plaintext;

cout << "Enter the key: ";

cin >> key;

cout << "Plain text: " << plaintext << endl;

cout << "Key: " << key << endl;

int keyLength = key.length();

int textLength = plaintext.length();

// Calculate the number of rows needed

int numRows = textLength / keyLength;

if (textLength % keyLength != 0)

{

numRows++;

}

// Create a 2D array to hold the plaintext in rows and columns

char matrix[numRows][keyLength];

// Fill the matrix with the plaintext

int index = 0;

for (int i = 0; i < numRows; i++)

{

for (int j = 0; j < keyLength; j++)

{

if (index < textLength)

{

matrix[i][j] = plaintext[index++];

}

else

{

// If we run out of plaintext, fill the matrix with random characters

matrix[i][j] = 'x';

}

}

}

// Print the matrix (optional)

cout << "Matrix after encripted: " << endl;

for (int i = 0; i<keyLength; i++)

{

cout << key[i] << " ";

}

cout << endl;

for (int i = 0; i < numRows; i++)

{

for (int j = 0; j < keyLength; j++)

{

int col = key[j] - '1';

cout << matrix[i][col] << " ";

}

cout << endl;

}

// printing encryption message.

cout << "Encrypted text: ";

for (int i = 0; i < numRows; i++)

{

for (int j = 0; j < keyLength; j++)

{

int col = key[j] - '1';

cout << matrix[i][col];

}

}

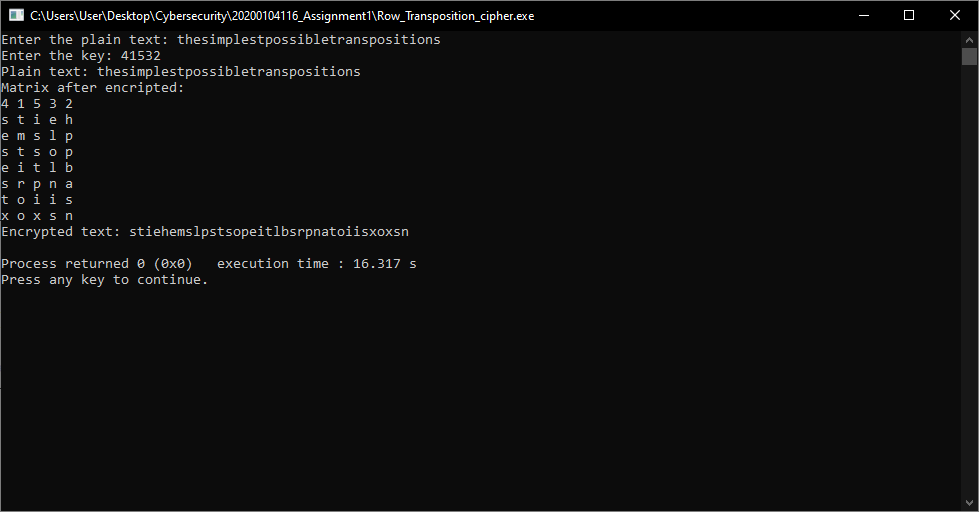
cout << endl;

return 0;

}

//thesimplestpossibletranspositions

//41532



1. Devise a code for implementation of Column Transposition cipher.

CODE:

#include<bits/stdc++.h>

using namespace std;

void encrypt(string plaintext, string key) {

int keyLength = key.length();

int textLength = plaintext.length();

// Calculate the number of columns needed

int numColumns = keyLength;

// Calculate the number of rows needed

int numRows = textLength / numColumns;

if (textLength % numColumns != 0) {

numRows++;

}

char matrix[100][100]; // Assuming a maximum size, adjust as needed

// Fill the matrix with the plaintext

int index = 0;

for (int j = 0; j < numRows; j++) {

for (int i = 0; i < numColumns; i++) {

if (index < textLength) {

matrix[j][i] = plaintext[index++];

} else {

// If we run out of plaintext, fill the matrix with random characters

matrix[j][i] = 'X';

}

}

}

// Print the matrix (optional)

cout << "Matrix after filling:" << endl;

for (int i = 0; i < numRows; i++) {

for (int j = 0; j < numColumns; j++) {

cout << matrix[i][j] << " ";

}

cout << endl;

}

// Encrypt the plaintext by reading the matrix column-wise based on the key numbers

cout << "Encrypted text: ";

for (int k = 1; k <= keyLength; k++) {

for (int i = 0; i < numColumns; i++) {

if (key[i] - '0' == k) {

for (int j = 0; j < numRows; j++) {

cout << matrix[j][i];

}

}

}

}

cout << endl;

}

int main() {

string plaintext;

string key;

cout << "Plain Text: ";

cin >> plaintext;

cout << "Key: ";

cin >> key;

//converting key to number form.

int len = key.length();

string key2;

char numberKey[len+1];

key2 = key;

sort(key2.begin(), key2.end());

for(int i=0; i<len; i++)

{

for(int j=0; j<len; j++)

{

if(key2[i]==key[j])

{

numberKey[j] = (i) + '1';

}

}

}

cout << "Original text: " << plaintext << endl;

cout << "Key: " << key << endl;

cout << "Key in number: " << numberKey << endl;

encrypt(plaintext, numberKey);

return 0;

}

//meetmeaftertheparty

//3124

