МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені Тараса Шевченка

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра програмних систем і технологій

**Практична робота №3**

з дисципліни

**«Захист програм та даних»**

**Тема :** **«Шифри гамування»**

Виконали:

Ємець Євгенія Ярославівна

Щука Богдан Юрійович

Група ІПЗ-33

Перевірила:

Супрун Ольга Миколаївна

Дата перевірки

2018

**Хід роботи**

**Завдання №1**

У практичній роботі необхідно зашифрувати своє прізвище і ім'я за допомогою шифру ADFGVX.

При оформленні звіту необхідно привести вихідне повідомлення (прізвище та ім'я), таблицю шіфрозамін, ключове слово, перестановки таблицю і зашифроване повідомлення.

Процес шифрування починається з малювання сітки розміру 5×5, кожна клітинка заповнюється 25 літерами латинського алфавіту (I та J шифруються однаково). Кожен рядок і стовпець сітки задається однією з 5 букв: «A», «D», «F», «G» і «X». Заповнення сітки здійснюється в довільному порядку, тому одержувач повинен знати розташування кожного елемента, щоб зробити дешифрування.

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

Розглянемо процес шифрування на прикладі невеликого повідомлення: «attack at dawn». На першому кроці кожен символ повідомлення замінюється на пару букв, що позначають рядок і стовпець відповідного символу в сітці. Наприклад, A буде замінено на FF, а B — на GA.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Повідомлення:** | attack at dawn | | | | | | | | | | | |
| **Відкритий текст:** | a | t | t | a | c | k | a | t | d | a | w | n |
| **Шифротекст на першому кроці:** | FF | XG | XG | FF | DX | GF | FF | XG | DD | FF | GX | ОГОЛОШЕННЯ |

На другому кроці застосовується перестановка, що значно ускладнює криптоаналіз. Перестановка здійснюється в залежності від ключового слова, яке повинно бути відомо одержувачу. Нехай, у нашому прикладі, таким словом буде «BATTLE». Процес перестановки полягає в наступному. Спочатку створюється нова сітка, у верхній частині якої записуються літери ключового слова. Потім, під цим словом порядково записується, отриманий на першому кроці зашифрований текст.

Далі, літери ключового слова переставляються в алфавітному порядку разом з відповідними стовпцями сітки.

Після чого літери кожного стовпця виписуються по черзі зверху вниз. Отримана послідовність букв утворює остаточний вигляд шифротексту.

Остаточний вигляд шифротекста: F F F F F F F F G F D D X G D A X D X G G X G X

У цьому вигляді шифротекст буде потім переданий за допомогою коду Морзе.

**Лістинг програми**

**using** System;  
**using** System.Collections.Generic;  
**using** System.Linq;  
**using** System.Text;  
  
**namespace** ADFGVX{  
 **public class** Adfgvx{  
 **private readonly string \_key**;  
 **public** Adfgvx(**string** key){  
 **\_key** = key;  
 }  
 **public string** DecryptMessage(**string** cipherText){  
 **var** splitCipher = cipherText.Split(**new**[] {**' '**}, StringSplitOptions.***RemoveEmptyEntries***);  
 **var** transformedCipher = TransformCipher(splitCipher);  
 **var** returnText = ReverseStepOne(transformedCipher);  
 **return** returnText.Replace(**"SPACE"**, **" "**);  
 }  
 **private string** ReverseStepOne(**string** transformedCipher){  
 **var** sb = **new** StringBuilder();  
 **for** (**var** i = 1; i < transformedCipher.**Length**; i += 2)  
 sb.Append(GetPlainText(transformedCipher[i - 1], transformedCipher[i]));  
 **return** sb.ToString();  
 }  
 **private static string** GetPlainText(**char** c, **char** c1){  
 **var** sb = **new** StringBuilder();  
 **var** sb2 = **new** StringBuilder();  
 sb2.Append(c);  
 sb2.Append(c1);  
 **switch** (sb2.ToString()){  
 **case "AA"**:  
 sb.Append(**"F"**);  
 **break**;  
 **case "AD"**:  
 sb.Append(**"U"**);  
 **break**;  
 **case "AF"**:  
 sb.Append(**"W"**);  
 **break**;  
 **case "AG"**:  
 sb.Append(**"5"**);  
 **break**;  
 **case "AV"**:  
 sb.Append(**"Q"**);  
 **break**;  
 **case "AX"**:  
 sb.Append(**"G"**);  
 **break**;  
 **case "DA"**:  
 sb.Append(**"M"**);  
 **break**;  
 **case "DD"**:  
 sb.Append(**"E"**);  
 **break**;  
 **case "DF"**:  
 sb.Append(**"V"**);  
 **break**;  
 **case "DG"**:  
 sb.Append(**"R"**);  
 **break**;  
 **case "DV"**:  
 sb.Append(**"H"**);  
 **break**;  
 **case "DX"**:  
 sb.Append(**"0"**);  
 **break**;  
 **case "FA"**:  
 sb.Append(**"1"**);  
 **break**;  
 **case "FD"**:  
 sb.Append(**"N"**);  
 **break**;  
 **case "FF"**:  
 sb.Append(**"D"**);  
 **break**;  
 **case "FG"**:  
 sb.Append(**"I"**);  
 **break**;  
 **case "FV"**:  
 sb.Append(**"6"**);  
 **break**;  
 **case "FX"**:  
 sb.Append(**"7"**);  
 **break**;  
 **case "GA"**:  
 sb.Append(**"2"**);  
 **break**;  
 **case "GD"**:  
 sb.Append(**"S"**);  
 **break**;  
 **case "GF"**:  
 sb.Append(**"J"**);  
 **break**;  
 **case "GG"**:  
 sb.Append(**"C"**);  
 **break**;  
 **case "GV"**:  
 sb.Append(**"X"**);  
 **break**;  
 **case "GX"**:  
 sb.Append(**"8"**);  
 **break**;  
 **case "VA"**:  
 sb.Append(**"T"**);  
 **break**;  
 **case "VD"**:  
 sb.Append(**"K"**);  
 **break**;  
 **case "VF"**:  
 sb.Append(**"#"**);  
 **break**;  
 **case "VG"**:  
 sb.Append(**"O"**);  
 **break**;  
 **case "VV"**:  
 sb.Append(**"B"**);  
 **break**;  
 **case "VX"**:  
 sb.Append(**"Y"**);  
 **break**;  
 **case "XA"**:  
 sb.Append(**"L"**);  
 **break**;  
 **case "XD"**:  
 sb.Append(**"Z"**);  
 **break**;  
 **case "XF"**:  
 sb.Append(**"4"**);  
 **break**;  
 **case "XG"**:  
 sb.Append(**"9"**);  
 **break**;  
 **case "XV"**:  
 sb.Append(**"P"**);  
 **break**;  
 **case "XX"**:  
 sb.Append(**"A"**);  
 **break**;  
 }  
 **return** sb.ToString();  
 }  
 **private string** TransformCipher(**string**[] splitCipher){  
 **var** alphabeticalString = **\_key**.ToList();  
 alphabeticalString.Sort();  
  
 **var** columnTransposition = **\_key**.ToDictionary(c => c, c => **new** List<**char**>());  
 **var** counter = 0;  
 **foreach** (**var** c **in** alphabeticalString){  
 columnTransposition[c].AddRange(splitCipher[counter].ToList());  
 counter++;  
 }  
 **var** sb = **new** StringBuilder();  
 **var** longestLength = GetLongestLength(splitCipher);  
 **for** (**var** i = 0; i < longestLength; i++){  
 **foreach** (**var** c **in \_key**){  
 **if** (columnTransposition[c].**Count** == 0) **continue**;  
 sb.Append(columnTransposition[c].FirstOrDefault());  
 columnTransposition[c].RemoveAt(0);  
 }  
 }  
 **return** sb.ToString();  
 }  
 **private static int** GetLongestLength(**string**[] splitCipher){  
 **var** largestLength = **int**.***MinValue***;  
 **foreach** (**var** each **in** splitCipher){  
 **if** (each.**Length** > largestLength)  
 largestLength = each.**Length**;  
 }  
 **return** largestLength;  
 }  
 **public string** EncryptMessage(**string** message){  
 message = message.Replace(**" "**, **"SPACE"**);  
 **var** stepOne = StepOne(message.ToLower());  
  
 **var** returnText = OutputText(stepOne);  
 **return** returnText;  
 }  
 **public string** CleanMessage(**string** message)  
 {  
 **var** sb = **new** StringBuilder();  
 **foreach** (**var** c **in** message){  
 **switch** (c){  
 **case '\r'**:  
 **case '\n'**:  
 **case '\t'**:  
 **case ' '**:  
 **continue**;  
 **default**:  
 sb.Append(c);  
 **break**;  
 }  
 }  
 **return** sb.ToString().ToLower();  
 }  
 **private string** OutputText(**string** message){  
 **var** counter = 0;  
 **var** columnTransposition = **\_key**.ToDictionary(c => c, c => **new** List<**char**>());  
 **foreach** (**var** each **in** message){  
 columnTransposition[**\_key**[counter]].Add(each);  
 counter++;  
 **if** (counter >= **\_key**.**Length**)  
 counter = 0;  
 }  
  
 **var** sb = **new** StringBuilder();  
 **var** alphabeticalKey = **\_key**.ToList();  
 alphabeticalKey.Sort();  
 **foreach** (**var** c **in** alphabeticalKey){  
 **foreach** (**var** each **in** columnTransposition[c])  
 sb.Append(each);  
 sb.Append(**' '**);  
 }  
 **return** sb.ToString();  
 }  
 **private static string** StepOne(**string** cleanedMessage)  
 {  
 **var** sb = **new** StringBuilder();  
 **foreach** (**var** c **in** cleanedMessage){  
 **switch** (c)  
 {  
 **case 'a'**:  
 sb.Append(**"XX"**);  
 **break**;  
 **case 'b'**:  
 sb.Append(**"VV"**);  
 **break**;  
 **case 'c'**:  
 sb.Append(**"GG"**);  
 **break**;  
 **case 'd'**:  
 sb.Append(**"FF"**);  
 **break**;  
 **case 'e'**:  
 sb.Append(**"DD"**);  
 **break**;  
 **case 'f'**:  
 sb.Append(**"AA"**);  
 **break**;  
 **case 'g'**:  
 sb.Append(**"AX"**);  
 **break**;  
 **case 'h'**:  
 sb.Append(**"DV"**);  
 **break**;  
 **case 'i'**:  
 sb.Append(**"FG"**);  
 **break**;  
 **case 'j'**:  
 sb.Append(**"GF"**);  
 **break**;  
 **case 'k'**:  
 sb.Append(**"VD"**);  
 **break**;  
 **case 'l'**:  
 sb.Append(**"XA"**);  
 **break**;  
 **case 'm'**:  
 sb.Append(**"DA"**);  
 **break**;  
 **case 'n'**:  
 sb.Append(**"FD"**);  
 **break**;  
 **case 'o'**:  
 sb.Append(**"VG"**);  
 **break**;  
 **case 'p'**:  
 sb.Append(**"XV"**);  
 **break**;  
 **case 'q'**:  
 sb.Append(**"AV"**);  
 **break**;  
 **case 'r'**:  
 sb.Append(**"DG"**);  
 **break**;  
 **case 's'**:  
 sb.Append(**"GD"**);  
 **break**;  
 **case 't'**:  
 sb.Append(**"VA"**);  
 **break**;  
 **case 'u'**:  
 sb.Append(**"AD"**);  
 **break**;  
 **case 'v'**:  
 sb.Append(**"DF"**);  
 **break**;  
 **case 'w'**:  
 sb.Append(**"AF"**);  
 **break**;  
 **case 'x'**:  
 sb.Append(**"GV"**);  
 **break**;  
 **case 'y'**:  
 sb.Append(**"VX"**);  
 **break**;  
 **case 'z'**:  
 sb.Append(**"XD"**);  
 **break**;  
 **case '0'**:  
 sb.Append(**"DX"**);  
 **break**;  
 **case '1'**:  
 sb.Append(**"FA"**);  
 **break**;  
 **case '2'**:  
 sb.Append(**"GA"**);  
 **break**;  
 **case '3'**:  
 sb.Append(**"VF"**);  
 **break**;  
 **case '4'**:  
 sb.Append(**"XF"**);  
 **break**;  
 **case '5'**:  
 sb.Append(**"AG"**);  
 **break**;  
 **case '6'**:  
 sb.Append(**"FV"**);  
 **break**;  
 **case '7'**:  
 sb.Append(**"FX"**);  
 **break**;  
 **case '8'**:  
 sb.Append(**"GX"**);  
 **break**;  
 **case '9'**:  
 sb.Append(**"XG"**);  
 **break**;  
 }  
 }  
 **return** sb.ToString();  
 }  
 }  
  
 **public class** mainProg  
 {  
 **public static void** Main(**string**[] Args)  
 {  
 Console.WriteLine(**"Enter key\t"**);  
 **var** code = **new** Adfgvx(Console.ReadLine());  
 Console.WriteLine(**"Enter message\t"**);  
 **var** message = Console.ReadLine();  
 Console.Write(**"Cleaned text\t{0}\n"**, code.CleanMessage(message));  
 **var** encrypt = code.EncryptMessage(message);  
 Console.Write(**"Coded text\t{0}\n"**, encrypt);  
 Console.Write(**"Decrypted text\t{0}"**, code.DecryptMessage(encrypt));  
 }  
 }  
}

**Приклади роботи**





