Задание по технологиям защиты информации

На тему: Древние шифры

Горшков Ян

**Задание:**

Построить графическую программу, позволяющую шифровать и дешифровать исходный текст методом четырех квадратов и шифром ADFGVX.

**Теоретические сведения:**

Шифр четырех квадратов был изобретен известным французским криптографом Феликсом Деластелем в 1902 году как модифицированный шифр Плейфера. Этот метод обеспечивает более высокий уровень безопасности защищенных данных и является методом ручного симметрического шифрования.

Данный метод шифрует пары букв и, тем самым, попадает в категорию шифров, известных как полиграфические подстановочные шифры. Это добавляет значительную прочность шифрованию по сравнению с шифрами простой замены, которые действуют на отдельные символы. Использование биграмм делает шифр четырех квадратов менее восприимчивым к частотному криптоанализу, так как он должен быть применен к 676 возможным парам букв (в английском алфавите), а не только к 26 символам для монографического замещения. Частотный анализ для биграмм возможен, но он гораздо сложнее, и для того, чтобы этот анализ был полезен, требуется, как правило, намного больше шифротекста.

4 квадрата случайным образом наполняют значения таким образом. Для шифрования слова необходимо соблюдать следующий алгоритм:

1. Начинаем с первого квадрата. Находим в нем первую букву.
2. Запоминаем её координаты.
3. В квадрате, находящимся по диагонали от первого, по этому местоположению выбираем новую букву.
4. Переходим к следующей букве, чередуя квадраты.

Пример случайного заполнения квадрата изображен на рисунке 1.

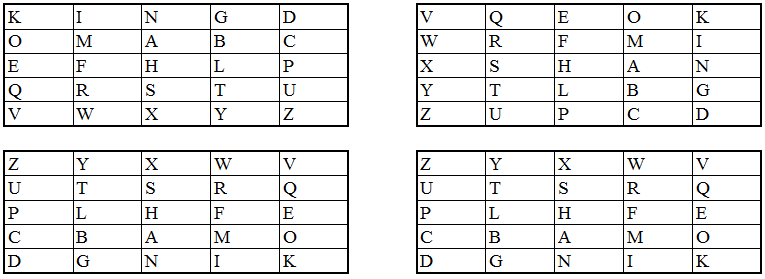
****

Рисунок 1 – Пример заполненных 4 квадратов

Таким образом, слово Hello будет зашифровано как HXFAU

Шифр ADFGVX. Это один из самых известных шифров времён Первой мировой войны, который использовался немецкой армией на западном фронте. Особенность шифра заключается в том, что он построен на соединении базовых операций замены и перестановки. Часть шифра, отвечающая замене, основывается на квадрате Полибия. Данный шифр представляет собой моноалфавитный шифр билитеральной замены. Ключом к замене является буквосочетание ADFGVX.

Алгоритм шифрования: cтроится квадрат 6х6, в который в произвольном порядке вписываются все буквы латинского алфавита и цифры. В таблице находится исходная буква и заменяется биграммой букв строки и столбца. Данная последовательность подписывается под лозунговым словом, которое выбирается заранее и должно быть известно обеим сторонам. Из лозунгового слова, начиная с первой по алфавиту буквы, выписывается столбец в строку, причем длина отдельного слова должна равняться длине лозунгового слова. При необходимости, если длина последнего слова меньше, добавляется любая биграмма.

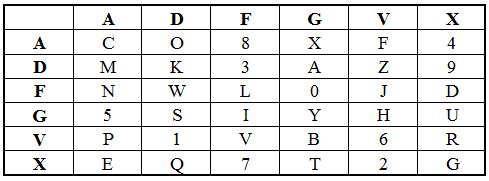
****

Рисунок 2 - Пример заполненной таблицы

Таким образом, слово Hello получится, как GVXAFFFFAD

**Разработанная программа:**

Программа разрабатывалась на языке C# при помощи Windows Forms. Интерфейс программы изображен на рисунке 3.

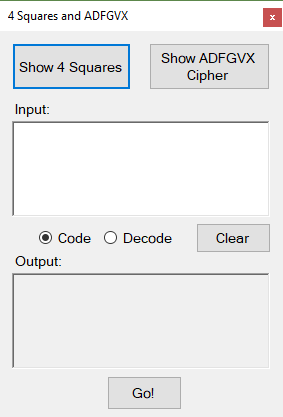


Рисунок 3 – Интерфейс программы

Программа позволяет шифровать и дешифровать текст английском языке. Шифруемый текст должен быть введен вручную в текстовое поле. Кнопка “Clear” позволяет очистить текстовое поле. Программа работает как шифратор, так и дешифратор в зависимости от выбора пользователя. Программа шифрует строку сначала шифром четырех квадратов (выводя результат на экран), после чего шифрует результат шифром ADFGVX, и выводит результат в выходное поле. Для просмотра 4 квадратов и таблицу шифра ADFGVX, необходимо нажать на одну из кнопок. Результаты нажатия обоих кнопок изображены на рисунках 4 и 5. Чтобы изменить таблицу ADFGVX шифра, необходимо залезть в файл cipher.txt.

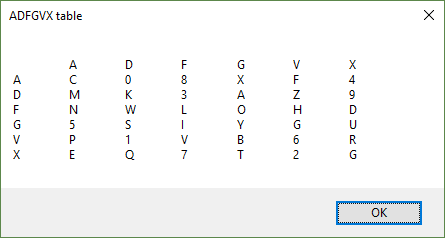


Рисунок 4 – Таблица ADFGVX

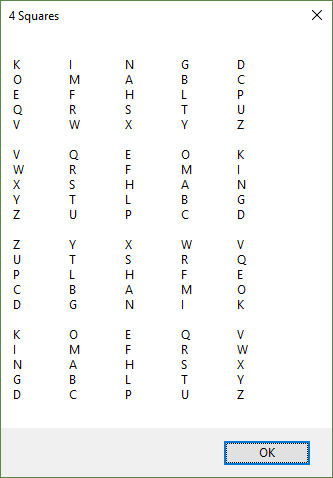


Рисунок 5 – Четыре квадрата

**Вывод:**

В данной практической работе была разработана программа шифратор-дешифратор, работающая по принципу шифра 4-х квадратов и шифра ADFGVX. Данная работа позволила понять принцип шифрования, реализовать его при помощи языка программирования, и проверить работоспособность на практике.

**Листинг:**

MainForm.cs:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Cipher

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

try

{

if (radioButton\_Code.Checked)

{

Squares s = new Squares();

string codeString = s.EncodeToSquaresCipher(richTextBox\_Input.Text.ToUpper());

MessageBox.Show("Code in 4 Squares Cipher: " + codeString);

Adfgvx a = new Adfgvx();

richTextBox\_Output.Text = a.EncodeToAdfgvx(codeString);

}

else

{

Adfgvx a = new Adfgvx();

string decodeString = a.DecodeFromAdfgvx(richTextBox\_Input.Text.ToUpper());

MessageBox.Show("Decode from Adfgvx Cipher: " + decodeString);

Squares s = new Squares();

richTextBox\_Output.Text = s.DecodeFromSquaresCipher(decodeString);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void button2\_Click(object sender, EventArgs e)

{

Adfgvx a = new Adfgvx();

MessageBox.Show(a.ToString(),"ADFGVX table");

}

private void button3\_Click(object sender, EventArgs e)

{

Squares s = new Squares();

MessageBox.Show(s.ToString(),"4 Squares");

}

private void button\_Clear\_Click(object sender, EventArgs e)

{

richTextBox\_Input.Text = "";

richTextBox\_Output.Text = "";

}

}

}

Squares.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Cipher

{

class Squares

{

char[,] square1 = new char[5, 5];

char[,] square2 = new char[5, 5];

char[,] square3 = new char[5, 5];

char[,] square4 = new char[5, 5];

public Squares()

{

#region Square1

square1[0, 0] = 'K';

square1[0, 1] = 'I';

square1[0, 2] = 'N';

square1[0, 3] = 'G';

square1[0, 4] = 'D';

square1[1, 0] = 'O';

square1[1, 1] = 'M';

square1[1, 2] = 'A';

square1[1, 3] = 'B';

square1[1, 4] = 'C';

square1[2, 0] = 'E';

square1[2, 1] = 'F';

square1[2, 2] = 'H';

square1[2, 3] = 'L';

square1[2, 4] = 'P';

square1[3, 0] = 'Q';

square1[3, 1] = 'R';

square1[3, 2] = 'S';

square1[3, 3] = 'T';

square1[3, 4] = 'U';

square1[4, 0] = 'V';

square1[4, 1] = 'W';

square1[4, 2] = 'X';

square1[4, 3] = 'Y';

square1[4, 4] = 'Z';

#endregion

#region Square2

square2[0, 0] = 'V';

square2[0, 1] = 'Q';

square2[0, 2] = 'E';

square2[0, 3] = 'O';

square2[0, 4] = 'K';

square2[1, 0] = 'W';

square2[1, 1] = 'R';

square2[1, 2] = 'F';

square2[1, 3] = 'M';

square2[1, 4] = 'I';

square2[2, 0] = 'X';

square2[2, 1] = 'S';

square2[2, 2] = 'H';

square2[2, 3] = 'A';

square2[2, 4] = 'N';

square2[3, 0] = 'Y';

square2[3, 1] = 'T';

square2[3, 2] = 'L';

square2[3, 3] = 'B';

square2[3, 4] = 'G';

square2[4, 0] = 'Z';

square2[4, 1] = 'U';

square2[4, 2] = 'P';

square2[4, 3] = 'C';

square2[4, 4] = 'D';

#endregion

#region Square3

square3[0, 0] = 'Z';

square3[0, 1] = 'Y';

square3[0, 2] = 'X';

square3[0, 3] = 'W';

square3[0, 4] = 'V';

square3[1, 0] = 'U';

square3[1, 1] = 'T';

square3[1, 2] = 'S';

square3[1, 3] = 'R';

square3[1, 4] = 'Q';

square3[2, 0] = 'P';

square3[2, 1] = 'L';

square3[2, 2] = 'H';

square3[2, 3] = 'F';

square3[2, 4] = 'E';

square3[3, 0] = 'C';

square3[3, 1] = 'B';

square3[3, 2] = 'A';

square3[3, 3] = 'M';

square3[3, 4] = 'O';

square3[4, 0] = 'D';

square3[4, 1] = 'G';

square3[4, 2] = 'N';

square3[4, 3] = 'I';

square3[4, 4] = 'K';

#endregion

#region Square4

square4[0, 0] = 'K';

square4[0, 1] = 'O';

square4[0, 2] = 'E';

square4[0, 3] = 'Q';

square4[0, 4] = 'V';

square4[1, 0] = 'I';

square4[1, 1] = 'M';

square4[1, 2] = 'F';

square4[1, 3] = 'R';

square4[1, 4] = 'W';

square4[2, 0] = 'N';

square4[2, 1] = 'A';

square4[2, 2] = 'H';

square4[2, 3] = 'S';

square4[2, 4] = 'X';

square4[3, 0] = 'G';

square4[3, 1] = 'B';

square4[3, 2] = 'L';

square4[3, 3] = 'T';

square4[3, 4] = 'Y';

square4[4, 0] = 'D';

square4[4, 1] = 'C';

square4[4, 2] = 'P';

square4[4, 3] = 'U';

square4[4, 4] = 'Z';

#endregion

}

public string EncodeToSquaresCipher(string word)

{

string result = String.Empty;

byte b = 1;

foreach (var sybmol in word)

{

if (!Char.IsLetter(sybmol) || sybmol == 'J')

result += sybmol;

else

result += GetCharToSquareCipher(sybmol, b);

if (b == 1)

b++;

else if (b == 2)

b--;

}

return result;

}

public string DecodeFromSquaresCipher(string word)

{

string result = String.Empty;

byte b = 3;

foreach (var sybmol in word)

{

if (!Char.IsLetter(sybmol) || sybmol=='J')

result += sybmol;

else

result += GetCharToSquareCipher(sybmol, b);

if (b == 3)

b++;

else if (b == 4)

b--;

}

return result;

}

public char GetCharToSquareCipher(char symbol,byte square)

{

char newSymbol=' ';

if (square == 1)

{

for (int i = 0; i < 5; i++)

for (int j = 0; j < 5; j++)

if (square1[i, j] == symbol)

return square3[i, j];

}

else if (square == 2)

{

for (int i = 0; i < 5; i++)

for (int j = 0; j < 5; j++)

if (square2[i, j] == symbol)

return square4[i, j];

}

else if (square == 3)

{

for (int i = 0; i < 5; i++)

for (int j = 0; j < 5; j++)

if (square3[i, j] == symbol)

return square1[i, j];

}

else if (square == 4)

{

for (int i = 0; i < 5; i++)

for (int j = 0; j < 5; j++)

if (square4[i, j] == symbol)

return square2[i, j];

}

return newSymbol;

}

public override string ToString()

{

string temp1 = "", temp2="", temp3="", temp4="";

for (int i=0; i<5; i++)

{

for (int j = 0; j < 5; j++)

{

temp1 += square1[i, j].ToString() + '\t';

temp2 += square2[i, j].ToString() + '\t';

temp3 += square3[i, j].ToString() + '\t';

temp4 += square4[i, j].ToString() + '\t';

}

temp1 += '\n';

temp2 += '\n';

temp3 += '\n';

temp4 += '\n';

}

return temp1+'\n' + temp2 + '\n' + temp3 + '\n' + temp4 ;

}

}

}

Adfgvx.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

using System.Threading.Tasks;

namespace Cipher

{

class Adfgvx

{

private Dictionary<char, Dictionary<char, char>> cipher;

public Adfgvx()

{

cipher = new Dictionary<char, Dictionary<char, char>>();

string str = "ADFGVX";

StreamReader read = new StreamReader("cipher.txt");

foreach (var symbol1 in str)

{

cipher[symbol1] = new Dictionary<char, char>();

foreach (var symbol2 in str)

cipher[symbol1][symbol2] = read.ReadLine()[0];

}

}

public string EncodeToAdfgvx(string word)

{

string result = String.Empty;

bool end=false;

foreach (var symbol in word)

{

if (!Char.IsLetterOrDigit(symbol) || symbol == 'J')

{

result += symbol;

continue;

}

end = false;

foreach (var horizontal in cipher.Keys)

{

if (end) break;

foreach (var vertical in cipher[horizontal].Keys)

if (cipher[horizontal][vertical] == symbol)

{

result += horizontal.ToString() + vertical;

end = true;

break;

}

}

}

return result;

}

public string DecodeFromAdfgvx(string word)

{

string result = String.Empty, temp=String.Empty;

foreach (var symbol in word)

{

if (!Char.IsLetterOrDigit(symbol) || symbol=='J')

{

result += symbol;

continue;

}

temp += symbol;

if (temp.Length == 2)

{

result += cipher[temp[0]][temp[1]];

temp = String.Empty;

}

}

return result;

}

public override string ToString()

{

string temp = String.Empty;

temp += " \t" + "A\t" + "D\t" + "F\t" + "G\t" + "V\t" + "X\n";

foreach (var h in cipher.Keys)

{

temp += h.ToString().ToUpper() + '\t';

foreach (var v in cipher[h].Values)

temp += v.ToString().ToUpper() + '\t';

temp += '\n';

}

return temp;

}

public bool CheckCipher()

{

bool b = true;

Dictionary<char, bool> d = new Dictionary<char, bool>();

for (char i = 'A'; i <= 'Z'; i++)

{

if (i == 'J')

d[i] = true;

else

d[i] = false;

}

foreach (var horizontal in cipher.Keys)

{

foreach (var vertical in cipher[horizontal].Keys)

d[cipher[horizontal][vertical]] = true;

}

foreach (var item in d.Values)

{

if (!item)

{

b = false;

break;

}

}

return b;

}

}

}