Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

Направление 09.03.01 – «Информатика и вычислительная техника»

Дисциплина: «Защита информации»

Профиль: «Разработка программно-информационных систем»

Семестр 5

ОТЧЕТ

по лабораторной работе №1

Тема: «Шифр “Роторные машины”»

Выполнил: студент группы РИС 20-2б

Уржумов В.И. \_\_\_\_\_\_\_\_\_\_\_

Проверил: доцент кафедры ИТАС

Шереметьев В. Г. \_\_\_\_\_\_\_\_\_\_\_

Дата \_\_\_\_\_\_

Пермь, 2022

**ЦЕЛЬ РАБОТЫ**

Получить практические навыки по применению «Роторные машины».

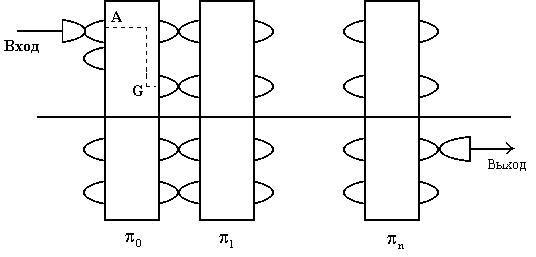
**ЗАДАНИЕ**

Реализовать шифрование текстового сообщения, используя шифр «Роторные машины»

**ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

В 20-х годах XX века были изобретены электромеханические устройства шифрования, автоматизирующие процесс шифрования. Принцип работы таких машин основан на многоалфавитной замене символов исходного текста по длинному ключу согласно версии шифра Вижинера. Большинство из них - американская машина SIGABA (М-134), английская TYPEX, немецкая ENIGMA, японская PURPLE были роторными машинами.

Главной деталью роторной машины является ротор (или колесо) с проволочными перемычками внутри. Ротор имеет форму диска (размером с хоккейную шайбу). На каждой стороне диска расположены равномерно по окружности m электрических контактов, где m - число знаков алфавита (в случае латинского алфавита m = 26). Каждый контакт на передней стороне диска соединен с одним из контактов на задней стороне. В результате электрический сигнал, представляющий знак, будет переставлен в соответствии с тем, как он проходит через ротор от передней стороны к задней. Например, ротор можно закоммутировать проволочными перемычками для подстановки G вместо A, U вместо В, L вместо С и т.д.



Банк роторов

При повороте ротора из одного положения в другое подстановка, которую он осуществляет в приходящем сигнале, будет изменяться. В общем случае эту подстановку можно записать в виде

T = C j p C -j

где p - подстановка, реализуемая ротором в его начальном положении; С - циклический сдвиг на одну позицию; C j - циклический сдвиг на j позиций.

Например, если начальная подстановка ротора p(А) = G и ротор сдвигается на три позиции (j = 3), то открытый текст D будет против того контакта ротора, который используется

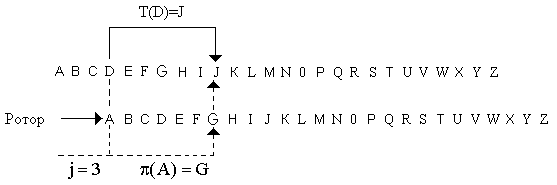


Схема формирования подстановки при сдвиге ротора (j =3) для представления открытого текста А, а шифрованный текст J окажется против того контакта ротора, который используется для представления шифрованного текста G , и результирующая подстановка Т(D) = G при j = 3. Алгебраически это записывается в виде

Т (D) = С3pС-3 (D) = С3 p (А) = С3 (G) = J.

Роторы можно объединить в банк роторов таким образом, чтобы выходные контакты одного ротора касались входных контактов следующего ротора. При этом электрический импульс от нажатой клавиши с буквой исходного текста, входящий с одного конца банка роторов, будет переставляться каждым из роторов, до тех пор, пока не покинет банк.

Такой банк может реализовывать большое число подстановок, соответствующих различным комбинациям положений роторов. Для получения сильной криптографической системы расположение роторов должно меняться при переходе от знака к знаку сообщения.

Роторная машина состоит из банка роторов и механизма для изменения положения роторов с каждым зашифрованным знаком, объединенного с устройствами ввода и вывода, такими как устройство считывания с перфоленты и печатающее устройство.

Простейшее из возможных движений ротора - это движение по принципу одометра; оно использовалось в немецкой машине Enigma во время второй мировой войны. При шифровании одного знака правое крайнее колесо поворачивается на одну позицию. Когда это (и любое другое) колесо переместится на m позиций и совершит полный оборот, колесо, расположенное слева от него, передвинется на одну позицию, и процесс будет повторяться. Этот процесс проведет банк роторов сквозь все его возможные положения, прежде чем цикл повторится. Поскольку все роторы перемещаются с разными скоростями, период n-роторной машины составляет 26n (при m = 26).

Для закона движения ротора желательны следующие характеристики:

* период должен быть большим;
* после шифрования каждого знака все роторы или большая их часть должны повернуться друг относительно друга.

Движение по принципу одометра оптимально в смысле первого требования, но совершенно неудовлетворительно в отношении второго требования. Улучшение движения по принципу одометра можно получить, если поворачивать каждый ротор более чем на одну позицию. Если смещения каждого ротора не имеют общих множителей с объемом алфавита m, то период останется максимальным.

Другое решение заключается в ограничении числа допустимых остановочных мест для каждого ротора за счет введения внешнего фиксирующего кольца, на котором определенным способом зафиксированы места остановок. При использовании латинского алфавита можно заставить машины поворачиваться и останавливаться следующим образом. Первому колесу разрешается останавливаться в каждой из 26 позиций, второму колесу - только в 25 позициях, третьему колесу - только в 23 позициях, и так далее до шестого колеса, которому разрешается останавливаться только в 17 позициях. Период такой роторной машины теперь составляет 101 млн, а не266»309 млн, как в случае движения по принципу одометра. Потеря в длине периода с успехом окупается полученной сложностью движения роторов. Теперь второе требование удовлетворяется довольно хорошо, поскольку каждое из колес перемещается после шифрования каждого знака и многие колеса могут двигаться друг относительно друга.

Роторная машина может быть настроена по ключу изменением любых ее переменных:

* роторов;
* порядка расположения роторов;
* числа мест остановки на колесо;
* характера движения и т.д.

Поскольку перекоммутировать роторы трудно, то обычно на практике машины обеспечивали комплектом роторов, в котором находилось больше роторов, чем можно одновременно поместить в машину. Первичная настройка по ключу производилась выбором роторов, составляющих комплект. Вторичная настройка по ключу производилась выбором порядка расположения роторов в машине и установкой параметров, управляющих движением машины. С целью затруднения расшифрованияшифртекстов противником роторы ежедневно переставляли местами или заменяли. Большая часть ключа определяла начальные положения роторов (2б3=17576 возможных установок) и конкретные перестановки на коммутационной доске, с помощью которой осуществлялась начальная перестановка исходного текста до его шифрования (26!= 4·1026 возможностей).

Роторные машины были самыми важными криптографическими устройствами во время второй мировой войны и доминировали по крайней мере до конца 50-х годов.

**ХОД РАБОТЫ**

На рисунке 1 представлена форма, в первое поле вводится шифруемая фраза. По нажатию кнопки «Запуск» исходный текст шифруется и выводится во второе поле. Пример

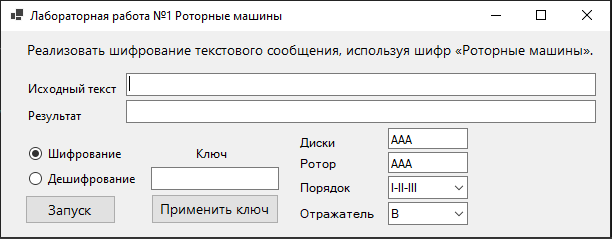


Рисунок 1- Форма для шифрования/дешифрования

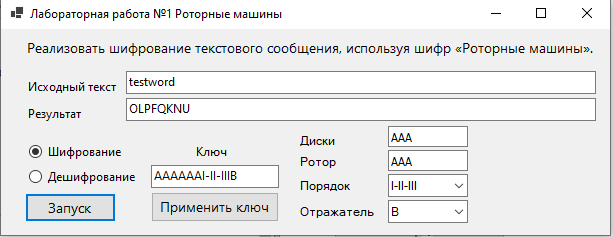


Рисунок 2-Пример работы программы

**ПРИЛОЖЕНИЕ А**

**Листинг программы**

public partial class Labs1 : Form

{

EnigmaMachine machine = new EnigmaMachine();

EnigmaSettings eSettings = new EnigmaSettings();

public Labs1()

{

InitializeComponent();

comboBox1.SelectedIndex = 0;

comboBox2.SelectedIndex = 1;

textBox4.Text = "AAA";

textBox5.Text = "AAA";

}

private void Labs1\_FormClosed(object sender, FormClosedEventArgs e)

{

MainForm form1 = new MainForm();

form1.Show();

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

if (radioButton1.Checked == true)

radioButton2.Checked = false;

else if (radioButton1.Checked == false)

{

radioButton2.Checked = true;

textBox1.Text = textBox2.Text;

textBox2.Text = "";

}

}

public void Encryped()

{

string message = textBox1.Text;

message = message.Replace(" ", "").ToUpper();

machine.setSettings(eSettings.rings, eSettings.grund, eSettings.order, eSettings.reflector);

foreach (string plug in eSettings.plugs)

{

char[] p = plug.ToCharArray();

machine.addPlug(p[0], p[1]);

}

textBox2.Text = machine.runEnigma(message);

}

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

{

SettingEngima(eSettings);

Encryped();

textBox3.Text = textBox4.Text + textBox5.Text + comboBox1.SelectedItem.ToString() + comboBox2.SelectedItem.ToString();

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

if ((!Regex.IsMatch(textBox1.Text, @"^[a-zA-Z ]+$")) && textBox1.Text.Length > 0)

{

MessageBox.Show("Недопустимый символ, доступны только A-Z");

}

}

public void SettingEngima(EnigmaSettings e)

{

string rings = textBox4.Text;

string rotors = textBox5.Text;

string order = comboBox1.SelectedItem.ToString();

string reflector = comboBox2.SelectedItem.ToString();

if (textBox3.Text == "")

e.setDefault();

else

{

//rings

if (rings == "")

e.grund = new char[] { 'A', 'A', 'A' };

else

e.grund = rings.ToCharArray();

//rotors

if (rotors == "")

e.grund = new char[] { 'A', 'A', 'A' };

else

e.grund = rotors.ToCharArray();

//order

e.order = "I-II-III";

e.order = order;

//reflector

e.reflector = 'B';

e.reflector = reflector.ToCharArray()[0];

}

}

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

{

textBox4.Text = "";

textBox5.Text = "";

string order = "";

char[] key = textBox3.Text.ToCharArray();

for (int i = 0; i < 3; i++)

textBox4.Text += key[i].ToString();

for (int i = 3; i < 6; i++)

textBox5.Text += key[i].ToString();

for (int i = 6; i < 14; i++)

{

order += key[i];

for (int j = 0; j < 6; j++)

{

if (comboBox1.Items[j].ToString() == order)

comboBox1.SelectedIndex = j;

}

}

for (int j = 0; j < 3; j++)

{

if (comboBox2.Items[j].ToString() == key[14].ToString())

comboBox2.SelectedIndex = j;

}

}

}

class EnigmaMachine

{

private Dictionary<char, char> plugBoard;

private Rotor[] rotors;

private Rotor reflector;

private const string alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

private const string rotorIconf = "EKMFLGDQVZNTOWYHXUSPAIBRCJ";

private const string rotorIIconf = "AJDKSIRUXBLHWTMCQGZNPYFVOE";

private const string rotorIIIconf = "BDFHJLCPRTXVZNYEIWGAKMUSQO";

private const string reflectorAconf = "EJMZALYXVBWFCRQUONTSPIKHGD";

private const string reflectorBconf = "YRUHQSLDPXNGOKMIEBFZCWVJAT";

private const string reflectorCconf = "FVPJIAOYEDRZXWGCTKUQSBNMHL";

private class Rotor

{

private int outerPosition;

public char outerChar { get; set; }

private string wiring;

private char turnOver;

public string name { get; }

public char ring { get; set; }

public int[] map { get; }

public int[] revMap { get; }

public Rotor(string w, char to, string n)

{

turnOver = to;

outerPosition = 0;

ring = 'A';

name = n;

map = new int[26];

revMap = new int[26];

setWiring(w);

}

public void setWiring(string newW)

{

wiring = newW;

outerChar = wiring.ToCharArray()[outerPosition];

for (int i = 0; i < 26; i++)

{

int match = wiring.ToCharArray()[i] - 65;

map[i] = (26 + match - i) % 26;

revMap[match] = (26 + i - match) % 26;

}

}

public void setOuterPosition(int i)

{

outerPosition = i;

outerChar = alphabet.ToCharArray()[outerPosition];

}

public int getOuterPosition()

{

return outerPosition;

}

public void setOuterChar(char c)

{

outerChar = c;

outerPosition = alphabet.IndexOf(outerChar);

}

public void step()

{

outerPosition = (outerPosition + 1) % 26;

outerChar = alphabet.ToCharArray()[outerPosition];

}

public bool isInTurnOver()

{

return outerChar == turnOver;

}

}

private void rotateRotors(Rotor[] r)

{

if (r.Length == 3)

{

if (r[1].isInTurnOver())

{

r[0].step();

r[1].step();

}

else if (r[2].isInTurnOver())

r[1].step();

r[2].step();

}

}

private char rotorMap(char c, bool reverse)

{

int cPos = c - 65;

if (!reverse)

for (int i = rotors.Length - 1; i >= 0; i--)

cPos = rotorValue(rotors[i], cPos, reverse);

else

for (int i = 0; i < rotors.Length; i++)

cPos = rotorValue(rotors[i], cPos, reverse);

return alphabet.ToCharArray()[cPos];

}

private int rotorValue(Rotor r, int cPos, bool reverse)

{

int rPos = r.ring - 65;

int d;

if (!reverse)

d = r.map[(26 + cPos + r.getOuterPosition() - rPos) % 26];

else

d = r.revMap[(26 + cPos + r.getOuterPosition() - rPos) % 26];

return (cPos + d) % 26;

}

private char reflectorMap(char c)

{

int cPos = c - 65;

cPos = (cPos + reflector.map[cPos]) % 26;

return alphabet.ToCharArray()[cPos];

}

public EnigmaMachine()

{

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

Rotor rI = new Rotor(rotorIconf, 'Q', "I");

Rotor rII = new Rotor(rotorIIconf, 'E', "II");

Rotor rIII = new Rotor(rotorIIIconf, 'V', "III");

rotors = new Rotor[] { rI, rII, rIII };

reflector = new Rotor(reflectorAconf, ' ', "");

}

public void setReflector(char conf)

{

if (conf != 'A' && conf != 'B' && conf != 'C')

{

throw new ArgumentException("Invalid argument");

}

string wiring = "";

switch (conf)

{

case 'A':

wiring = reflectorAconf;

break;

case 'B':

wiring = reflectorBconf;

break;

case 'C':

wiring = reflectorCconf;

break;

}

reflector.setWiring(wiring);

}

public void setSettings(char[] rings, char[] grund)

{

if (rings.Length != rotors.Length || grund.Length != rotors.Length)

throw new ArgumentException("Invalid argument lengths");

for (int i = 0; i < rotors.Length; i++)

{

rotors[i].ring = char.ToUpper(rings[i]);

rotors[i].setOuterChar(char.ToUpper(grund[i]));

}

}

public void setSettings(char[] rings, char[] grund, string rotorOrder)

{

Rotor rI = null;

Rotor rII = null;

Rotor rIII = null;

for (int i = 0; i < rotors.Length; i++)

{

if (rotors[i].name == "I")

rI = rotors[i];

if (rotors[i].name == "II")

rII = rotors[i];

if (rotors[i].name == "III")

rIII = rotors[i];

}

string[] order = rotorOrder.Split('-');

for (int i = 0; i < order.Length; i++)

{

if (order[i] == "I")

rotors[i] = rI;

if (order[i] == "II")

rotors[i] = rII;

if (order[i] == "III")

rotors[i] = rIII;

}

setSettings(rings, grund);

}

public void setSettings(char[] rings, char[] grund, string rotorOrder, char reflectorConf)

{

setReflector(reflectorConf);

setSettings(rings, grund, rotorOrder);

}

public string runEnigma(string msg)

{

StringBuilder encryptedMessage = new StringBuilder();

msg = msg.ToUpper();

foreach (char c in msg)

encryptedMessage.Append(encryptChar(c));

return encryptedMessage.ToString();

}

private char encryptChar(char c)

{

rotateRotors(rotors);

if (plugBoard.ContainsKey(c))

c = plugBoard[c];

c = rotorMap(c, false);

c = reflectorMap(c);

c = rotorMap(c, true);

if (plugBoard.ContainsKey(c))

{

c = plugBoard[c];

}

return c;

}

public void addPlug(char c, char cc)

{

if (char.IsLetter(c) && char.IsLetter(cc))

{

c = char.ToUpper(c);

cc = char.ToUpper(cc);

if (c != cc && !plugBoard.ContainsKey(c))

{

plugBoard.Add(c, cc);

plugBoard.Add(cc, c);

}

}

else

throw new ArgumentException("Invalid character");

}

}

public class EnigmaSettings

{

public char[] rings { get; set; }

public char[] grund { get; set; }

public string order { get; set; }

public char reflector { get; set; }

public List<string> plugs = new List<string>();

public EnigmaSettings()

{

setDefault();

}

public void setDefault()

{

rings = new char[] { 'A', 'A', 'A' };

grund = new char[] { 'A', 'A', 'A' };

order = "I-II-III";

reflector = 'B';

plugs.Clear();

}

}