Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ**

Современные симметричные шифры. AES-128/AES-256

Руководитель от университета \_\_\_\_\_\_\_\_\_\_ Сидарас А.А.

подпись, дата

Студент КИ19-06б, 031940417 \_\_\_\_\_\_\_\_\_\_ Шнайдер А.В.

подпись, дата

Красноярск 2020

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc73175109)

[Ход выполнения работы 4](#_Toc73175110)

[1. Описание алгоритма шифрования 4](#_Toc73175111)

[2. Листинг составленной программы 4](#_Toc73175112)

[3. Контрольные примеры работы программы 17](#_Toc73175113)

[ЗАКЛЮЧЕНИЕ 21](#_Toc73175114)

ВВЕДЕНИЕ

Задание к работе: реализовать шифр AES-256 на любом из языков программирования кроме Python (с оговорками) с любым интерфейсом.

Ход выполнения работы

1. Описание алгоритма шифрования

Предварительно входные данные разбиваются на блоки по 16 байт, если полный размер не кратен 16 байтам, то данные дополняется до размера, кратного 16 байтам. Блоки представляются в виде матрицы 4x4 — state. Далее происходит процедура расширения ключа и к каждому блоку state применяются операции 2-4. Итак, алгоритм состоит из следующих шагов:

1. Расширение ключа - KeyExpansion;
2. Начальный раунд - сложение state с основным ключом;
3. 13 раундов шифрования, каждый из которых состоит из преобразований:

* SubBytes
* ShiftRows
* MixColumns
* AddRoundKey

1. Финальный раунд, состоящий из преобразований:

* SubBytes
* ShiftRows
* AddRoundKey

2. Листинг составленной программы

Файл programs.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace IS\_LAB3

{

class Program

{

[STAThread]

static void Main(string[] args)

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

Файл Form1.designer.cs:

namespace IS\_LAB3

{

partial class Form1

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.splitContainer1 = new System.Windows.Forms.SplitContainer();

this.InputText = new System.Windows.Forms.TextBox();

this.OutputText = new System.Windows.Forms.TextBox();

this.splitContainer2 = new System.Windows.Forms.SplitContainer();

this.tableLayoutPanel1 = new System.Windows.Forms.TableLayoutPanel();

this.splitContainer3 = new System.Windows.Forms.SplitContainer();

this.label1 = new System.Windows.Forms.Label();

this.KeyTextBox = new System.Windows.Forms.MaskedTextBox();

this.DecryptButton = new System.Windows.Forms.Button();

this.EncryptButton = new System.Windows.Forms.Button();

((System.ComponentModel.ISupportInitialize)(this.splitContainer1)).BeginInit();

this.splitContainer1.Panel1.SuspendLayout();

this.splitContainer1.Panel2.SuspendLayout();

this.splitContainer1.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.splitContainer2)).BeginInit();

this.splitContainer2.Panel1.SuspendLayout();

this.splitContainer2.Panel2.SuspendLayout();

this.splitContainer2.SuspendLayout();

this.tableLayoutPanel1.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.splitContainer3)).BeginInit();

this.splitContainer3.Panel1.SuspendLayout();

this.splitContainer3.Panel2.SuspendLayout();

this.splitContainer3.SuspendLayout();

this.SuspendLayout();

//

// splitContainer1

//

this.splitContainer1.Dock = System.Windows.Forms.DockStyle.Fill;

this.splitContainer1.Location = new System.Drawing.Point(0, 0);

this.splitContainer1.Margin = new System.Windows.Forms.Padding(2);

this.splitContainer1.Name = "splitContainer1";

this.splitContainer1.Orientation = System.Windows.Forms.Orientation.Horizontal;

//

// splitContainer1.Panel1

//

this.splitContainer1.Panel1.Controls.Add(this.InputText);

//

// splitContainer1.Panel2

//

this.splitContainer1.Panel2.Controls.Add(this.OutputText);

this.splitContainer1.Size = new System.Drawing.Size(703, 436);

this.splitContainer1.SplitterDistance = 218;

this.splitContainer1.SplitterWidth = 3;

this.splitContainer1.TabIndex = 7;

this.splitContainer1.TabStop = false;

//

// InputText

//

this.InputText.BackColor = System.Drawing.Color.White;

this.InputText.Dock = System.Windows.Forms.DockStyle.Fill;

this.InputText.Font = new System.Drawing.Font("Consolas", 13.8F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.InputText.ForeColor = System.Drawing.SystemColors.MenuText;

this.InputText.Location = new System.Drawing.Point(0, 0);

this.InputText.Margin = new System.Windows.Forms.Padding(2);

this.InputText.Multiline = true;

this.InputText.Name = "InputText";

this.InputText.Size = new System.Drawing.Size(703, 218);

this.InputText.TabIndex = 0;

this.InputText.TextChanged += new System.EventHandler(this.InputText\_TextChanged);

//

// OutputText

//

this.OutputText.Dock = System.Windows.Forms.DockStyle.Fill;

this.OutputText.Font = new System.Drawing.Font("Consolas", 13.8F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.OutputText.Location = new System.Drawing.Point(0, 0);

this.OutputText.Margin = new System.Windows.Forms.Padding(2);

this.OutputText.Multiline = true;

this.OutputText.Name = "OutputText";

this.OutputText.Size = new System.Drawing.Size(703, 215);

this.OutputText.TabIndex = 1;

//

// splitContainer2

//

this.splitContainer2.Dock = System.Windows.Forms.DockStyle.Fill;

this.splitContainer2.FixedPanel = System.Windows.Forms.FixedPanel.Panel1;

this.splitContainer2.Location = new System.Drawing.Point(0, 0);

this.splitContainer2.Margin = new System.Windows.Forms.Padding(2);

this.splitContainer2.Name = "splitContainer2";

this.splitContainer2.Orientation = System.Windows.Forms.Orientation.Horizontal;

//

// splitContainer2.Panel1

//

this.splitContainer2.Panel1.Controls.Add(this.tableLayoutPanel1);

//

// splitContainer2.Panel2

//

this.splitContainer2.Panel2.Controls.Add(this.splitContainer1);

this.splitContainer2.Size = new System.Drawing.Size(703, 569);

this.splitContainer2.SplitterDistance = 130;

this.splitContainer2.SplitterWidth = 3;

this.splitContainer2.TabIndex = 9;

//

// tableLayoutPanel1

//

this.tableLayoutPanel1.ColumnCount = 2;

this.tableLayoutPanel1.ColumnStyles.Add(new System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent, 51.1254F));

this.tableLayoutPanel1.ColumnStyles.Add(new System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent, 48.8746F));

this.tableLayoutPanel1.Controls.Add(this.splitContainer3, 0, 1);

this.tableLayoutPanel1.Controls.Add(this.DecryptButton, 1, 0);

this.tableLayoutPanel1.Controls.Add(this.EncryptButton, 0, 0);

this.tableLayoutPanel1.Dock = System.Windows.Forms.DockStyle.Fill;

this.tableLayoutPanel1.Location = new System.Drawing.Point(0, 0);

this.tableLayoutPanel1.Margin = new System.Windows.Forms.Padding(2);

this.tableLayoutPanel1.Name = "tableLayoutPanel1";

this.tableLayoutPanel1.RowCount = 2;

this.tableLayoutPanel1.RowStyles.Add(new System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent, 53.90625F));

this.tableLayoutPanel1.RowStyles.Add(new System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent, 46.09375F));

this.tableLayoutPanel1.Size = new System.Drawing.Size(703, 130);

this.tableLayoutPanel1.TabIndex = 8;

//

// splitContainer3

//

this.splitContainer3.Location = new System.Drawing.Point(2, 72);

this.splitContainer3.Margin = new System.Windows.Forms.Padding(2);

this.splitContainer3.Name = "splitContainer3";

this.splitContainer3.Orientation = System.Windows.Forms.Orientation.Horizontal;

//

// splitContainer3.Panel1

//

this.splitContainer3.Panel1.Controls.Add(this.label1);

//

// splitContainer3.Panel2

//

this.splitContainer3.Panel2.Controls.Add(this.KeyTextBox);

this.splitContainer3.Size = new System.Drawing.Size(355, 54);

this.splitContainer3.SplitterDistance = 25;

this.splitContainer3.SplitterWidth = 3;

this.splitContainer3.TabIndex = 1;

//

// label1

//

this.label1.AutoSize = true;

this.label1.Dock = System.Windows.Forms.DockStyle.Fill;

this.label1.Location = new System.Drawing.Point(0, 0);

this.label1.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(179, 13);

this.label1.TabIndex = 0;

this.label1.Text = "Ключ не более 32 ASCII символов";

this.label1.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;

this.label1.Click += new System.EventHandler(this.label1\_Click);

//

// KeyTextBox

//

this.KeyTextBox.AsciiOnly = true;

this.KeyTextBox.Culture = new System.Globalization.CultureInfo("en-US");

this.KeyTextBox.Dock = System.Windows.Forms.DockStyle.Fill;

this.KeyTextBox.Location = new System.Drawing.Point(0, 0);

this.KeyTextBox.Margin = new System.Windows.Forms.Padding(2);

this.KeyTextBox.Name = "KeyTextBox";

this.KeyTextBox.Size = new System.Drawing.Size(355, 20);

this.KeyTextBox.TabIndex = 0;

//

// DecryptButton

//

this.DecryptButton.Dock = System.Windows.Forms.DockStyle.Fill;

this.DecryptButton.Location = new System.Drawing.Point(361, 2);

this.DecryptButton.Margin = new System.Windows.Forms.Padding(2);

this.DecryptButton.Name = "DecryptButton";

this.DecryptButton.Size = new System.Drawing.Size(340, 66);

this.DecryptButton.TabIndex = 4;

this.DecryptButton.Text = "Расшифровать";

this.DecryptButton.UseVisualStyleBackColor = true;

this.DecryptButton.Click += new System.EventHandler(this.ButtonDecrypt\_Clicked);

//

// EncryptButton

//

this.EncryptButton.AutoSize = true;

this.EncryptButton.Dock = System.Windows.Forms.DockStyle.Fill;

this.EncryptButton.Location = new System.Drawing.Point(2, 2);

this.EncryptButton.Margin = new System.Windows.Forms.Padding(2);

this.EncryptButton.Name = "EncryptButton";

this.EncryptButton.Size = new System.Drawing.Size(355, 66);

this.EncryptButton.TabIndex = 3;

this.EncryptButton.Text = "Зашифровать";

this.EncryptButton.UseVisualStyleBackColor = true;

this.EncryptButton.Click += new System.EventHandler(this.ButtonEncrypt\_Clicked);

//

// Form1

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(703, 569);

this.Controls.Add(this.splitContainer2);

this.Margin = new System.Windows.Forms.Padding(2);

this.Name = "Form1";

this.Text = "Form1";

this.Load += new System.EventHandler(this.Form1\_Load);

this.splitContainer1.Panel1.ResumeLayout(false);

this.splitContainer1.Panel1.PerformLayout();

this.splitContainer1.Panel2.ResumeLayout(false);

this.splitContainer1.Panel2.PerformLayout();

((System.ComponentModel.ISupportInitialize)(this.splitContainer1)).EndInit();

this.splitContainer1.ResumeLayout(false);

this.splitContainer2.Panel1.ResumeLayout(false);

this.splitContainer2.Panel2.ResumeLayout(false);

((System.ComponentModel.ISupportInitialize)(this.splitContainer2)).EndInit();

this.splitContainer2.ResumeLayout(false);

this.tableLayoutPanel1.ResumeLayout(false);

this.tableLayoutPanel1.PerformLayout();

this.splitContainer3.Panel1.ResumeLayout(false);

this.splitContainer3.Panel1.PerformLayout();

this.splitContainer3.Panel2.ResumeLayout(false);

this.splitContainer3.Panel2.PerformLayout();

((System.ComponentModel.ISupportInitialize)(this.splitContainer3)).EndInit();

this.splitContainer3.ResumeLayout(false);

this.ResumeLayout(false);

}

#endregion

private System.Windows.Forms.SplitContainer splitContainer1;

private System.Windows.Forms.TextBox InputText;

private System.Windows.Forms.TextBox OutputText;

private System.Windows.Forms.SplitContainer splitContainer2;

private System.Windows.Forms.TableLayoutPanel tableLayoutPanel1;

private System.Windows.Forms.SplitContainer splitContainer3;

private System.Windows.Forms.Label label1;

private System.Windows.Forms.MaskedTextBox KeyTextBox;

private System.Windows.Forms.Button DecryptButton;

private System.Windows.Forms.Button EncryptButton;

}

}

Файл AES256.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace IS\_LAB3

{

class AES\_256

{

static int nb = 4; // число столбцов State (в AES = 4)

static int nr = 14; // количество раундов (если nb = 4, то nr = 14)

static int nk = 8; // длина ключа в 32-битных словах

static Dictionary<char, int> hex\_symbols\_to\_int = new Dictionary<char, int> {

{'a', 10},

{'b', 11},

{'c', 12},

{'d', 13},

{'e', 14},

{'f', 15}

};

static List<byte> sbox = new List<byte>() {

0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76,

0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0,

0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15,

0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75,

0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84,

0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf,

0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8,

0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2,

0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73,

0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb,

0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79,

0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08,

0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a,

0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e,

0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf,

0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16

};

static List<byte> inv\_sbox = new List<byte>() {

0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38, 0xbf, 0x40, 0xa3, 0x9e, 0x81, 0xf3, 0xd7, 0xfb,

0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87, 0x34, 0x8e, 0x43, 0x44, 0xc4, 0xde, 0xe9, 0xcb,

0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d, 0xee, 0x4c, 0x95, 0x0b, 0x42, 0xfa, 0xc3, 0x4e,

0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2, 0x76, 0x5b, 0xa2, 0x49, 0x6d, 0x8b, 0xd1, 0x25,

0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xd4, 0xa4, 0x5c, 0xcc, 0x5d, 0x65, 0xb6, 0x92,

0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda, 0x5e, 0x15, 0x46, 0x57, 0xa7, 0x8d, 0x9d, 0x84,

0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a, 0xf7, 0xe4, 0x58, 0x05, 0xb8, 0xb3, 0x45, 0x06,

0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02, 0xc1, 0xaf, 0xbd, 0x03, 0x01, 0x13, 0x8a, 0x6b,

0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea, 0x97, 0xf2, 0xcf, 0xce, 0xf0, 0xb4, 0xe6, 0x73,

0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85, 0xe2, 0xf9, 0x37, 0xe8, 0x1c, 0x75, 0xdf, 0x6e,

0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89, 0x6f, 0xb7, 0x62, 0x0e, 0xaa, 0x18, 0xbe, 0x1b,

0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20, 0x9a, 0xdb, 0xc0, 0xfe, 0x78, 0xcd, 0x5a, 0xf4,

0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31, 0xb1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xec, 0x5f,

0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d, 0x2d, 0xe5, 0x7a, 0x9f, 0x93, 0xc9, 0x9c, 0xef,

0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0, 0xc8, 0xeb, 0xbb, 0x3c, 0x83, 0x53, 0x99, 0x61,

0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26, 0xe1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0c, 0x7d

};

static List<List<byte>> rcon = new List<List<byte>>()

{

new List<byte> {0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36},

new List<byte> {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},

new List<byte> {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},

new List<byte> {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}

};

public static List<byte> encrypt(List<byte> input\_bytes, string key)

{

List<List<byte>> state = new List<List<byte>>();

for (int i = 0; i < 4; i++)

{

state.Add(new List<byte>());

}

for (int r = 0; r < 4; r++)

{

for (int c = 0; c < nb; c++)

{

state[r].Add(input\_bytes[r + 4 \* c]);

}

}

List<List<byte>> key\_schedule = key\_expansion(key);

state = add\_round\_key(state, key\_schedule);

for (int rnd = 1; rnd < nr; rnd++)

{

state = sub\_bytes(state);

state = shift\_rows(state);

state = mix\_columns(state);

state = add\_round\_key(state, key\_schedule, rnd);

}

state = sub\_bytes(state);

state = shift\_rows(state);

state = add\_round\_key(state, key\_schedule, nr);

List<byte> output = Enumerable.Repeat<byte>(0x00, 4 \* nb).ToList();

for (int r = 0; r < 4; r++)

{

for (int c = 0; c < nb; c++)

{

output[r + 4 \* c] = state[r][c];

}

}

return output;

}

public static List<byte> decrypt(List<byte> cipher, string key)

{

List<List<byte>> state = new List<List<byte>>();

for (int i = 0; i < 4; i++)

{

state.Add(new List<byte>());

}

for (int r = 0; r < 4; r++)

{

for (int c = 0; c < nb; c++)

{

state[r].Add(cipher[r + 4 \* c]);

}

}

List<List<byte>> key\_schedule = key\_expansion(key);

state = add\_round\_key(state, key\_schedule, nr);

int rnd = nr - 1;

while (rnd >= 1)

{

state = shift\_rows(state, inv: true);

state = sub\_bytes(state, inv: true);

state = add\_round\_key(state, key\_schedule, rnd);

state = mix\_columns(state, inv: true);

rnd -= 1;

}

state = shift\_rows(state, inv: true);

state = sub\_bytes(state, inv: true);

state = add\_round\_key(state, key\_schedule, rnd);

List<byte> output = Enumerable.Repeat<byte>(0x00, 4 \* nb).ToList();

for (int r = 0; r < 4; r++)

{

for (int c = 0; c < nb; c++)

{

output[r + 4 \* c] = state[r][c];

}

}

return output;

}

static List<List<byte>> key\_expansion(string key)

{

List<byte> key\_symbols = Encoding.ASCII.GetBytes(key).ToList();

int key\_length = key\_symbols.Count();

if (key\_length < 4 \* nk)

{

for (int i = 0; i < 4 \* nk - key\_length ; i++)

{

key\_symbols.Add(0x01);

}

}

List<List<byte>> key\_schedule = new List<List<byte>>() {

new List<byte>(),

new List<byte>(),

new List<byte>(),

new List<byte>()

};

for (int r = 0; r < 4; r++)

{

for (int c = 0; c < nk; c++)

{

key\_schedule[r].Add(key\_symbols[r + 4 \* c]);

}

}

for (int col = nk; col < nb \* (nr + 1); col++)

{

List<byte> tmp = new List<byte>();

if (col % nk == 0)

{

for (int row = 1; row < 4; row++)

{

tmp.Add(key\_schedule[row][col - 1]);

}

tmp.Add(key\_schedule[0][col - 1]);

for (int j = 0; j < tmp.Count(); j++)

{

int sbox\_row = tmp[j] / 0x10;

int sbox\_col = tmp[j] % 0x10;

byte sbox\_elem = sbox[16 \* sbox\_row + sbox\_col];

tmp[j] = sbox\_elem;

}

for (int row = 0; row < 4; row++)

{

byte s = (byte)((key\_schedule[row][col - 4]) ^ (tmp[row]) ^ (rcon[row][col / nk - 1]));

key\_schedule[row].Add(s);

}

}

else

{

for (int row = 0; row < 4; row++)

{

byte s = (byte)((key\_schedule[row][col - 4]) ^ key\_schedule[row][col - 1]);

key\_schedule[row].Add(s);

}

}

}

return key\_schedule;

}

static List<List<byte>> add\_round\_key(List<List<byte>> state, List<List<byte>> key\_schedule, int round = 0)

{

for (int col = 0; col < nb; col++)

{

byte s0 = (byte)(state[0][col] ^ key\_schedule[0][nb \* round + col]);

byte s1 = (byte)(state[1][col] ^ key\_schedule[1][nb \* round + col]);

byte s2 = (byte)(state[2][col] ^ key\_schedule[2][nb \* round + col]);

byte s3 = (byte)(state[3][col] ^ key\_schedule[3][nb \* round + col]);

state[0][col] = s0;

state[1][col] = s1;

state[2][col] = s2;

state[3][col] = s3;

}

return state;

}

static List<List<byte>> sub\_bytes(List<List<byte>> state, bool inv = false)

{

List<byte> box;

if (inv == false) // шифрует, иначе дешифрует

box = sbox;

else

box = inv\_sbox;

for (int i = 0; i < state.Count(); i++)

{

for (int j = 0; j < state[i].Count(); j++)

{

int row = state[i][j] / 0x10;

int col = state[i][j] % 0x10;

byte box\_elem = box[16 \* row + col];

state[i][j] = box\_elem;

}

}

return state;

}

static List<List<byte>> shift\_rows(List<List<byte>> state, bool inv = false)

{

int count = 1;

if (inv == false) // шифрует, иначе дешифрует

{

for (int i = 1; i < nb; i++)

{

state[i] = left\_shift(state[i], count);

count++;

}

}

else

{

for (int i = 1; i < nb; i++)

{

state[i] = right\_shift(state[i], count);

count++;

}

}

return state;

}

static List<List<byte>> mix\_columns(List<List<byte>> state, bool inv = false)

{

byte s0, s1, s2, s3;

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

{

if (inv == false) // шифрует, иначе дешифрует

{

s0 = (byte)(mul\_by\_02(state[0][i]) ^ mul\_by\_03(state[1][i]) ^ state[2][i] ^ state[3][i]);

s1 = (byte)( state[0][i] ^ mul\_by\_02(state[1][i]) ^ mul\_by\_03(state[2][i]) ^ state[3][i]);

s2 = (byte)( state[0][i] ^ state[1][i] ^ mul\_by\_02(state[2][i]) ^ mul\_by\_03(state[3][i]));

s3 = (byte)(mul\_by\_03(state[0][i]) ^ state[1][i] ^ state[2][i] ^ mul\_by\_02(state[3][i]));

}

else

{

s0 = (byte)(mul\_by\_0e(state[0][i]) ^ mul\_by\_0b(state[1][i]) ^ mul\_by\_0d(state[2][i]) ^ mul\_by\_09(state[3][i]));

s1 = (byte)(mul\_by\_09(state[0][i]) ^ mul\_by\_0e(state[1][i]) ^ mul\_by\_0b(state[2][i]) ^ mul\_by\_0d(state[3][i]));

s2 = (byte)(mul\_by\_0d(state[0][i]) ^ mul\_by\_09(state[1][i]) ^ mul\_by\_0e(state[2][i]) ^ mul\_by\_0b(state[3][i]));

s3 = (byte)(mul\_by\_0b(state[0][i]) ^ mul\_by\_0d(state[1][i]) ^ mul\_by\_09(state[2][i]) ^ mul\_by\_0e(state[3][i]));

}

state[0][i] = s0;

state[1][i] = s1;

state[2][i] = s2;

state[3][i] = s3;

}

return state;

}

static List<T> left\_shift<T>(List<T> list, int count)

{

List<T> res = list;

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

{

List<T> tmp;

tmp = res.GetRange(1, res.Count() - 1);

tmp.Add(res[0]);

res = tmp;

}

return res;

}

static List<T> right\_shift<T>(List<T> list, int count)

{

List<T> res = list;

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

{

List<T> tmp = new List<T>();

tmp.Add(res.Last());

tmp.AddRange(res.GetRange(0, res.Count() - 1));

res = tmp;

}

return res;

}

static byte mul\_by\_02(byte num)

{

byte res;

if (num < 0x80)

{

res = (byte)(num << 1);

}

else

{

res = (byte)((num << 1) ^ 0x1b);

}

return (byte)(res % 0x100);

}

static byte mul\_by\_03(byte num)

{

return (byte)(mul\_by\_02(num) ^ num);

}

static byte mul\_by\_09(byte num)

{

return (byte)(mul\_by\_02(mul\_by\_02(mul\_by\_02(num))) ^ num);

}

static byte mul\_by\_0b(byte num)

{

return (byte)(mul\_by\_02(mul\_by\_02(mul\_by\_02(num))) ^ mul\_by\_02(num) ^ num);

}

static byte mul\_by\_0d(byte num)

{

return (byte)(mul\_by\_02(mul\_by\_02(mul\_by\_02(num))) ^ mul\_by\_02(mul\_by\_02(num)) ^ num);

}

static byte mul\_by\_0e(byte num)

{

return (byte)(mul\_by\_02(mul\_by\_02(mul\_by\_02(num))) ^ mul\_by\_02(mul\_by\_02(num)) ^ mul\_by\_02(num));

}

}

}

**3. Контрольные примеры работы программы**

Примеры работы представлены на рисунках 1 – 10.

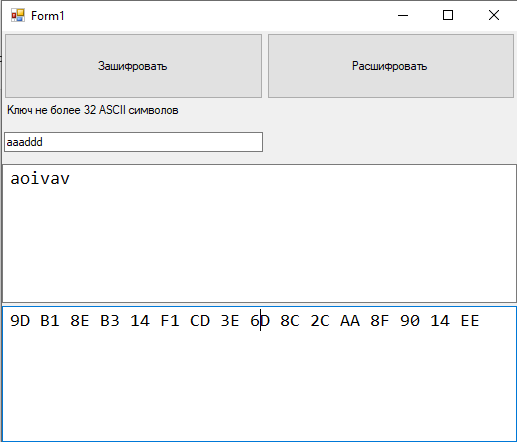


Рисунок 1 – Зашифровка строки “aoivav” с ключом “aaaddd”

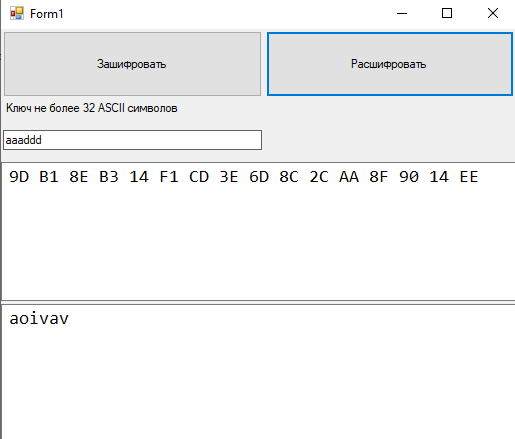


Рисунок 2 – Расшифровка строки “9D B1 8E B3 14 F1 CD 3E 6D 8C 2C AA 8F 90 14 EE” с ключом “aaaddd”

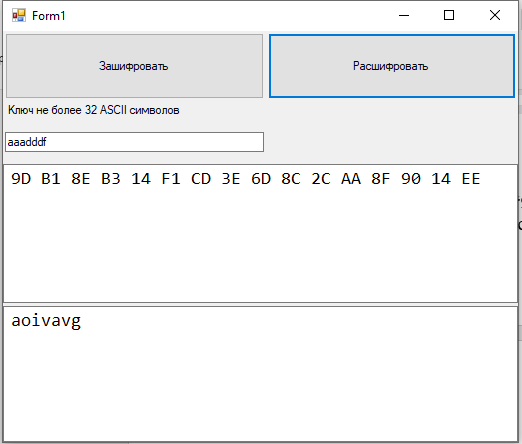


Рисунок 3 – Расшифровка строки “9D B1 8E B3 14 F1 CD 3E 6D 8C 2C AA 8F 90 14 EE” с ключом “aaadddf”

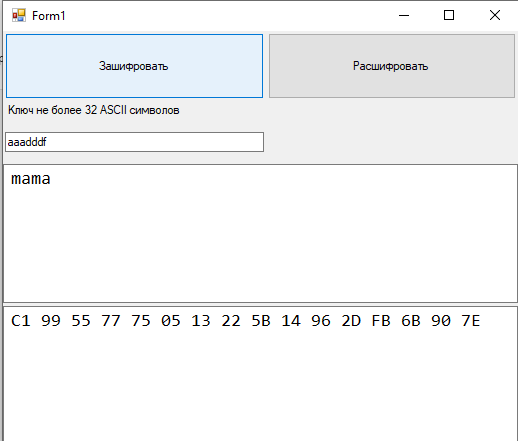


Рисунок 4 – Зашифровка строки “mama” с ключом “aaadddf”

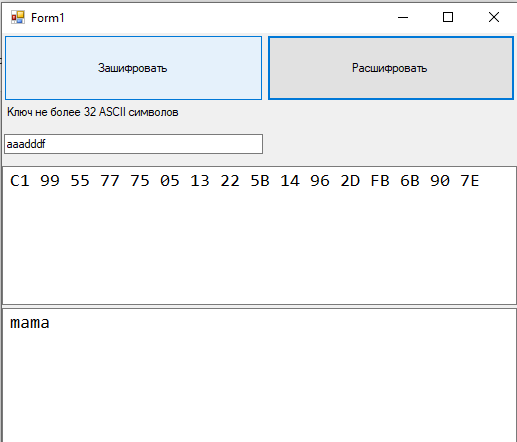


Рисунок 5 - Расшифровка строки “ C1 99 55 77 75 05 13 22 5B 14 96 2D FB 6B 90 7E” с ключом “aaadddf”

ЗАКЛЮЧЕНИЕ

В процессе выполнения данной лабораторной работы я познакомился с алгоритмом шифрования AES, а также научился его использовать для составления программ шифраторов-дешифраторов.