МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ СИСТЕМ И ТЕХНОЛОГИЙ

Кафедра «Информационные системы»

Дисциплина «Информационная безопасность»

Лабораторная работа № 2

Выполнил:

студент гр. ПИбд -41

Калядин М. С.

Проверил:

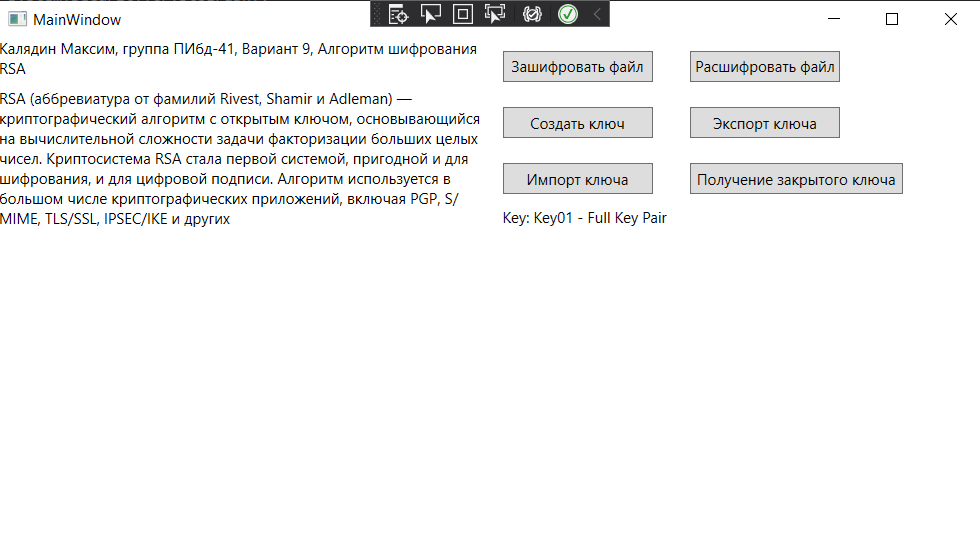
Мытарев П. В.

Ульяновск, 2021г.

Программа написана на языке C# на системе для построения клиентских приложений WPF.

Ссылка на репозиторий - <https://github.com/MaximKalyadin/IB>

Скриншоты экранов



Слева на экране представлена необходимая информация об авторе и об метое шифрования. Справа кнопки позволяющие манипулировать файлами.

Листинг программы

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace IB\_2

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

// Объявление CspParmeters и RsaCryptoServiceProvider

// объекты с глобальной областью видимости вашего класса Form.

private readonly CspParameters cspp = new CspParameters();

private RSACryptoServiceProvider rsa;

// Переменные пути для источника, шифрования и

// расшифровка папок. Должен заканчиваться обратной косой чертой.

const string EncrFolder = @"C:\Users\Maxim\Desktop\";

const string DecrFolder = @"C:\Users\Maxim\Desktop\";

const string SrcFolder = @"C:\Users\Maxim\Desktop\";

// Файл открытого ключа

const string PubKeyFile = @"C:\Users\Maxim\Desktop\rsaPublicKey.txt";

// Имя ключевого контейнера для

// пара значений закрытого / открытого ключа.

const string keyName = "Key01";

private readonly OpenFileDialog openFileDialog1 = new OpenFileDialog();

private readonly OpenFileDialog openFileDialog2 = new OpenFileDialog();

public MainWindow()

{

InitializeComponent();

}

/// <summary>

/// Событие шифрования файла

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Button\_Click(object sender, RoutedEventArgs e)

{

if (rsa == null)

{

MessageBox.Show("Key not set.");

}

else

{

// Отображение диалогового окна для выбора файла для шифрования.

openFileDialog1.InitialDirectory = SrcFolder;

if (openFileDialog1.ShowDialog() == true)

{

string fName = openFileDialog1.FileName;

if (fName != null)

{

FileInfo fInfo = new FileInfo(fName);

// Передаем имя файла без пути.

string name = fInfo.FullName;

EncryptFile(name);

}

}

}

}

/// <summary>

/// Событие расшифрования файла

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

if (rsa == null)

{

MessageBox.Show("Key not set.");

}

else

{

// Отображение диалогового окна для выбора зашифрованного файла.

openFileDialog2.InitialDirectory = EncrFolder;

if (openFileDialog2.ShowDialog() == true)

{

string fName = openFileDialog2.FileName;

if (fName != null)

{

FileInfo fi = new FileInfo(fName);

string name = fi.Name;

DecryptFile(name);

}

}

}

}

/// <summary>

/// Событие создания файла

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Button\_Click\_2(object sender, RoutedEventArgs e)

{

// Сохраняет пару ключей в контейнере ключей.

cspp.KeyContainerName = keyName;

rsa = new RSACryptoServiceProvider(cspp)

{

PersistKeyInCsp = true

};

label1.Text = rsa.PublicOnly ? "Key: " + cspp.KeyContainerName + " - Public Only" : "Key: " + cspp.KeyContainerName + " - Full Key Pair";

}

/// <summary>

/// Метод шифрования файла

/// </summary>

/// <param name="inFile">путь файла куда будем сохранять шифрованные данные</param>

private void EncryptFile(string inFile)

{

// Создаем экземпляр Aes для

// симметричное шифрование данных.

Aes aes = Aes.Create();

ICryptoTransform transform = aes.CreateEncryptor();

// Используйте RSACryptoServiceProvider для

// зашифровать ключ AES.

// rsa создается ранее:

// rsa = новый RSACryptoServiceProvider (cspp);

byte[] keyEncrypted = rsa.Encrypt(aes.Key, false);

// Создаем байтовые массивы для хранения

// значения длины ключа и IV.

byte[] LenK = new byte[4];

byte[] LenIV = new byte[4];

int lKey = keyEncrypted.Length;

LenK = BitConverter.GetBytes(lKey);

int lIV = aes.IV.Length;

LenIV = BitConverter.GetBytes(lIV);

// Записываем в FileStream следующее

// для зашифрованного файла (outFs):

// - длина ключа

// - длина IV

// - зашифрованный ключ

// - IV

// - зашифрованное содержимое шифра

int startFileName = inFile.LastIndexOf("\\") + 1;

// Измените расширение файла на ".enc"

string outFile = EncrFolder + inFile.Substring(startFileName, inFile.LastIndexOf(".") - startFileName) + ".enc";

using (FileStream outFs = new FileStream(outFile, FileMode.Create))

{

outFs.Write(LenK, 0, 4);

outFs.Write(LenIV, 0, 4);

outFs.Write(keyEncrypted, 0, lKey);

outFs.Write(aes.IV, 0, lIV);

// Теперь напишем зашифрованный текст, используя

// CryptoStream для шифрования.

using (CryptoStream outStreamEncrypted = new CryptoStream(outFs, transform, CryptoStreamMode.Write))

{

// Шифрованием чанка в

// время можно сэкономить память

// и разместить большие файлы.

int count = 0;

int offset = 0;

// blockSizeBytes может быть любого произвольного размера.

int blockSizeBytes = aes.BlockSize / 8;

byte[] data = new byte[blockSizeBytes];

int bytesRead = 0;

using (FileStream inFs = new FileStream(inFile, FileMode.Open))

{

do

{

count = inFs.Read(data, 0, blockSizeBytes);

offset += count;

outStreamEncrypted.Write(data, 0, count);

bytesRead += blockSizeBytes;

}

while (count > 0);

inFs.Close();

}

outStreamEncrypted.FlushFinalBlock();

outStreamEncrypted.Close();

}

outFs.Close();

}

}

/// <summary>

/// Метод расшифрования файла

/// </summary>

/// <param name="inFile">путь файла для расшифровки</param>

private void DecryptFile(string inFile)

{

// Создание экземпляра Aes для

// симметричного дешифрования данных.

Aes aes = Aes.Create();

// Создаем байтовые массивы, чтобы получить длину

// зашифрованный ключ и IV.

// Эти значения были сохранены как 4 байта каждое

// в начале зашифрованного пакета.

byte[] LenK = new byte[4];

byte[] LenIV = new byte[4];

// Construct the file name for the decrypted file.

string outFile = DecrFolder + inFile.Substring(0, inFile.LastIndexOf(".")) + ".txt";

// Используем объекты FileStream для чтения зашифрованного

// файл (inFs) и сохраняем расшифрованный файл (outFs).

using (FileStream inFs = new FileStream(EncrFolder + inFile, FileMode.Open))

{

inFs.Seek(0, SeekOrigin.Begin);

inFs.Seek(0, SeekOrigin.Begin);

inFs.Read(LenK, 0, 3);

inFs.Seek(4, SeekOrigin.Begin);

inFs.Read(LenIV, 0, 3);

// Преобразуем длины в целые числа.

int lenK = BitConverter.ToInt32(LenK, 0);

int lenIV = BitConverter.ToInt32(LenIV, 0);

// Определение начальной позиции

// зашифрованного текста (крахмал)

// и его длины (lenS).

int startC = lenK + lenIV + 8;

int lenC = (int)inFs.Length - startC;

// Создаем байтовые массивы для

// зашифрованный ключ Aes,

// IV и зашифрованный текст.

byte[] KeyEncrypted = new byte[lenK];

byte[] IV = new byte[lenIV];

// Извлекаем ключ и IV

// начиная с индекса 8

// после значений длины.

inFs.Seek(8, SeekOrigin.Begin);

inFs.Read(KeyEncrypted, 0, lenK);

inFs.Seek(8 + lenK, SeekOrigin.Begin);

inFs.Read(IV, 0, lenIV);

Directory.CreateDirectory(DecrFolder);

// Используйте RSACryptoServiceProvider

// для расшифровки ключа AES.

byte[] KeyDecrypted = rsa.Decrypt(KeyEncrypted, false);

// Расшифровать ключ.

ICryptoTransform transform = aes.CreateDecryptor(KeyDecrypted, IV);

// Расшифровываем зашифрованный текст из

// из FileSteam зашифрованного

// файл (inFs) в FileStream

// для расшифрованного файла (outFs).

using (FileStream outFs = new FileStream(outFile, FileMode.Create))

{

int count = 0;

int offset = 0;

// blockSizeBytes может быть любого произвольного размера.

int blockSizeBytes = aes.BlockSize / 8;

byte[] data = new byte[blockSizeBytes];

// Расшифровывая кусок за раз,

// можно сэкономить память и

// вмещаем большие файлы.

// Начать с начала

// зашифрованного текста.

inFs.Seek(startC, SeekOrigin.Begin);

using (CryptoStream outStreamDecrypted = new CryptoStream(outFs, transform, CryptoStreamMode.Write))

{

do

{

count = inFs.Read(data, 0, blockSizeBytes);

offset += count;

outStreamDecrypted.Write(data, 0, count);

}

while (count > 0);

outStreamDecrypted.FlushFinalBlock();

outStreamDecrypted.Close();

}

outFs.Close();

}

inFs.Close();

}

}

/// <summary>

/// Событие экспорта ключа для расшифрования и шифрования файла

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Button\_Click\_3(object sender, RoutedEventArgs e)

{

// Сохраняем открытый ключ, созданный RSA

// в файл. Осторожно, сохраняя

// ключ к файлу представляет угрозу безопасности.

Directory.CreateDirectory(EncrFolder);

StreamWriter sw = new StreamWriter(PubKeyFile, false);

sw.Write(rsa.ToXmlString(false));

sw.Close();

}

/// <summary>

/// Событие импорта ключа для расшифрования и шифрования файла

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Button\_Click\_4(object sender, RoutedEventArgs e)

{

StreamReader sr = new StreamReader(PubKeyFile);

cspp.KeyContainerName = keyName;

rsa = new RSACryptoServiceProvider(cspp);

string keytxt = sr.ReadToEnd();

rsa.FromXmlString(keytxt);

rsa.PersistKeyInCsp = true;

label1.Text = rsa.PublicOnly ? "Key: " + cspp.KeyContainerName + " - Public Only" : "Key: " + cspp.KeyContainerName + " - Full Key Pair";

sr.Close();

}

/// <summary>

/// Событие получение закрытого ключа для расшифрования и шифрования файла

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Button\_Click\_5(object sender, RoutedEventArgs e)

{

cspp.KeyContainerName = keyName;

rsa = new RSACryptoServiceProvider(cspp)

{

PersistKeyInCsp = true

};

label1.Text = rsa.PublicOnly ? "Key: " + cspp.KeyContainerName + " - Public Only" : "Key: " + cspp.KeyContainerName + " - Full Key Pair";

}

}

}