|  |  |  |
| --- | --- | --- |
| Министерство науки и высшего образования  Российской Федерации | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования | | |
| «Новосибирский государственный технический университет» | | |
|  | | |
| Прикладной математики | | |
|  | | |
| Практическая работа № 4 | | |
| по дисциплине «Основы криптографии» | | |
|  | | |
| **Симметричные криптографические алгоритмы** | | |
|  | | |
|  | Факультет: | ПМИ |
| Группа: | ПМ-81 |
| Вариант: | 1 |
| Студент: | Ефремов Артур |
| Преподаватель: | Ступаков Илья Михайлович |
|
|  |  |
| Новосибирск | | |
| 2020 | | |

1. **Цель работы**  
   Научится использовать готовые криптографические примитивы для шифрования данных.
2. **Задание**  
   **Обязательная часть (15 баллов)**

Сделать программу которая шифрует и дешифрует некоторый файл с помощью алгоритма AES. В качестве ключа использовать хеш (с возможностью выбора алгоритма) от вводимого пользователем пароля. Сам ключ в итоге нигде сохраняться не должен. Использовать режим CBC и в качестве IV взять ключ (такой подход считается плохим, подумайте почему).

Сделать селфи бригады (можно по отдельности и составить коллаж), зашифровать и выложить на общий диск. Изображение и пароль добавить в отчет.

**Доп. Задания**

**(5 баллов)** Сделать чтобы шифровалась только часть файла отвечающая за данные, т.е. после шифрования файл должен остаться валидным изображением и корректно открываться. Привести зашифрованное изображение в отчете. Зашифровать и привести изображение используя режим ECB.

1. **Текст программы**

using System;

using System.Linq;

using System.IO;

using System.Security.Cryptography;

using System.Text;

using System.Drawing;

using System.Collections.Generic;

namespace CryptLab4

{

class Program

{

static public string path = "../../../files/";

static void Main(string[] args)

{

bool repeat = true;

while (repeat)

{

Console.WriteLine("Выберете опцию:");

Console.WriteLine("<1> Зашифровать файл");

Console.WriteLine("<2> Дешифровать файл");

Console.WriteLine("<3> Зашифровать только пиксели файла");

Console.WriteLine("<0> Выйти(");

string choice = Console.ReadLine();

switch (choice)

{

case "1":

{

Console.WriteLine("Введите название файла для шифрования:");

string inFileName = Console.ReadLine();

byte[] ToEncrypt;

try

{

ToEncrypt = File.ReadAllBytes(path + inFileName);

}

catch (Exception)

{

Console.WriteLine("Нет такого файла!");

break;

}

Console.WriteLine("Введите название файла для вывода:");

string outFileName = Console.ReadLine();

Console.WriteLine("Введите пароль для шифрования:");

byte[] Password = Encoding.UTF8.GetBytes(Console.ReadLine());

var hashAlg = GetHashAlgorithm();

File.WriteAllBytes(path + outFileName, Encrypt(ToEncrypt, Password, hashAlg, CipherMode.CBC));

Console.WriteLine("Шифрование прошло успешно!");

break;

}

case "2":

{

Console.WriteLine("Введите название файла для дешифровки:");

string inFileName = Console.ReadLine();

byte[] ToDecrypt;

try

{

ToDecrypt = File.ReadAllBytes(path + inFileName);

}

catch (Exception)

{

Console.WriteLine("Нет такого файла!");

break;

}

Console.WriteLine("Введите название файла для вывода:");

string outFileName = Console.ReadLine();

Console.WriteLine("Введите пароль для дешифровки:");

byte[] Password = Encoding.UTF8.GetBytes(Console.ReadLine());

var hashAlg = GetHashAlgorithm();

try

{

File.WriteAllBytes(path + outFileName, Decrypt(ToDecrypt, Password, hashAlg, CipherMode.CBC));

}

catch(Exception)

{

Console.WriteLine("Неверный пароль или алгоритм генерации хеша!");

break;

}

Console.WriteLine("Дешифровка прошла успешно!");

break;

}

case "3":

{

Console.WriteLine("Введите название файла для шифрования:");

string inFileName = Console.ReadLine();

Bitmap img;

try

{

img = new Bitmap(path + inFileName);

}

catch (Exception)

{

Console.WriteLine("Нет такого файла!");

break;

}

int w = img.Width;

int h = img.Height;

byte[] toEncrypt = new byte[h \* w \* 4];

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

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

{

Color pixel = img.GetPixel(j, i);

toEncrypt[i \* w \* 4 + j \* 4 + 0] = pixel.A;

toEncrypt[i \* w \* 4 + j \* 4 + 1] = pixel.R;

toEncrypt[i \* w \* 4 + j \* 4 + 2] = pixel.G;

toEncrypt[i \* w \* 4 + j \* 4 + 3] = pixel.B;

}

Console.WriteLine("Введите название файла для вывода:");

string outFileName = Console.ReadLine();

Console.WriteLine("Введите пароль для шифрования:");

byte[] Password = Encoding.UTF8.GetBytes(Console.ReadLine());

var hashAlg = GetHashAlgorithm();

byte[] encData = Encrypt(toEncrypt.ToArray(), Password, hashAlg, CipherMode.ECB);

Bitmap outImage = new Bitmap(w, h, System.Drawing.Imaging.PixelFormat.Format32bppArgb);

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

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

{

byte A = encData[i \* w \* 4 + j \* 4 + 0];

byte R = encData[i \* w \* 4 + j \* 4 + 1];

byte G = encData[i \* w \* 4 + j \* 4 + 2];

byte B = encData[i \* w \* 4 + j \* 4 + 3];

outImage.SetPixel(j, i, Color.FromArgb(A, R, G, B));

}

outImage.Save(path + outFileName);

Console.WriteLine("Шифрование прошло успешно!");

break;

}

case "0":

{

repeat = false;

break;

}

default:

{

Console.WriteLine("Неверный выбор!");

break;

}

}

}

}

public static HashAlgorithm GetHashAlgorithm()

{

bool repeat = true;

HashAlgorithm res = HashAlgorithm.Create("SHA-256");

while (repeat)

{

repeat = false;

Console.WriteLine("Выберете алгоритм генерации хеша:");

Console.WriteLine("<1> SHA-256");

Console.WriteLine("<2> MD5");

Console.WriteLine("<3> SHA1");

string hashChoice = Console.ReadLine();

switch (hashChoice)

{

case "1":

{

res = HashAlgorithm.Create("SHA-256");

break;

}

case "2":

{

res = HashAlgorithm.Create("MD5");

break;

}

case "3":

{

res = HashAlgorithm.Create("SHA1");

break;

}

default:

{

repeat = true;

Console.WriteLine("Неверный выбор!");

break;

}

}

}

return res;

}

public static byte[] Encrypt(byte[] dataToEncrypt, byte[] password, HashAlgorithm hashAlg, CipherMode cipherMode)

{

using (var aes = Aes.Create())

{

aes.Mode = cipherMode;

aes.Key = hashAlg.ComputeHash(password);

aes.IV = aes.Key.ToList().Take(16).ToArray();

aes.Padding = PaddingMode.PKCS7;

return PerformCrypt(aes.CreateEncryptor(), dataToEncrypt);

}

}

public static byte[] Decrypt(byte[] dataToDecrypt, byte[] password, HashAlgorithm hashAlg, CipherMode cipherMode)

{

using (var aes = Aes.Create())

{

aes.Mode = cipherMode;

aes.Key = hashAlg.ComputeHash(password);

aes.IV = aes.Key.ToList().Take(16).ToArray();

aes.Padding = PaddingMode.PKCS7;

var decryptor = aes.CreateDecryptor();

return PerformCrypt(aes.CreateDecryptor(), dataToDecrypt);

}

}

public static byte[] PerformCrypt(ICryptoTransform transform, byte[] data)

{

using (var msDecrypt = new MemoryStream())

{

using (var csEncrypt = new CryptoStream(msDecrypt, transform, CryptoStreamMode.Write))

{

csEncrypt.Write(data, 0, data.Length);

csEncrypt.FlushFinalBlock();

return msDecrypt.ToArray();

}

}

}

}

}

1. **Тестирование**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Тест** | **Назначение** | **Результат** |
| **1** | Выберете опцию:  <1> Зашифровать файл  <2> Дешифровать файл  <3> Зашифровать только пиксели файла  <0> Выйти(  1  Введите название файла для шифрования:  Arthur.png  Введите название файла для вывода:  Arthur\_encrypted1.png  Введите пароль для шифрования:  crypt4  Выберете алгоритм генерации хэша:  <1> SHA-256  <2> MD5  <3> SHA1  1  Шифрование прошло успешно! | Шифрование файла целиком. | Файл открывается, но изображение не отображается. |
| **2** | Выберете опцию:  <1> Зашифровать файл  <2> Дешифровать файл  <3> Зашифровать только пиксели файла  <0> Выйти(  2  Введите название файла для дешифровки:  Arthur\_encrypted1.png  Введите название файла для вывода:  Arthur\_decrypted1.png  Введите пароль для дешифровки:  crypt4  Выберете алгоритм генерации хэша:  <1> SHA-256  <2> MD5  <3> SHA1  1  Дешифровка прошла успешно! | Дешифровка файла правильным паролем. | C:\Users\Артур\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Arthur_decrypted1.png  Все нормально. |
| **3** | Выберете опцию:  <1> Зашифровать файл  <2> Дешифровать файл  <3> Зашифровать только пиксели файла  <0> Выйти(  2  Введите название файла для дешифровки:  Arthur\_encrypted1.png  Введите название файла для вывода:  Arthur\_decrypted2.png  Введите пароль для дешифровки:  random\_password  Выберете алгоритм генерации хэша:  <1> SHA-256  <2> MD5  <3> SHA1  1  Неверный пароль или алгоритм генерации хеша! | Дешифровка файла неправильным паролем. | Получаем соответствующее сообщение. |
| **4** | Выберете опцию:  <1> Зашифровать файл  <2> Дешифровать файл  <3> Зашифровать только пиксели файла  <0> Выйти(  2  Введите название файла для дешифровки:  Arthur\_encrypted1.png  Введите название файла для вывода:  Arthur\_decrypted3.png  Введите пароль для дешифровки:  crypt4  Выберете алгоритм генерации хэша:  <1> SHA-256  <2> MD5  <3> SHA1  2  Неверный пароль или алгоритм генерации хеша! | Дешифровка файла неправильным алгоритмом генерации хеша. | Получаем соответствующее сообщение. |
| **5** | Выберете опцию:  <1> Зашифровать файл  <2> Дешифровать файл  <3> Зашифровать только пиксели файла  <0> Выйти(  3  Введите название файла для шифрования:  Arthur.png  Введите название файла для вывода:  Arthur\_encrypted2.png  Введите пароль для шифрования:  crypt4  Выберете алгоритм генерации хеша:  <1> SHA-256  <2> MD5  <3> SHA1  1  Шифрование прошло успешно! | Шифрование только пикселей изображения. | **C:\Users\Артур\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Arthur_encrypted2.png**  Видим черты исходного изображения, так как использовали режим шифрования ECB. |

1. **Данные**

|  |  |  |
| --- | --- | --- |
| **№** | **Содержание** | **Назначение** |
| **1** |  | Исходное изображение Arthur.png |
| **2** | C:\Users\Артур\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Arthur_encrypted2.png | Изображение, с шифрованием только пикселей. |
| **3** | crypt4 | Пароль |