|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра прикладной математики | | |
| Лабораторная работа № 3 | | |
| по дисциплине «Основы криптографии» | | |
| **Асимметричные криптографические алгоритмы** | | |
|  | | |
|  | Группы ПМ-23 & 21 | Румянцев Артём |
|  | уваров Артём |
|  | чёрный Антон |
|  |  |
|  |  |
|  |  |
| Преподаватель | ступаков илья михайлович |
|  |  |
| Новосибирск, 2024 | | |

1. **Цель работы**

Знакомство с симметричными криптографическими алгоритмами.

1. **Ход работы**

**Обязательная часть** (*15 баллов*)

1. Сделать программу, которая шифрует и дешифрует некоторый текст с помощью алгоритма AES. В качестве ключа использовать хеш (SHA-256) от вводимого пользователем пароля. Сам ключ в итоге нигде сохраняться не должен. Использовать режим CBC+PKCS7.

([текст программы](#текст_программы))

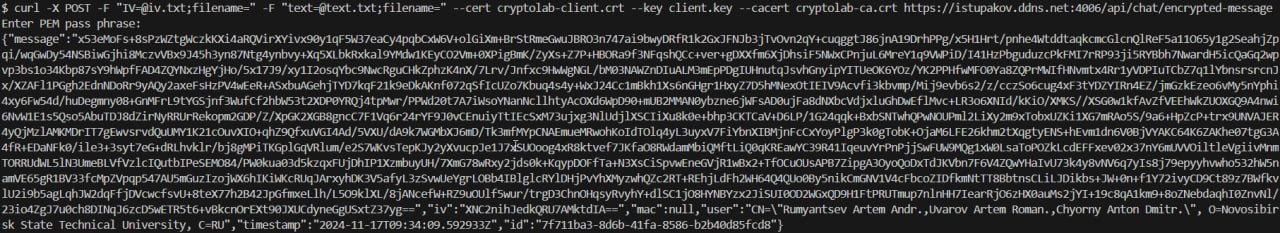
1. Отправить зашифрованное сообщение в чат с помощью multipart POST запроса на адрес https://istupakov.ddns.net:4006/api/chat/encrypted-message, с полями text и iv закодированными в base64. Ссылку на сообщение и пароль добавить в отчет.

Текст:

|  |  |
| --- | --- |
| Звонит заика в милицию и говорит:  - З-З-Здравствуйте. Вы з-з-знаете у нас тут л-л-лошадь дохлая в-в-валяется, воняет н-н-невыносимо, заберите п-п-пожалуйста!  Милиционер спрашивает:  - Улица какая?  Заика:  - Г-Г-Г-Г...  Милиционер:  - Гоголя?  - Н-ННет..  - Горького?  - Н-Н-Нет..  - Вспомнишь перезвонишь. Через пять минут заика звонит снова:  - З-З-Здравствуйте. Вы з-з-знаете у нас тут л-л-лошадь дохлая в-в-валяется, воняет н-н-невыносимо, заберите п-п-пожалуйста!  Милиционер спрашивает:  - Улица какая?  Заика:  - Г-Г-Г... | Милиционер:  - Гоголя?  - Н-ННет...  - Горького?  - Н-Н-Нет...  - Вспомнишь перезвонишь.  Проходит десять минут. Тот же заика звонит опять:  - З-З-Здравствуйте. Вы з-з-знаете у нас тут л-л-лошадь дохлая в-в-валяется, воняет н-н-невыносимо, заберите п-п-пожалуйста!  Милиционер спрашивает:  - Улица какая?  Заика:  - Г-Г-Г...  Милиционер:  - Гоголя?  - Н-ННет...  - Горького?  Заика:  - Д-Д-Да... Я её т-т-туда п-п-перетащил. |

Зашифрованный текст:



IV: XNC2nihJedkQRU7AMktdIA==

Ссылка на сообщение: <https://istupakov.ddns.net:4006/chat/message/7f711ba3-8d6b-41fa-8586-b2b40d85fcd8>

Пароль: pm21

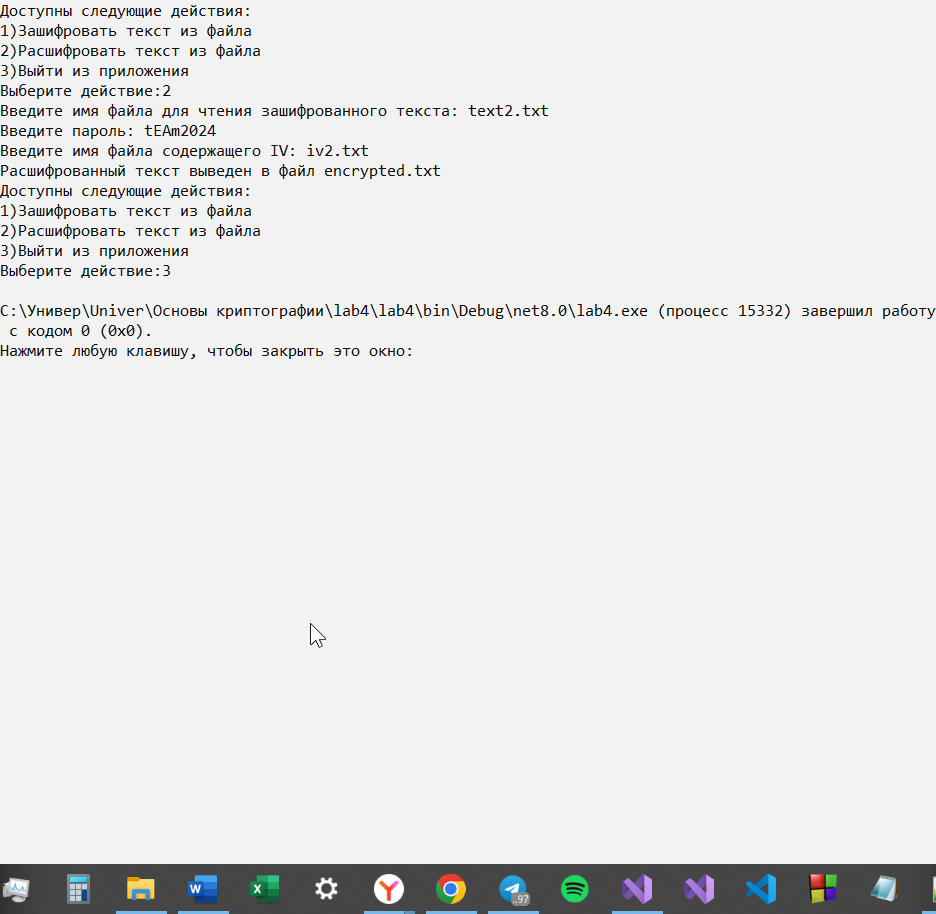
1. Обменяться с другой бригадой ссылками и паролями и расшифровать сообщения друг-друга.

Ссылка, полученная от другой бригады: <https://istupakov.ddns.net:4006/chat/message/55d67258-d390-45af-83bc-345a77273e6a>

Пароль: tEAm2024

Зашифрованный текст:



IV: j3l1cETNfrN71u4CZg/7cg==

Результат расшифрования:

Хоккей это не просто игра, это настоящая страсть, которая объединяет миллионы поклонников по всему миру. Ледяная арена превращается в поле битвы, где скорость, ловкость и стратегия сталкиваются в захватывающем противостоянии. Каждое движение игроков, их стремительное взаимодействие на льду под ярким светом арены, создает атмосферу ожидания. История хоккея уходит корнями в канадские просторы, где азарт и состязательность традиционно занимали центральное место в культуре местных жителей. С развитием игры возникли свои ритуалы, правила и тактики, которые делают хоккей сложным, но в то же время невероятно завораживающим видом спорта.

**Дополнительная часть** (*10 баллов*)

1. Сделать узнаваемое селфи бригады (в полном составе, должно быть видно лица), сохранить в формате PNG (32 бита на пиксель, размер изображения 256x256).
2. Зашифровать часть файла, отвечающую за пиксели (файл должен остаться валидным изображением и корректно открываться) используя AES режимы CBC и ECB (без padding).
3. Вычислить HMAC для зашифрованных файлов используя SHA-256.

Загрузить полученные файлы в чат с помощью multipart POST запроса на https://istupakov.ddns.net:4006/api/chat/encrypted-image, с полями file, iv (для режима CBC) и mac. Ссылки на сообщения и пароли добавить в отчет.

1. **[Текст программы](#ссылка_на_текст_программы)**

using System;

using System.IO;

using System.Reflection.Metadata;

using System.Security.Cryptography;

using System.Text;

using static System.Runtime.InteropServices.JavaScript.JSType;

public class AESCipher

{

public static byte[] ComputeSha256Hash(string rawData)

{

using (SHA256 sha256Hash = SHA256.Create())

{

return sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(rawData));

}

}

public static byte[] EncryptAES(string plainText, byte[] Key, byte[] iv)

{

byte[] encrypted;

using (Aes aesAlg = Aes.Create())

{

aesAlg.Key = Key;

aesAlg.IV = iv;

aesAlg.Mode = CipherMode.CBC;

aesAlg.Padding = PaddingMode.PKCS7;

ICryptoTransform encryptor = aesAlg.CreateEncryptor(aesAlg.Key, aesAlg.IV);

using (MemoryStream msEncrypt = new MemoryStream())

{

using (CryptoStream csEncrypt = new CryptoStream(msEncrypt, encryptor, CryptoStreamMode.Write))

{

using (StreamWriter swEncrypt = new StreamWriter(csEncrypt))

{

swEncrypt.Write(plainText);

}

encrypted = msEncrypt.ToArray();

}

}

}

return encrypted;

}

public static string DecryptAES(byte[] cipherText, byte[] Key, byte[] IV)

{

string plaintext = null;

using (Aes aesAlg = Aes.Create())

{

aesAlg.Key = Key;

aesAlg.IV = IV;

aesAlg.Mode = CipherMode.CBC;

aesAlg.Padding = PaddingMode.PKCS7;

ICryptoTransform decryptor = aesAlg.CreateDecryptor(aesAlg.Key, aesAlg.IV);

using (MemoryStream msDecrypt = new MemoryStream(cipherText))

{

using (CryptoStream csDecrypt = new CryptoStream(msDecrypt, decryptor, CryptoStreamMode.Read))

{

using (StreamReader srDecrypt = new StreamReader(csDecrypt))

{

plaintext = srDecrypt.ReadToEnd();

}

}

}

}

return plaintext;

}

}

public class Program

{

public static void Main()

{

int action = 0;

while (action != 3)

{

Console.Write("Доступны следующие действия:\n" +

"1)Зашифровать текст из файла\n" +

"2)Расшифровать текст из файла\n" +

"3)Выйти из приложения\n" +

"Выберите действие:");

while ((!int.TryParse(Console.ReadLine(), out action)) || action < 1 || action > 3)

Console.WriteLine("В списке действий нет соответствующего пункта. Введите число заново:");

switch (action)

{

case 1:

{

Console.Write("Введите имя файла для чтения текста: ");

string inputFileName = Console.ReadLine();

string original = File.ReadAllText(inputFileName);

Console.Write("Введите пароль: ");

string password = Console.ReadLine();

byte[] key = AESCipher.ComputeSha256Hash(password);

byte[] iv;

using (Aes aes = Aes.Create())

{

aes.GenerateIV();

iv = aes.IV;

}

byte[] encrypted = AESCipher.EncryptAES(original, key, iv);

string encryptedBase64 = Convert.ToBase64String(encrypted);

string ivBase64 = Convert.ToBase64String(iv);

string outputEncryptedFileName = "text.txt";

Console.WriteLine($"Зашифрованный текст выведен в файл {outputEncryptedFileName}");

File.WriteAllText(outputEncryptedFileName, encryptedBase64);

string outputIvFileName = "iv.txt";

Console.WriteLine($"Зашифрованный текст выведен в файл {outputIvFileName}");

File.WriteAllText(outputIvFileName, ivBase64);

break;

}

case 2:

{

Console.Write("Введите имя файла для чтения зашифрованного текста: ");

string inputFileName = Console.ReadLine();

string encryptedStr = File.ReadAllText(inputFileName);

byte[] encrypted = Convert.FromBase64String(encryptedStr);

Console.Write("Введите пароль: ");

string password = Console.ReadLine();

byte[] key = AESCipher.ComputeSha256Hash(password);

Console.Write("Введите имя файла содержащего IV: ");

string ivFileName = Console.ReadLine();

string ivStr = File.ReadAllText(ivFileName);

byte[] iv = Convert.FromBase64String(ivStr);

string decrypted = AESCipher.DecryptAES(encrypted, key, iv);

string outputDecryptedFileName = "encrypted.txt";

Console.WriteLine($"Расшифрованный текст выведен в файл {outputDecryptedFileName}");

File.WriteAllText(outputDecryptedFileName, decrypted);

break;

}

case 3:

{

return;

}

}

}

}

}