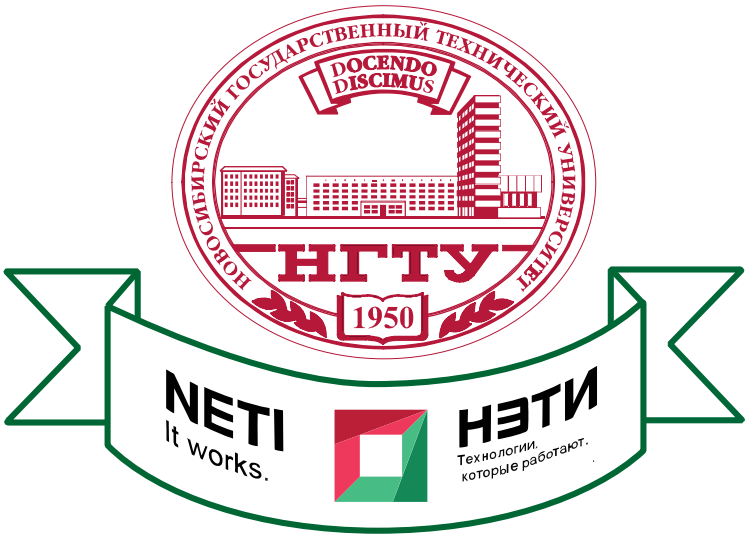
Министерство науки и высшего образования
Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Новосибирский государственный технический университет»

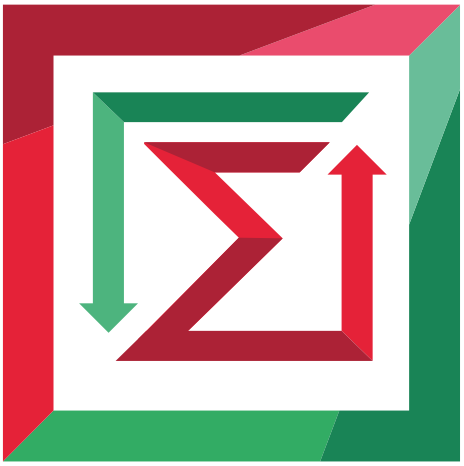


Кафедра теоретической и прикладной информатики

Практическое задание № 4

по дисциплине «Основы криптографии»

.



Факультет:	ПМИ
Группа:	ПМ-71
Бригада:	24
Студенты:	Востриков Вячеслав, Баштовой Павел
Преподаватели:	Ступаков Илья Михайлович

Новосибирск

2019

Цель

Научится использовать готовые криптографические примитивы для шифрования данных.

Ход работы

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).

Текст программы

```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab_4
{
    class Program
    {
        public static byte[] AES_Cipher(byte[] password_hash, byte[] data)
        {
            password_hash = password_hash.Where((item, i) => i < 16).ToArray();
            Aes crypt_AES = Aes.Create();
            crypt_AES.IV = password_hash;
            crypt_AES.Key = password_hash;
            crypt_AES.Mode = CipherMode.CBC;
            crypt_AES.Padding = PaddingMode.Zeros;
            byte[] encrypted;
            using (ICryptoTransform crypt = crypt_AES.CreateEncryptor(crypt_AES.Key,
crypt_AES.IV))
            {
                using (MemoryStream ms = new MemoryStream())
                {
                    using (CryptoStream cs = new CryptoStream(ms, crypt,
CryptoStreamMode.Write))
                    {
                        cs.Write(data, 0, data.Length);
                    }

                    encrypted = ms.ToArray();
                }
            }

            return encrypted;
        }

        public static byte[] AES_Decipher(byte[] password_hash2, byte[] cip_data)
        {
            password_hash2 = password_hash2.Where((item, i) => i < 16).ToArray();
            var crypt_AES = Aes.Create();
            crypt_AES.Key = password_hash2;
            crypt_AES.IV = password_hash2;
            crypt_AES.Mode = CipherMode.CBC;
            crypt_AES.Padding = PaddingMode.Zeros;
            byte[] data;
```

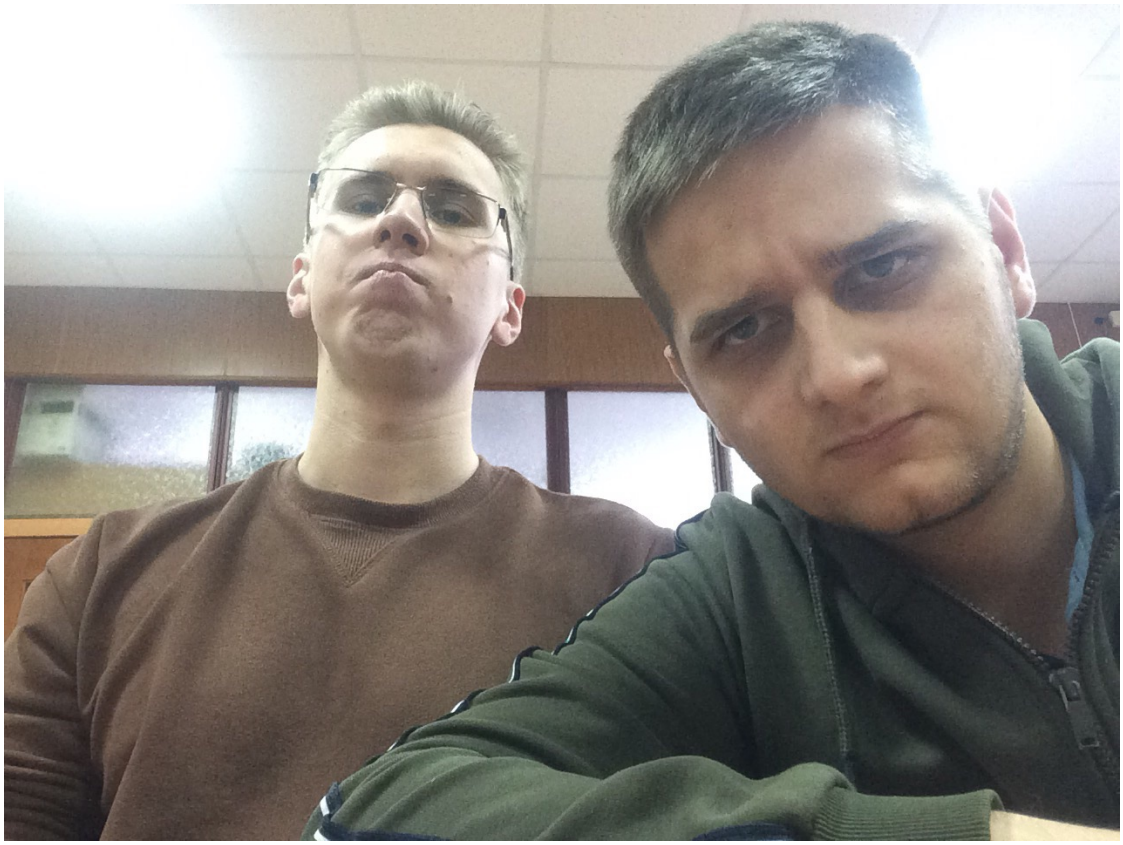
```
            using (ICryptoTransform crypt = crypt_AES.CreateDecryptor(crypt_AES.Key,
crypt_AES.IV))
            {
                using (MemoryStream ms = new MemoryStream())
                {
                    using (CryptoStream cs = new CryptoStream(ms, crypt,
CryptoStreamMode.Write))
                    {
                        cs.Write(cip_data, 0, cip_data.Length);
                    }
                    data = ms.ToArray();
                }
            }
            return data;
        }


        static void Main(string[] args)
        {
            int choice = 0;
            Console.WriteLine("1 - Шифровать файл, 2 - Дешифровка файла.");
            choice = Console.Read();
            Console.ReadLine();

            if (choice == '1') // Шифрование
            {
                Console.WriteLine("Введите пароль: ");
                string password = Console.ReadLine();
                byte[] password_byte = Encoding.Default.GetBytes(password);

                Console.WriteLine("Выберите алгоритм для хэширования. 1 - SHA256, 2 - SHA1, 3
- MD5");
                int p = Console.Read();
                Console.ReadLine();

                bool flag = true;
                byte[] password_hash = new byte[32];

                switch (p)
                {
                    case '1':
                        {
                            var hash_sha256 = HashAlgorithm.Create("SHA256");
                            password_hash = hash_sha256.ComputeHash(password_byte);
                            break;
                        }
                    case '2':
                        {
                            var hash_sha1 = HashAlgorithm.Create("SHA1");
                            password_hash = hash_sha1.ComputeHash(password_byte);
                            break;
                        }
                    case '3':
                        {
                            var hash_md5 = HashAlgorithm.Create("MD5");
                            password_hash = hash_md5.ComputeHash(password_byte);
                            break;
                        }
                    default:
                        flag = false;
                        break;
                }

                if (flag)
                {
```

```
                Console.WriteLine("Введите имя оригинального файла!");
                var string1 = Console.ReadLine();
                var bytes_fr_image = File.ReadAllBytes(string1);

                var crp_data = AES_Cipher(password_hash, bytes_fr_image);

                Console.WriteLine("Введите имя для результата шифрования!");
                var string2 = Console.ReadLine();

                File.WriteAllBytes(string2, crp_data);
            }
        }
        else if (choice == '2') // Дешифровка
        {
            Console.WriteLine("Введите пароль: ");
            string password2 = Console.ReadLine();
            byte[] password_byte2 = Encoding.Default.GetBytes(password2);

            Console.WriteLine("Выберите алгоритм для хэширования. 1 - SHA256, 2 - SHA1, 3
- MD5");
            int p = Console.Read();
            Console.ReadLine();


            byte[] password_hash2 = new byte[32];
            bool flag = true;

            switch (p)
            {
                case '1':
                    {
                        var hash_sha256 = HashAlgorithm.Create("SHA256");
                        password_hash2 = hash_sha256.ComputeHash(password_byte2);
                        break;
                    }
                case '2':
                    {
                        var hash_sha1 = HashAlgorithm.Create("SHA1");
                        password_hash2 = hash_sha1.ComputeHash(password_byte2);
                        break;
                    }
                case '3':
                    {
                        var hash_md5 = HashAlgorithm.Create("MD5");
                        password_hash2 = hash_md5.ComputeHash(password_byte2);
                        break;
                    }
                default:
                    flag = false;
                    break;
            }

            if (flag)
            {
                Console.WriteLine("Введите имя шифрованного файла!");
                var string1 = Console.ReadLine();
                var bytes_fr_image = File.ReadAllBytes(string1);

                var decr_data = AES_Decipher(password_hash2, bytes_fr_image);

                Console.WriteLine("Введите имя для записи результата расшифровки!");
                var string2 = Console.ReadLine();

                File.WriteAllBytes(string2, decr_data);
```

```
                }
            }
            else
                Console.WriteLine("Некорректный выбор! Повторите снова.");

        }
    }
}
```

Сделать селфи бригады (в полном составе, должно быть видно лица), зашифровать и выложить на общий диск. Пароль указать в отчете.

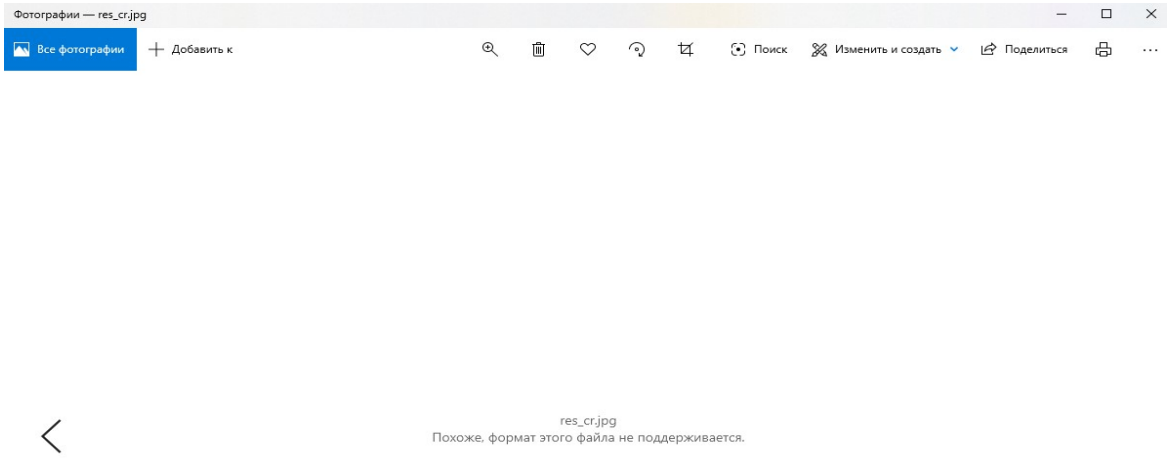
Файл 1.jpg. Пароль: postv’te_zachet



Шифрование картинки:

```
1 - Шифровать файл, 2 - Дешифровка файла.
1
Введите пароль:
postv'te_zachet
Выберите алгоритм для хэширования. 1 - SHA256, 2 - SHA1, 3 - MD5
1
Введите имя оригинального файла!
1.jpg
Введите имя для результата шифрования!
res_cr.jpg
```

Результат шифрования res_cr.jpg



Расшифровка картинки:

```
1 - Шифровать файл, 2 - Дешифровка файла.
2
Введите пароль:
postv'te_zachet
Выберите алгоритм для хэширования. 1 - SHA256, 2 - SHA1, 3 - MD5
1
Введите имя шифрованного файла!
res_cr.jpg
Введите имя для записи результата расшифровки!
out.jpg
```

Результат расшифровки out.jpg