

Министерство образования и науки Российской Федерации
Новосибирский государственный технический университет
Кафедра прикладной математики

Основы криптографии
Лабораторная работа №3

Факультет: прикладной математики и информатики
Группа: ПМ-63
Студенты: Шепрут И.М.
 Кожекин М.В.
 Утюганов Д.С.
Преподаватель: Ступаков И.М.

Новосибирск

2018

1. Цель работы

Изучить вычисление хеш сумм и работу с ECDH и HMAC в языке C#

2. Ход работы

Задача 1

Хеш функции. Взять файл (лучше с известными из надежного источника хешами) и вычислить хеши по алгоритмам MD5 и SHA-256. Использовать класс HashAlgorithm

Checksum.cs

```
using System;
using System.IO;
using System.Security.Cryptography;

class Checksum
{
    static string CalculateMD5(string filename)
    {
        using (var md5 = MD5.Create())
        {
            using (var stream = File.OpenRead(filename))
            {
                var hash = md5.ComputeHash(stream);
                return BitConverter.ToString(hash).Replace("-", "").ToLowerInvariant();
            }
        }
    }

    static string CalculateSHA1(string filename)
    {
        using (var sha1 = SHA1.Create())
        {
            using (var stream = File.OpenRead(filename))
            {
                var hash = sha1.ComputeHash(stream);
                return BitConverter.ToString(hash).Replace("-", "").ToLowerInvariant();
            }
        }
    }

    static string CalculateSHA256(string filename)
    {
        using (var sha256 = SHA256.Create())
        {
            using (var stream = File.OpenRead(filename))
            {
                var hash = sha256.ComputeHash(stream);
                return BitConverter.ToString(hash).Replace("-", "").ToLowerInvariant();
            }
        }
    }

    static string CalculateSHA512(string filename)
    {
        using (var sha512 = SHA512.Create())
        {
            using (var stream = File.OpenRead(filename))
            {
```

```

        var hash = sha512.ComputeHash(stream);
        return BitConverter.ToString(hash).Replace("-", "").ToLowerInvariant();
    }
}

static void CalculateCheckSums(string filename)
{
    Console.WriteLine("MD5 checksum: " + CalculateMD5(filename));
    Console.WriteLine("SHA-1 checksum: " + CalculateSHA1(filename));
    Console.WriteLine("SHA256 checksum: " + CalculateSHA256(filename));
    Console.WriteLine("SHA512 checksum: " + CalculateSHA512(filename));
}

static void Main(string[] args)
{
    // Task 1
    CalculateCheckSums("npp.7.3.Installer.x64.exe");
}
}

```

Проверим работу программы, скачав установщик Notepad++ 7.3 и сверим его контрольную сумму с сайтом и сторонней программой:

```

C:\Windows\system32\cmd.exe
MD5 checksum: 205789f139a7125f90603be2694c2724
SHA-1 checksum: e7306df1d6e81801fb4be0868610db70e979b0aa
SHA256 checksum: 857a71b794695839181706c689b36f23a8ffa64eb060b49cb860a2dfe8315e38
SHA512 checksum: 7f3632c2faf3aca8ec45ab3d95ef25ef715f224deb0d13faf0313683a381372ba1
d776e81bd5458b2096e5768ceaea4ca5718b068b94192586fd92749cd1f14b
Для продолжения нажмите любую клавишу . . .

```

Контрольная сумма	
Имя	npp.7.3.Installer.x64.exe
Размер	2851328 байтов (2784 KiB)
CRC32	CF7BFEC2
CRC64	127025038725537C
SHA256	857A71B794695839181706C689B36F23A8FFA64EB060B49CB860A2DFE8315E38
SHA1	E7306DF1D6E81801FB4BE0868610DB70E979B0AA
BLAKE2sp	AC5C6E4E0BE1A955E43673914142756E33FAFB437A383EB04D10E6061ECB2B94

OK Отмена

https://notepad-plus-plus.org/re... x + - □ ×

← → ↻ https://notepad-plus-plus.org/repository/7.x/7.... ☆ 👤 ⋮

SHA-1 Digest

c1e0b163ae709d0795a47c3dbe148b9203483e64	npp.7.3.bin.7z
e7306df1d6e81801fb4be0868610db70e979b0aa	npp.7.3.Installer.x64.exe
d4c403675a21cc381f640b92e596bae3ef958dc6	npp.7.3.Installer.exe
84fcfbb6eacd6a7a3beb36f8192306f54c99a941	npp.7.3.bin.zip
098fe55aee4d0ffd06ea0d2d660f3af2ae61d665	npp.7.3.bin.x64.zip
a18a57e0c95480e2dd8d49cab4fbf25b061d3f00	npp.7.3.bin.x64.7z
442047cba62a8b410cd769d396ebb5377c27b09f	npp.7.3.bin.minimalist.x64.7z
b5261c8a16bf638c2e1990fff77e770a7fd14e1d	npp.7.3.bin.minimalist.7z

MD5 Digest

231f2bc1f1bf4f69947a1552c4611cba	npp.7.3.bin.7z
2a0fabf83b894dcbc8d322f2def1ae1a	npp.7.3.bin.minimalist.7z
3b56bb725c9daf1915883e70d98aefa0	npp.7.3.bin.minimalist.x64.7z
3723b8039dff95f437704f3cd8fd6e72	npp.7.3.bin.x64.7z
14e3a92f0cc053793419e7142d788676	npp.7.3.bin.x64.zip
c55109908179eecdacba02a38d854d6	npp.7.3.bin.zip
78e778c48b8376fe6adfa67f2cadf05e	npp.7.3.Installer.exe
205789f139a7125f90603be2694c2724	npp.7.3.Installer.x64.exe

Все контрольные суммы совпали.

Задача 2

Используя класс ECDiffieHellman реализовать обмен ключами.

Exchange.cs

```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Linq;

class Program
{
    static byte[] StringToByteArray(string hash)
    {
        return Enumerable.Range(0, hash.Length)
            .Where(x => x % 2 == 0)
            .Select(x => Convert.ToByte(hash.Substring(x, 2), 16))
            .ToArray();
    }

    static void keysExchange()
    {
        byte[] inputBuffer = new byte[1024];
        Stream inputStream = Console.OpenStandardInput(inputBuffer.Length);
        Console.SetIn(new StreamReader(inputStream, Console.InputEncoding, false,
inputBuffer.Length));

        byte[] OurPublicKey;
        byte[] TheirPublicKey;

        using (var ecd = new ECDiffieHellmanCng())
        {
            ecd.KeyDerivationFunction = ECDiffieHellmanKeyDerivationFunction.Hash;
            ecd.HashAlgorithm = CngAlgorithm.Sha256;
            OurPublicKey = ecd.PublicKey.ToByteArray();
            Console.WriteLine("Our public key: {0}",
BitConverter.ToString(OurPublicKey).Replace("-", "").ToLowerInvariant());

            Console.WriteLine("Input their public key:");
            string str = Console.ReadLine();
            TheirPublicKey = StringToByteArray(str);
            byte[] ourSecret = ecd.DeriveKeyMaterial(CngKey.Import(TheirPublicKey,
CngKeyBlobFormat.EccPublicBlob));
            Console.WriteLine("Our secret key: {0}",
BitConverter.ToString(ourSecret).Replace("-", "").ToLowerInvariant());
        }

        static void Main(string[] args)
        {
            // Task 2
            keysExchange();
        }
    }
}
```

```
C:\Windows\system32\cmd.exe
Our public key: 45434b35420000000203054cd820f2e96b6e211af964eba5217f1364aea4e8ddc4b13964069d41aa47866c254c148e36a094d41a55053c5a8373ca63741292497b8f0147cdc566f14b000dc062853ba965e4b8105b50dd384804bc4f2be71ca97f7c6bebf43769d74e5b8966a59be452ae6200d6fb622256606b5293baf9c8436caa241cd1103a1349ca462
Input their public key:45434b3542000000015a9eea34c5ecef8b1989ef84297283854cc3a33703958dc43c9add3105d23f8c5c041775b9ccbfe50d3a4332823c4e3d489ff194109b13930677c48afa1c87a25000d72e7d2094dc8ecef7a1a95def386406ad48bbe5b48fc3e48cbcc309481f9c470305bebecefd8cf8dcd2761f9cbf28d09fc2d6228b8c2ee42d59a7f8293e1bcec
Our secret key: c8a149083a970c89ebe959c5238254f63d8a0fcfb29acf35623038fdad752827
Для продолжения нажмите любую клавишу . . .

C:\Windows\system32\cmd.exe
Our public key: 45434b3542000000015a9eea34c5ecef8b1989ef84297283854cc3a33703958dc43c9add3105d23f8c5c041775b9ccbfe50d3a4332823c4e3d489ff194109b13930677c48afa1c87a25000d72e7d2094dc8ecef7a1a95def386406ad48bbe5b48fc3e48cbcc309481f9c470305bebecefd8cf8dcd2761f9cbf28d09fc2d6228b8c2ee42d59a7f8293e1bcec
Input their public key:45434b35420000000203054cd820f2e96b6e211af964eba5217f1364aea4e8ddc4b13964069d41aa47866c254c148e36a094d41a55053c5a8373ca63741292497b8f0147cdc566f14b000dc062853ba965e4b8105b50dd384804bc4f2be71ca97f7c6bebf43769d74e5b8966a59be452ae6200d6fb622256606b5293baf9c8436caa241cd1103a1349ca462
Our secret key: c8a149083a970c89ebe959c5238254f63d8a0fcfb29acf35623038fdad752827
Для продолжения нажмите любую клавишу . . .
```

Задача 3

Используя класс HMAC и полученный секретный ключ, реализовать генерацию и валидацию подписи сообщения.

HMAC.cs

```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Linq;
using System.Collections;

class Program
{
    static byte[] StringToByteArray(string hash)
    {
        return Enumerable.Range(0, hash.Length)
            .Where(x => x % 2 == 0)
            .Select(x => Convert.ToByte(hash.Substring(x, 2), 16))
            .ToArray();
    }

    static byte[] SignFile(string filename, byte[] key)
    {
        using (var HMAC = new HMACSHA256(key))
        {
            using (var file = new FileStream(filename, FileMode.Open))
            {
                return HMAC.ComputeHash(file);
            }
        }
    }
}
```

```

    }
}

static bool ValidateFile(string filename, byte[] key, byte[] expectedHash)
{
    byte[] realHash;

    using (var HMAC = new HMACSHA256(key))
    {
        using (var file = new FileStream(filename, FileMode.Open))
        {
            realHash = HMAC.ComputeHash(file);
        }
    }

    return StructuralComparisons.StructuralEqualityComparer.Equals(realHash,
expectedHash);
}

static void SignAndValidate()
{
    Console.WriteLine("Enter name of a file");
    string filename = Console.ReadLine();
    Console.WriteLine("Enter the key");
    byte[] key = StringToByteArray(Console.ReadLine());
    byte[] hmac = SignFile(filename, key);

    Console.WriteLine("HMAC = {0}", BitConverter.ToString(hmac).Replace("-",
"").ToLowerInvariant());

    Console.WriteLine("Enter HMAC:");
    hmac = StringToByteArray(Console.ReadLine());
    bool isVerified = ValidateFile(filename, key, hmac);

    if (isVerified)
        Console.WriteLine("HMAC matched");
    else
        Console.WriteLine("HMAC didn't matched");
}

static void Main(string[] args)
{
    // Task 3
    SignAndValidate();
}
}

```

Файл: npp.7.3.Installer.x64.exe

Ключ (256 бит): 857a71b794695839181706c689b36f23a8ffa64eb060b49cb860a2dfe8315e38

НМАС: 783271db0368bf4la246ede33b4feceabObS72912376f6S3fddb9S0c2cdb6e2e

Результат:

```
C:\Windows\system32\cmd.exe
Enter name of a file
npp.7.3.Installer.x64.exe
Enter the key
857a71b794695839181706c689b36f23a8ffa64eb060b49cb860a2dfe8315e38
HMAC = 783271dbf868bf41a240ede33b4feceab0b572912376f683fddb950c2cdb6e2e
Enter HMAC:
783271dbf868bf41a240ede33b4feceab0b572912376f683fddb950c2cdb6e2e
HMAC matched
Для продолжения нажмите любую клавишу . . .
```

Задача 4 (бонусная)

Написать собственный HMAC.

MyHMAC.cs

```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Linq;
using System.Collections;

public class MyHMAC : IDisposable
{
    public MyHMAC(byte[] new_key)
    {
        key = new_key;
        ipad = 0x36;
        opad = 0x5C;
        blockSize = 64;
    }

    private byte[] StringToByteArray(string str)
    {
        return Enumerable.Range(0, str.Length)
            .Where(x => x % 2 == 0)
            .Select(x => Convert.ToByte(str.Substring(x, 2), 16))
            .ToArray();
    }

    private byte[] XOR(byte[] buffer1, byte buffer2)
    {
        byte[] result = new byte[blockSize];
        for (int i = 0; i < blockSize; i++)
            result[i] = (byte)(buffer1[i] ^ buffer2);
        return result;
    }

    public byte[] ComputeHash(Stream s)
    {
        byte[] text = StreamToByteArray(s);
        byte[] K = key;

        if (K.Length > blockSize)
            K = H(K);

        if (K.Length < blockSize)
        {
            byte[] newArray = new byte[blockSize];
```



```

        K.CopyTo(newArray, 0);
        for (int i = K.Length; i < blockSize; ++i)
            newArray[i] = 0;

        K = newArray;
    }

    return H(concat(XOR(K, opad), H(concat(XOR(K, ipad), text))));
}

private static byte[] concat(byte[] a, byte[] b)
{
    byte[] c = new byte[a.Length + b.Length];
    a.CopyTo(c, 0);
    b.CopyTo(c, a.Length);

    return c;
}

private byte[] H(byte[] arr)
{
    using (var sha256 = SHA256.Create())
        return sha256.ComputeHash(arr);
}

private byte[] StreamToByteArray(Stream input)
{
    using (MemoryStream ms = new MemoryStream())
    {
        input.CopyTo(ms);
        return ms.ToArray();
    }
}

public void Dispose()
{
}

private byte ipad;
private byte opad;
private int blockSize;
private byte[] key;
}

class Program
{
    static byte[] StringToByteArray(string hash)
    {
        return Enumerable.Range(0, hash.Length)
            .Where(x => x % 2 == 0)
            .Select(x => Convert.ToByte(hash.Substring(x, 2), 16))
            .ToArray();
    }

    static byte[] MySignFile(string filename, byte[] key)
    {
        using (MyHMAC HMAC = new MyHMAC(key))
        {
            using (var file = new FileStream(filename, FileMode.Open))
            {

```

```

        return HMAC.ComputeHash(file);
    }
}

static bool MyValidateFile(string filename, byte[] key, byte[] expectedHash)
{
    byte[] realHash;

    using (var HMAC = new MyHMAC(key))
    {
        using (var file = new FileStream(filename, FileMode.Open))
        {
            realHash = HMAC.ComputeHash(file);
        }
    }

    return StructuralComparisons.StructuralEqualityComparer.Equals(realHash,
expectedHash);
}

static void MySignAndValidate()
{
    Console.WriteLine("Enter name of a file");
    string filename = Console.ReadLine();
    Console.WriteLine("Enter the key");
    string str = Console.ReadLine();
    byte[] key = StringToByteArray(str);
    byte[] hmac = MySignFile(filename, key);

    Console.WriteLine("HMAC = {0}", BitConverter.ToString(hmac).Replace("-",
"").ToLowerInvariant());

    Console.WriteLine("Enter HMAC:");
    hmac = StringToByteArray(Console.ReadLine());
    bool isVerified = MyValidateFile(filename, key, hmac);

    if (isVerified)
        Console.WriteLine("HMAC matched");
    else
        Console.WriteLine("HMAC didn't matched");
}

static void Main(string[] args)
{
    // Task 4
    MySignAndValidate();
}
}

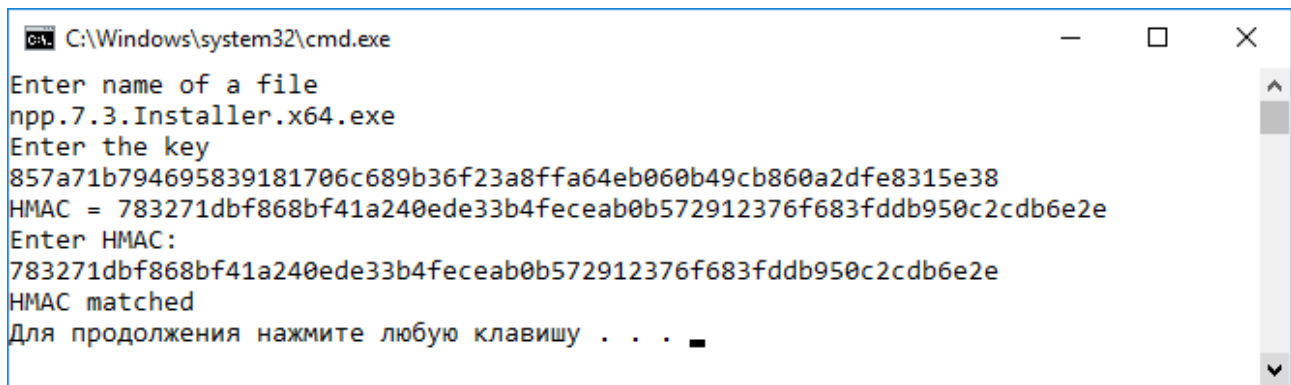
```

Файл: npp.7.3.Installer.x64.exe

Ключ (256 бит): 857a71b794695839181706c689b36f23a8ffa64eb060b49cb860a2dfe8315e38

НМАС: 783271db0368bf4la246ede33b4feceabObS72912376f6S3fddb9S0c2cdb6e2e

Результат:



```
C:\Windows\system32\cmd.exe
Enter name of a file
npp.7.3.Installer.x64.exe
Enter the key
857a71b794695839181706c689b36f23a8ffa64eb060b49cb860a2dfe8315e38
HMAC = 783271dbf868bf41a240ede33b4feceab0b572912376f683fddb950c2cdb6e2e
Enter HMAC:
783271dbf868bf41a240ede33b4feceab0b572912376f683fddb950c2cdb6e2e
HMAC matched
Для продолжения нажмите любую клавишу . . .
```

3. Выводы

В ходе выполнения 3 лабораторной работы было изучена работа классов ECDH и HMAC из библиотеки System.Security.Cryptography языка C#. Также был написан собственный HMAC.