



МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

НГТУ



НЭТИ

Кафедра прикладной математики

Практическое задание № 3

по дисциплине «Основы криптографии»

ВЫЧИСЛЕНИЕ ХЕШ-СУММЫ И ЦИФРОВОЙ ПОДПИСИ



Группа

ПМ-81

Бригада

3

БАСОВ ДЕНИС

ЮРГАНОВ ЕГОР

Преподаватель

СТУПАКОВ ИЛЬЯ МИХАЙЛОВИЧ

Дата

26.12.2020

Новосибирск

Часть 1

1. Цель:

Научиться использовать готовые криптографические примитивы для вычисления хеш-сумм и цифровой подписи.

2. Задача:

- Используя алгоритм RSA/ECDSA сгенерировать ключи.
- Сохранить публичный ключ в виде X.509 сертификата (файл с расширением .cer).
- Сохранить приватный ключ в виде PKCS #12 сертификата (файл с расширением .pem).
- Сделать программу генерации цифровой подписи файлов (должна принимать подписываемый файл и файл с расширением .pem и сохранять подпись в указанный файл).
- Сделать программу для вычисления хеш-суммы файла (MD5, SHA-1, SHA-256).
- Сделать два селфи бригады (в полном составе, должно быть видно лица), сгенерировать для одного из них подпись и вычислить хеш-сумму и выложить на гугл диск 5 файлов (2 фото, X.509 сертификат, файл с подписью и файл с хеш-суммой).
- Сделать программу валидации цифровой подписи файлов.
- Взять файлы выложенные бригадой с номером на единицу меньше и проверить подпись и хеш-сумму.

3. Программа

```
using System;
using System.Text;
using System.Linq;
using System.Security.Cryptography;
using System.IO;
using System.Security.Cryptography.X509Certificates;

namespace kripa3
{
    class Program
    {
        static void Keys(string name) //генерация ключей
        {
            var rsa = RSA.Create();

            //создаем запрос на сертификат
            var request = new CertificateRequest("CN=b8103", rsa, HashAlgorithmName.SHA256,
            RSASignaturePadding.Pkcs1);

            //создаем самоподписанный сертификат на год
            var cert = request.CreateSelfSigned(DateTimeOffset.Now,
            DateTimeOffset.Now.AddYears(1));

            File.WriteAllBytes(name + ".cer", cert.Export(X509ContentType.Cert)); //открытый
            File.WriteAllBytes(name + ".pem", cert.Export(X509ContentType.Pkcs12));
        }
        //вычисления хеш-суммы файла
        static string CreateHash(string destination, string input, string algo)
        {
            var hash = HashAlgorithm.Create(algo); //алгоритм хеширования

            //хешируем и записываем в указанный файл
            byte[] hashb = hash.ComputeHash(File.ReadAllBytes(input));
            File.WriteAllBytes(destination, hashb);
            return string.Join("-", hashb.Select(x => x.ToString("x2")));
        }
    }
}
```

```

    }

    static HashAlgorithmName hash(string cert) //алгоритм получения хэша сертификата
    {
        var PublicCert = new X509Certificate2(cert);
        var algo = PublicCert.SignatureAlgorithm.FriendlyName;
        HashAlgorithmName hashAlgorithmName;
        if (algo == "SHA1")
            hashAlgorithmName = HashAlgorithmName.SHA1;
        if (algo == "SHA256")
            hashAlgorithmName = HashAlgorithmName.SHA256;
        else
            hashAlgorithmName = HashAlgorithmName.MD5;
        return hashAlgorithmName;
    }
    //генерация цифровой подписи
    static string CreateSign(string destination, string input, string pem, string c)
    {
        HashAlgorithmName hashAlgorithmName = hash(c);

        var cert = new X509Certificate2(pem);

        //берем закрытый ключ, подписываем файлы
        var sign = cert.GetRSAPrivateKey().SignData(File.ReadAllBytes(input),
hashAlgorithmName, RSASignaturePadding.Pkcs1);
        File.WriteAllBytes(destination, sign);

        return string.Join("-", sign.Select(x => x.ToString("x2")));
    }
    //программа валидации цифровой подписи
    static string Verify(string name, string hashed, string signed, string cert)
    {
        var sign = File.ReadAllBytes(signed);
        var HashFile = File.ReadAllBytes(hashed);
        var file = File.ReadAllBytes(name);

        HashAlgorithmName hashAlgorithmName = hash(cert);

        var PublicCert = new X509Certificate2(cert);
        var algo = PublicCert.PublicKey.Key.SignatureAlgorithm;

        //проверка подписи и хэша
        if (algo == "RSA")//для rsa
        {
            var PublicKey = PublicCert.GetRSAPublicKey();
            if (PublicKey.VerifyData(file, sign, hashAlgorithmName,
RSASignaturePadding.Pkcs1) && PublicKey.VerifyHash(HashFile, sign, hashAlgorithmName,
RSASignaturePadding.Pkcs1))
                return (name + " VALID");
        }
        if (algo == "ECDsa")//для ecdsa
        {
            var PublicKey = PublicCert.GetECDsaPublicKey();
            if (PublicKey.VerifyData(file, sign, hashAlgorithmName) &&
PublicKey.VerifyHash(HashFile, sign))
                return (name + " VALID");
        }
        return (name + " INVALID");
    }
}

static void Main(string[] args)
{
    Keys("certificat");
    Console.WriteLine(CreateHash("hash.hash", "1.jpg", "SHA-256"));
}

```





```

        Console.WriteLine(" ");
        Console.WriteLine(CreateSign("sign.sgn", "1.jpg", "certificat.pem",
"certificat.cer"));
        Console.WriteLine(" ");
        Console.WriteLine(Verify("1.jpg", "hash.hash", "sign.sgn", "certificat.cer"));
        //Console.WriteLine(Verify("img10.jpg", "hash.txt", "sign.txt", "cert.cer"));
    }
}
}

```

4. Результат

Публичный ключ в виде X.509 с расширением .cer, приватный ключ в виде PKCS#12 с расширением .pem, цифровая подпись и хеш-сумма файла.(для 1.jpg)

 certificat	02.12.2020 20:06	Сертификат безо...
 certificat.pem	02.12.2020 20:06	Файл "PEM"
 hash.hash	02.12.2020 20:06	Файл "HASH"
 sign.sgn	02.12.2020 20:06	Файл "SGN"

64-aa-59-8e-f2-e4-58-9b-84-6f-ec-41-a4-62-e0-6a-11-92-e7-c1-0a-6c-b5-5c-b3-39-0f-85-0e-eb-9d-93

8d-6a-79-be-ec-b1-b4-f6-05-d0-fe-eb-cf-d9-65-d4-0a-19-e5-d6-3f-21-e1-60-03-4f-45-f7-58-05-c8-e5-b6-db-5f-b0-89-ac-0c-82-6b-68-34-3a-d5-e6-e0-90-7c-1d-c5-09-fa-66-d3-36-9d-4d-fc-7c-57-cb-58-7e-54-42-07-8b-9d-19-1e-86-8c-a7-a6-3f-14-d6-c3-69-33-61-6d-54-18-b3-0a-b3-ff-3b-d4-d1-d5-3c-8e-25-39-6f-db-d4-c2-7b-80-a5-b7-b9-ae-2e-30-f1-93-15-04-6b-a2-c8-18-03-ca-05-8e-51-d1-65-3c-3d-05-5c-c4-cb-bc-4d-8f-da-2e-dc-4b-53-03-d4-a1-d6-a7-67-4a-d6-b4-66-9e-e8-fe-22-b3-61-21-de-0b-6f-e3-77-dd-88-4b-4f-d3-4d-83-44-a5-b9-3b-40-c1-c2-d9-ee-a0-3b-26-f3-f1-d5-aa-43-c2-81-95-b5-fd-4c-1a-9f-d0-f0-e0-e8-a0-7d-98-d6-40-2d-b5-14-16-f8-ba-29-85-94-08-5e-92-e8-f2-f6-f8-f3-09-ff-ff-72-5d-21-48-80-fc-ed-72-0b-87-67-1f-fc-5b-3a-c0-95-20-8e-fd-79-0e-03-2e-6f-28-a8-2a-0a-78-24-97-1b-92-16



Сведения о сертификате

Нет доверия к этому корневому сертификату центра сертификации. Чтобы включить доверие, установите этот сертификат в хранилище доверенных корневых сертификатов центров сертификации.

Кому выдан: b8103

Кем выдан: b8103

Действителен с 17.12.2020 по 17.12.2021



1.jpg



2.jpg

Проверяли работу 10 бригады(Судьярова, Зуенок).

img10.jpg VALID

img10_1.jpg INVALID



img10.jpg



img10_1.jpg