УО «Белорусский государственный технологический университет»

Факультет информационных технологий

Кафедра «Информационных систем и технологий»

Специальность 1-40 05 01 «Информационные Системы и Технологии»

**Основы информационной безопасности**

**Практическое задание № 8**

**Вариант 17**

|  |  |  |  |
| --- | --- | --- | --- |
| **Выполнил** |  |  |  |
| Студент 2 курса группы 1 |  |  | Д.И. Велютич |
|  | подпись, дата |  | инициалы и фамилия |
| **Проверил(а)** |  |  | Н.В. Ржеутская |
|  | подпись, дата |  | инициалы и фамилия |

Минск 2023

**Тема «Изучение стандартных средств для реализации приложений, использующих симметричное и ассиметричное шифрование с использованием библиотеки** [**System.Security.Cryptography**](https://docs.microsoft.com/ru-ru/dotnet/api/system.security.cryptography)**»**

Цель: Изучить модель криптографии .NET Framework, основные классы и структуры данных, разработать приложение для шифрования файлов использующих симметричные и ассиметричные алгоритмы шифрования

**Теория**

В .Net Framework есть пространство имен под названием System.Security.Cryptography, которое предоставляет криптографические службы для безопасного кодирования и декодирования данных, хеширования сообщений, генерации случайных чисел и проверки подлинности сообщений.

Данное пространство имён поддерживает различные реализации алгоритмов, такие как CryptoApi (CAPI) и Cryptography Next Generation API (CNG API), а также алгоритмы на основе OpenSsl.

CryptoAPI предоставляет стандартные функции для работы с криптопровайдерами и входит в операционные системы Microsoft, начиная с Windows 2000.

Cryptography Next Generation (CNG) является долгосрочной заменой CAPI и поддерживает алгоритмы, предлагаемые CAPI, а также другие алгоритмы, включая те, перечисленные в правилах Suite B Агентства национальной безопасности США. CNG поддерживает различные длины ключей для разных алгоритмов.

Для симметричных алгоритмов в .Net, существует структура наследования, где SymmetricAlgorithm является абстрактным классом, и для каждого алгоритма существует соответствующий абстрактный класс для его реализации.

Пример шифрования сообщения с использованием алгоритма Aes включает создание объекта шифратора, потоков для шифрования и использование CryptoStream для связи потоков данных с криптографическим преобразованием.

Для асимметричных алгоритмов, таких как RSA, можно использовать встроенные функции Encrypt и Decrypt, а также генерировать ключи и выполнять шифрование и дешифрование данных. RSA является блочным алгоритмом, поэтому данные нужно дополнять до длины блока.

Хеширование в .Net выполняется с использованием классов, таких как SHA256, где после инициализации экземпляра для хеширования нужно вызвать метод ComputeHash для выполнения хеширования данных.

Все эти операции и множество других криптографических функций доступны в .Net Framework через пространство имен System.Security.Cryptography.

1. Основное назначение библиотеки System.Security.Cryptography:

Библиотека System.Security.Cryptography в .Net Framework предназначена для выполнения криптографических операций. Ее основное назначение заключается в обеспечении безопасности данных путем предоставления средств для выполнения следующих задач:

- Кодирование и декодирование данных, включая шифрование и расшифрование.

- Генерация хешей и вычисление цифровых подписей.

- Генерация случайных чисел.

- Проверка подлинности сообщений.

- Работа с различными алгоритмами шифрования и хеширования.

2. Интерфейс CryptoApi:

CryptoAPI (CAPI) представляет собой интерфейс программирования приложений, предназначенный для обеспечения разработчиков Windows-приложений стандартным набором функций для работы с криптопровайдерами. Он входит в состав операционных систем Microsoft, начиная с Windows 2000. CryptoAPI предоставляет средства для выполнения различных криптографических операций, включая шифрование, хеширование, генерацию ключей и другие задачи, связанные с криптографией.

3. Cryptography Next Generation (CNG):

Cryptography Next Generation (CNG) представляет собой набор интерфейсов и архитектуру для криптографии, которая стала долгосрочной заменой для CryptoAPI (CAPI). CNG поддерживает все алгоритмы, предлагаемые CAPI, а также другие алгоритмы, включая те, которые перечислены в правилах Suite B Агентства национальной безопасности США. CNG предоставляет более гибкий и мощный набор инструментов для работы с криптографией и поддерживает различные длины ключей для разных алгоритмов. Эта технология обеспечивает более современные и безопасные методы криптографии в операционных системах Windows.

**Условия задачи**

1. Ознакомиться с созданием криптографического приложения;
2. Выполнить шифрование, дешифрование и хеширование своей фамилии по указанным алгоритмам. Используя функции работы с файлами сохранить ключи шифрования, результаты шифрования и хеширования.
3. Для выше указанных алгоритмов используя Hex-редактор продемонстрировать ключи шифрования, зашифрованные и захешированные данные.

\*Реализовать проверку сообщения (фамилии) и хеша по примеру ЭЦП. Также продемонстрировать что будет если будет изменен хеш или сообщение.

|  |  |  |
| --- | --- | --- |
| 17 | AES (192bit) | SHA1 |

**Исполнительная часть**

using System;

using System.IO;

using System.Security.Cryptography;

using System.Text;

public class Program

{

public static void Main()

{

string originalData = "Велютич";

byte[] originalBytes = Encoding.UTF8.GetBytes(originalData);

byte[] hash; // Переменная для хранения хеша

// Step 1: AES Encryption and Saving Keys and Encrypted Data

using (Aes aesAlg = Aes.Create())

{

aesAlg.KeySize = 192;

aesAlg.GenerateKey();

aesAlg.GenerateIV();

ICryptoTransform encryptor = aesAlg.CreateEncryptor();

byte[] encryptedBytes;

using (MemoryStream msEncrypt = new MemoryStream())

{

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

{

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

}

encryptedBytes = msEncrypt.ToArray();

}

File.WriteAllBytes("aes\_key.bin", aesAlg.Key);

File.WriteAllBytes("aes\_iv.bin", aesAlg.IV);

File.WriteAllBytes("aes\_encrypted.bin", encryptedBytes);

}

Console.WriteLine("AES Encryption completed.");

// Step 2: AES Decryption and Verification

byte[] key = File.ReadAllBytes("aes\_key.bin");

byte[] iv = File.ReadAllBytes("aes\_iv.bin");

byte[] encryptedData = File.ReadAllBytes("aes\_encrypted.bin");

using (Aes aesAlg = Aes.Create())

{

aesAlg.Key = key;

aesAlg.IV = iv;

ICryptoTransform decryptor = aesAlg.CreateDecryptor();

byte[] decryptedBytes;

using (MemoryStream msDecrypt = new MemoryStream(encryptedData))

{

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

{

using (MemoryStream msDecrypted = new MemoryStream())

{

csDecrypt.CopyTo(msDecrypted);

decryptedBytes = msDecrypted.ToArray();

}

}

}

string decryptedData = Encoding.UTF8.GetString(decryptedBytes);

Console.WriteLine("AES Decrypted Data: " + decryptedData);

}

Console.WriteLine("AES Decryption completed.");

// Step 3: SHA-1 Hashing and Verification

using (SHA1 sha1 = SHA1.Create())

{

hash = sha1.ComputeHash(originalBytes); // Сохраняем хеш в переменную

File.WriteAllBytes("sha1\_hash.bin", hash);

}

byte[] hashFromFile = File.ReadAllBytes("sha1\_hash.bin");

bool isHashValid = CompareByteArrays(hash, hashFromFile);

Console.WriteLine("Original Data: " + originalData);

Console.WriteLine("SHA-1 Hash Validation: " + isHashValid);

}

private static bool CompareByteArrays(byte[] array1, byte[] array2)

{

if (array1.Length != array2.Length)

return false;

for (int i = 0; i < array1.Length; i++)

{

if (array1[i] != array2[i])

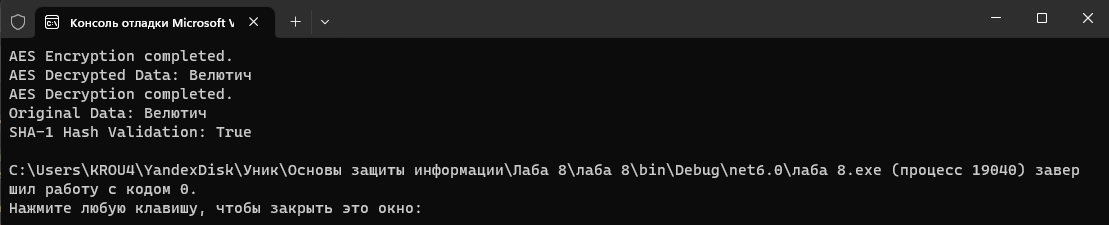
return false;

}

return true;

}

}

Вывод программы:  


**Вывод:** Я изучил модель криптографии .NET Framework, основные классы и структуры данных, разработал приложение для шифрования файлов, использующее как симметричные, так и ассиметричные алгоритмы шифрования. В результате изучения и практической работы с библиотекой System.Security.Cryptography в .NET Framework, я приобрел глубокие знания о принципах криптографии и о том, как она может быть применена для обеспечения безопасности данных. Я разработал полноценное приложение, которое способно шифровать и дешифровать файлы, применяя как симметричные алгоритмы, такие как AES, так и ассиметричные алгоритмы, включая RSA. Это приложение может эффективно защитить конфиденциальные данные, обеспечивая их сохранность и конфиденциальность при передаче и хранении. Это изучение и практический опыт позволили мне углубить свои знания в области криптографии и развивать навыки разработки безопасных приложений.