Учреждение образования

«Белорусский государственный технологический университет»

**Кафедра информационных систем и технологий**

**Практическое занятие №13**

**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)

Выполнил:

Студент 2 курса 7 группы ФИТ

Бобрович Глеб Сергеевич

Проверил:

Барковский Евгений Валерьевич

**Минск 2022 г.**

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

В .NET классы в [System.Security.Cryptography](https://docs.microsoft.com/ru-ru/dotnet/api/system.security.cryptography) пространстве имен управляют множеством сведений о криптографии. Некоторые из них являются оболочками для реализации операционных систем, а другие — исключительно управляемыми реализациями. При создании экземпляра одного из классов, реализующих алгоритмы шифрования, ключи создаются автоматически с целью удобства использования, а принятые по умолчанию значения свойств призваны обеспечить максимальную защищенность.

**Модель криптографии .NET Framework**

.NET Framework предоставляет реализацию многих стандартных криптографических алгоритмов. Эти алгоритмы просты в использовании и по умолчанию имеют наиболее безопасные из возможных значений свойств. Кроме того, в .NET Framework имеется криптографическая модель наследования объектов, поточно-ориентированный подход к разработке.

**Наследование объектов**

Система безопасности .NET Framework реализует расширяемую модель наследования производных классов. Иерархия имеет следующий вид:

- класс типа алгоритма, например SymmetricAlgorithm или HashAlgorithm. Это абстрактный уровень;

- класс алгоритма, является производным от класса типа алгоритма, например RC2 или SHA1. Это абстрактный уровень;

- реализация класса алгоритма, который является производным от класса алгоритма, например RC2CryptoServiceProvider или SHA1Managed. Это уровень реализации алгоритма.

Используя данный шаблон производных классов, можно легко добавить новый алгоритм или новую реализацию существующего алгоритма. Например, для создания нового алгоритма шифрования с открытым ключом можно выполнить наследование от класса AsymmetricAlgorithm.

**Поточно-ориентированный подход**

.NET использует поточно-ориентированный подход для реализации алгоритмов симметричного шифрования и хэширования. Основой такого подхода является класс CryptoStream, производный от класса Stream. Все основанные на потоках криптографические объекты поддерживают единый стандартный интерфейс (CryptoStream) для управления своими частями, ответственными за передачу данных. Благодаря тому, что все эти объекты построены на основе стандартного интерфейса, можно сцеплять вместе различные объекты (например, за объектом, реализующим хэширование, поставить объект, реализующий шифрование) и выполнять ряд операций над данными без использования промежуточных хранилищ данных. Поточная модель также позволяет строить объекты на основе меньших объектов. Например, связанные вместе алгоритмы шифрования и хэширования можно рассматривать как единый поточный объект, несмотря на то, что он может быть построен на основе набора некоторых поточных объектов.

**Основные классы и структуры данных**

Класс CSPParameters – содержит параметры, передаваемые поставщику служб шифрования (CSP), который выполняет криптографические вычисления.

Наличие CSP можно установить, проверив с помощью редактора реестра следующий раздел реестра:

HKEY\_LOCAL\_MACHINE\Software\Microsoft\Cryptography\Defaults\Pr ovider.

Класс CspParameters представляет параметры, которые можно передавать управляемым криптографическим классам, использующим службы шифрования (CSP), с помощью интерфейса Microsoft Cryptography API (CAPI). Классы, имена которых заканчиваются на "CryptoServiceProvider", являются оболочками управляемого кода для соответствующего CSP.

Класс CspParameters используется для выполнения следующих задач:

- задание конкретного CSP путем передачи типа поставщика свойству ProviderType или ProviderName. Можно также задать CSP с помощью перегруженной версии конструктора;

- создание контейнера ключей, в котором можно хранить криптографические ключи. Контейнеры ключей предоставляют самый безопасный способ хранения криптографических ключей и позволяют скрыть их от злоумышленников;

- определение с помощью свойства KeyNumber типа создаваемого асимметричного ключа: ключ подписи или ключ обмена.

Класс RSACryptoServiceProvder - выполняет шифрование и дешифрование данных с помощью реализации асимметричного алгоритма RSA, предоставляемого поставщиком служб шифрования (CSP). Позволяет выполнить экспорт, импорт данных ассиметричной пары ключей. Поддерживаются ключи длиной от 384 до 16384 бит с приращениями по 8 бит, если установлен Microsoft Enhanced Cryptographic Provider, и ключи длиной от 384 до 512 бит с приращениями по 8 бит, если установлен Microsoft Base Cryptographic Provider.

<http://msdn.microsoft.com/ruru/library/system.security.cryptography.rsacryptoserviceprovider.aspx>

Структура RSAParameters - представляет стандартные параметры для алгоритма RSA (значения d, e, n, p, q и т.д.) <http://msdn.microsoft.com/ruru/library/system.security.cryptography.rsaparameters.aspx>

Класс CryptoStream – определяет поток, который связывает потоки данных с криптографическими преобразованиями. <http://msdn.microsoft.com/ruru/library/system.security.cryptography.cryptostream.aspx>

Класс RijndaelManaged – реализует симметричный алгоритм шифрования Rijndael. Поддерживаются ключи длиной 128, 192 и 256 бит. <http://msdn.microsoft.com/ruru/library/system.security.cryptography.rijndaelmanaged.aspx>

Пример приложения для шифрования и дешифровки данных

Предлагаемый пример приложения демонстрирует основные принципы шифрования и расшифровки данных. В приложении используется класс RijndaelManaged, симметричный алгоритм, для шифрования и расшифровки данных из файла с использованием автоматически генерируемых объектов Key и IV. Используется класс RSACryptoServiceProvider, асимметричный алгоритм, для шифрования и дешифровки ключа, используемого в классе RijndaelManaged.

**Обязательные компоненты**: Пространство имен System.IO, System.Security.Cryptography

**Создание ассиметричного ключа**

При выполнении данной задачи создается асимметричный ключ, с помощью которого производится шифрование и дешифровка ключа симметричного алгоритма RijndaelManaged. Этот ключ использовался для шифрования содержимого файла, на форме в элементе управления типа "метка" отображается имя контейнера ключа.

**Шифрование файла**

Для выполнения этой задачи используются два метода: обработчик события для кнопки Encrypt File (buttonEncryptFile\_Click) и метод EncryptFile. Первый метод используется для вывода диалогового окна выбора файла и передачи имени файла второму методу, который выполняет шифрование.

Зашифрованное содержимое, ключ и вектор инициализации сохраняются в объект FileStream, который называется пакетом шифрования.

Метод EncryptFile выполняет следующие действия:

- создает объект RijndaelManaged для шифрования содержимого файла;

- создает объект RSACryptoServiceProvider и выполняется шифрование ключа RijndaelManaged;

- использует объект CryptoStream для чтения и шифрования объекта FileStream исходного файла в виде байтовых блоков в объект назначения FileStream для зашифрованного файла;

- определяет длину зашифрованного ключа и вектора инициализации и создает байтовые массивы соответствующей длины;

- записывает ключ, вектор инициализации и значения их длин в пакет шифрования.

Пакет шифрования имеет следующий формат:

-длина ключа, байты 0-3;

-длина вектора инициализации, байты 4-7;

-зашифрованный ключ; -вектор инициализации;

-зашифрованный текст;

Значения длины ключа и вектора инициализации могут использоваться для определения начальных точек и длин всех частей пакета шифрования, которые затем могут использоваться при расшифровке файла.

**Деширование файла**

Для выполнения этой задачи используются два метода: обработчик события для кнопки Decrypt File (buttonEncryptFile\_Click) и метод DecryptFile. Первый метод используется для вывода диалогового окна выбора файла и передачи имени файла второму методу, который выполняет расшифровку.

Метод Decrypt выполняет следующие действия.

- создает объект симметричного алгоритм RijndaelManaged для расшифровки содержимого;

- считывает первые восемь байтов объекта FileStream зашифрованного пакета в байтовые массивы для получения значений длин зашифрованного ключа и вектора инициализации;

- извлекает ключ и вектор инициализации из пакета шифрования в байтовые массивы;

- создает объект RSACryptoServiceProvider и дешифрирует ключ RijndaelManaged.

- использует объект CryptoStream для чтения и расшифровки зашифрованного текста пакета шифрования FileStream в виде байтовых блоков и загрузки их в объект FileStream для расшифрованного файла. По завершении этой операции дешифровка считается выполненной.

**Экспорт открытого ключа**

В рамках этой задачи ключ, созданный при нажатии кнопки Create Keys, сохраняется в файл. Экспортируются только открытый ключ.

Данная задача воссоздает ситуацию, в которой Алиса предоставляет Бобу открытый ключ, чтобы он мог зашифровывать для нее файлы. Боб и другие лица, имеющие открытый ключ, не смогут расшифровывать их, поскольку они не имеют полной пары ключей.

**Импорт открытого ключа**

В рамках данной задачи производится загрузка открытого ключа. Этот ключ был создан при нажатии кнопки Export Public Key.

Данная задача воссоздает ситуацию, в которой Боб загружает открытый ключ Алисы, чтобы зашифровывать для нее файлы.

**Получение закрытого ключа**

В рамках этой задачи контейнеру ключа присваивается имя, соответствующее имени ключа, созданного при нажатии кнопки Create Keys. Контейнер ключа будет содержать полную пару ключей.

Данная задача воссоздает ситуацию, в которой Алиса использует свой закрытый ключ для расшифровки файлов, зашифрованных Бобом.

Структура папок при работе с приложением:

В папку c:\temp\encrypt сохраняются зашифрованные файлы и экспортируется открытый ключ.

В папку c:\temp\decrypt сохраняются расшифрованные файлы.

В папке c:\temp\doc находятся исходные файлы данных.

Перечень ссылок:

1. Службы криптографии

<http://msdn.microsoft.com/ruru/library/93bskf9z.aspx>

2. Создание криптографического приложения

<http://msdn.microsoft.com/ru-ru/library/bb397867.aspx>

**Ответы на вопросы**

**1) Какие симметричные алгоритмы шифрования Вы знаете?**

- Простая перестановка, перестановка по ключу, двойная перестановка

**2) Какие ассиметричные алгоритмы шифрования Вы знаете?**

- RSA, шифрование Диффи-хеллмана, шифрование Эль Гамаля.

**3) Основное назначение библиотеки System.Security.Cryptography?**

- В .NET классы в [System.Security.Cryptography](https://docs.microsoft.com/ru-ru/dotnet/api/system.security.cryptography) пространстве имен управляют множеством сведений о криптографии. При создании экземпляра одного из классов, реализующих алгоритмы шифрования, ключи создаются автоматически с целью удобства использования, а принятые по умолчанию значения свойств призваны обеспечить максимальную защищенность.

**4) Влияет ли размер ключа на криптостойкость алгоритма?**

**-** Да, влияет. Чем больше размер ключа, тем более надежна криптостойкость алгоритма.

**5) Назовите основные классы библиотеки System.Security.Cryptography?**

