Лабораторная работа №22

Разработка утилиты Менеджер паролей

1 Цель

1.1 Научиться выполнять шифрование и дешифрование данных в приложениях на C#, используя встроенные алгоритмы шифрования.

2 Литература

2.1 Фленов, М. Е. Библия C#. 4 изд / М. Е. Фленов. – Санкт-Петербург: БХВПетербург, 2019. – 512 с. – URL: https://ibooks.ru/bookshelf/366634/reading. – Режим доступа: только для зарегистрированных пользователей. – Текст : электронный. – п.14.2-14.4.

3 Подготовка к работе

3.1 Повторить теоретический материал (см. п.2).

3.2 Изучить описание лабораторной работы.

4 Основное оборудование

4.1 Персональный компьютер.

5 Задание

5.1 Отображение регистрационных данных пользователя

Разработать оконное приложение, отображающее данные из файла passwords.txt в ListView (столбцы: сайт/приложение, логин, пароль). Данные должны считываться из текстового файла passwords.txt (каждый набор значений – на отдельной строке, значения в наборе отделяются друг от друга точкой с запятой).

5.2 Добавление регистрационных данных пользователя

Реализовать в приложении из п.5.1 возможность добавления новых регистрационных данных. У пользователя должны запрашиваться сайт/приложение, логин, пароль. Введенные данные должны записываться в конец текстового файла passwords.txt при нажатии на кнопку «Добавить».

5.3 Генерация пароля

Реализовать в приложении из п.5.2 возможность генерации пароля заданной длины из символов английского алфавита при нажатии на кнопку «Сгенерировать пароль». Длина пароля указывается в поле ввода NumericUpDown. Сгенерированный пароль должен отображаться в поле ввода пароля.

5.4 Шифрование данных

Реализовать в приложении из п.5.2 шифрование пароля. Для шифрования использовать алгоритм AES (**смотреть пункт 9 Приложение**)

5.5 Дешифрование данных

Реализовать в приложении из п.5.4 отображение расшифрованных паролей.

6 Порядок выполнения работы

6.1 Запустить MS Visual Studio и создать оконное приложение C#.

6.2 Выполнить все задания из п.5 в одном решении.

6.3 Ответить на контрольные вопросы.

7 Содержание отчета

7.1 Титульный лист

7.2 Цель работы

7.3 Ответы на контрольные вопросы

7.4 Вывод

8 Контрольные вопросы

8.1 Что такое «шифрование»?

8.2 Что такое «дешифрование»?

8.3 Что такое «AES»?

8.4 Какими могут быть размеры ключа в алгоритме AES?

8.5 Какое пространство имен требуется подключить для применениястандартных алгоритмов шифрования?

9 Приложение

9.1 Для шифрования и записи данных в файл можно использовать:

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

```
//Ключ шифрования
byte[] key =
        {
            0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08,
            0x09, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16
        };
try
{
    // Создаем объект для работы с файлом TestData.txt
    using (FileStream fileStream = new("TestData.txt", FileMode.OpenOrCreate))
    {
        // Создаем экземпляр класса Aes для симметричного шифрования
        using (Aes aes = Aes.Create())
        {
            // Определяем ключ шифрования

            aes.Key = key;
            // Получаем вектор инициализации (IV)
            byte[] iv = aes.IV;
            // Записываем IV в файл
            fileStream.Write(iv, 0, iv.Length);

            // Создаем поток для шифрования данных
            using (CryptoStream cryptoStream = new(
                fileStream,
                aes.CreateEncryptor(),
                CryptoStreamMode.Write))
            {
                // Пишем зашифрованные данные в файл через StreamWriter
                using (StreamWriter encryptWriter = new(cryptoStream))
                {
                    encryptWriter.WriteLine("LALALALA213123321");
                }
            }
        }
    }

    Console.WriteLine("The file was encrypted.");
}
catch (Exception ex)
{
    Console.WriteLine($"The encryption failed. {ex}");
}
```

9.2 Для расшифровки данных из этого же файла можно использовать следующий код:

```
try
{
    using (FileStream fileStream = new("TestData.txt", FileMode.Open))
    {
        using (Aes aes = Aes.Create())
        {
            // Считывание IV из файла
            byte[] iv = new byte[aes.IV.Length];
            int numBytesToRead = aes.IV.Length;
            int numBytesRead = 0;
            while (numBytesToRead > 0)
            {
```

```
            int n = fileStream.Read(iv, numBytesRead, numBytesToRead);
            if (n == 0) break;

            numBytesRead += n;
            numBytesToRead -= n;
        }

        // Создание потока для дешифрования
        using (CryptoStream cryptoStream = new(
          fileStream,
          aes.CreateDecryptor(key, iv),
          CryptoStreamMode.Read))
        {
            // Чтение декодированных данных из файла
            using (StreamReader decryptReader = new(cryptoStream))
            {
                string decryptedMessage = await decryptReader.ReadToEndAsync();
                Console.WriteLine($"The decrypted original message: {decryptedMessage}");
            }
        }
    }
}
catch (Exception ex)
{
    Console.WriteLine($"The decryption failed. {ex}");
}
```