УДК 004.9

Пример шифрование данных с помощью DPAPI на языке C#

Николаев Сергей Валерьевич
Приамурский государственный университет им. Шолом-Алейхема
магистрант

Пасюков Александр Андреевич
Приамурский государственный университет им. Шолом-Алейхема
магистрант

Якимов Антон Сергеевич
Приамурский государственный университет им. Шолом-Алейхема
магистрант

Баженов Руслан Иванович
Приамурский государственный университет им. Шолом-Алейхема
к. п. н., доцент, зав. кафедрой информационных систем, математики и методик обучения

Аннотация
Информационная безопасность одна из острых проблем 21 века. С появлением интернета, развитием клиент-серверных приложения, облачных технологий, веб-приложений обмен информацией и ее доступность достигли высокого, небывалого, до этого дня, уровня. Но есть данные, которые требуют строгой конфиденциальности, примером могут служить персональные данные, доступ к которым третьим лицам должен быть ограничен. Шифрование – один из способов решения данной задачи. Сейчас большие объемы информации хранятся в СУБД. Но что если нам необходимо зашифровать небольшой объем информации? Примером может быть строка подключения, в которой содержится информация для доступа к БД. В данной статье с помощью разработанного, небольшого, консольного приложения, будет продемонстрировано использование шифрования данных с помощью DPAPI (Data Protection API) – криптографический интерфейс программирования приложений в ОС семейства Windows, обеспечивающий защиту (конфиденциальность) данных путём их шифрования. Проект реализован на языке программирования C#. Исходный код проекта можно посмотреть на электронном ресурсе GitHub: https://github.com/single1992/DPAPI-Example.
Ключевые слова: Информационные технологии, языки программирования, информационная безопасность, DPAPI, криптография, шифрование, C#.

Example data encryption using DPAPI in C

Nikolaev Sergey Valerievich
Sholom-Aleichem Priamursky State University
Master student

Pasyukov Alexandr Andreevich
Sholom-Aleichem Priamursky State University
Master student

Yakimov Anton Sergeevich
Sholom-Aleichem Priamursky State University
Master student

Bazhenov Ruslan Ivanovich
Sholom-Aleichem Priamursky State University
Candidate of pedagogical sciences, associate professor, Head of the Department of Information Systems, Mathematics and teaching methods

Abstract
Information security is one of the most acute problems of the 21st century. With the advent of the Internet, the development of client-server applications, cloud technologies, web applications, information exchange and its availability have reached a high, unprecedented level up to this day. But there are data that require strict confidentiality, for example, personal data, access to which third parties should be limited. Encryption is one of the ways to solve this problem. Now a lot of information is stored in the DBMS. But what if we need to encrypt a small amount of information? An example might be a connection string that contains information for accessing the database. In this article, using a developed, small, console application, we will demonstrate the use of data encryption with the help of DPAPI (Data Protection API) - a cryptographic application programming interface in the Windows family of operating systems that protects (confidentials) data by encrypting them. The project is implemented in the C # programming language. The source code for the project can be viewed on the GitHub electronic resource: https://github.com/single1992/DPAPI-Example.
Keywords: Information technologies, programming languages, information security, DPAPI, cryptography, encryption, C #.

Со времен Древнего Египта, люди использовали криптографию, для защиты необходимой информации. В век информационных технологий данная сфера актуальна, как никогда. Огромные суммы тратят компании и государства, для разработки и исследования методов шифрования. Трудно представить современный мессенджер, без криптографической защиты личной переписки пользователя. Огромное количество серверов, где хранятся персональные данные, требующие конфиденциальности. Такая

сфера, как информационная безопасность развивается и будет развиваться, так как необходимость защиты определенных данных не подвергается сомнениям.

О криптографии пишут Б. Шнайер в книге «Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си» [10], В. Мао в своем труде «Современная криптография» [4], Алферов А.П., Зубов А.Ю., Кузьмин А.С., Черемушкин А.В также написали об основах криптографии [1], о службах криптографии .Net Framework-а можно прочесть на их официальном сайте [7].

В данный момент для хранения большого количества информации используют системы управления базами данных (СУБД) со встроенными механизмами защиты данных. Но что если объем информации невелик, и нет смысла разрабатывать базу данных? Что если, данная информация мала, но требует защиты в виде шифрования и недоступности для злоумышленников?

Смоделируем ситуацию. Допустим, у нас есть программа, цель которой является рассылать оповещения пользователю через мессенджер Telegram. Подробности о данной программе можно прочитать в статье [6]. Данная программа хранит ряд данных, требующих защиты: учетные записи почты, идентификационный токен чат-бота, идентификационный номер пользователя мессенджера Telegram. В данном проекте все эти данные «вшиты в исходный код», но при дальнейшей доработке, эти данные придется вынести, так как их необходимо предоставить пользователю для изменений и определенным образом состояние этих параметров. Также стоит учесть, что данные из исходного кода C# также можно извлечь, используя Дизассемблер IL. Еще один фактор – это тип приложения. Его с легкостью можно перенести с одного компьютера на другой, вместе со всеми данными, предыдущего пользователя.

Сформулируем задачу. Необходимо, чтобы для каждого пользователя шифрование происходило индивидуально, чтоб даже при переносе программы с одного компьютера на другой, данные были защищены.

В качестве решения такой задачи можно использовать DPAPI (Data Protection API). DPAPI – это криптографический интерфейс программирования приложений в ОС семейства Windows, обеспечивающий защиту (конфиденциальность) данных путём их шифрования. Другими словами, с помощью данной технологии можно обеспечить доступ к данным только конкретному пользователю ОС Windows, т.е. для каждого пользователя предоставляется свой ключ шифрования. Таким образом решается проблема не только с переносом программы с компьютера на компьютер, но и проблема защиты данных от других пользователей операционной системы Windows на данном компьютере.

Продемонстрируем использование данной технологии. Разработаем консольный проект на языке программирования C#. О данном языке пишут много авторов: Эндрю Троелсен [8], Герберт Шилдт [9], Джеффри Рихтер [5]. Цель данного приложения разработать небольшую программу для

шифрования данных. Исходный код данного проекта можно посмотреть на электронном ресурсе GitHub [2].

В основе лежит класс ProtectedData, располагающийся в пространстве имен System.Security.Cryptography. Его описание описано в документации Microsoft [3]. Ниже, на рисунке 1 приведен класс, включающий в себя использование класса ProtectedData.

```
public static class Security
{
    ссылка: 1 | Sergey, 59 мин назад | Автор: 1, изменение: 1
    public static byte[] Protect(string strData)
    {
        try
        {
            var data = Encoding.UTF8.GetBytes(strData);
            return ProtectedData.Protect(data, null, DataProtectionScope.CurrentUser);
        }
        catch (CryptographicException e)
        {
            Console.WriteLine("Данные не были зашифрованы.");
            Console.WriteLine(e.ToString());
            return null;
        }
    }

    ссылка: 1 | Sergey, 56 мин назад | Автор: 1, изменение: 1
    public static string Unprotect(byte[] data)
    {
        try
        {
            var unprotectedData = ProtectedData.Unprotect(data, null, DataProtectionScope.CurrentUser);
            return Encoding.UTF8.GetString(unprotectedData);
        }
        catch (CryptographicException e)
        {
            Console.WriteLine("Данные не были расшифрованы.");
            Console.WriteLine(e.ToString());
            return null;
        }
    }
}
```

Рисунок 1 – Класс Security, использующий класс ProtectedData

Рассмотрим консольный интерфейс программы (Рисунок 2). Для работы предоставлено шесть команд:

1. чтение данных без использования DPAPI;
2. запись данных без использования DPAPI;
3. чтение данных с использованием DPAPI;
4. запись данных с использованием DPAPI;
5. чтение, без использования DPAPI, данных, которые зашифрованы с помощью DPAPI.
6. выход из приложения.

Команды 1-2 работают с файлом, к которому не применяется шифрование DPAPI.

Команды 3-4 работают с файлом, данные которого шифруются.

Команда 5 - попытка считать зашифрованный файл командой, предназначенной для чтения обычного файла.

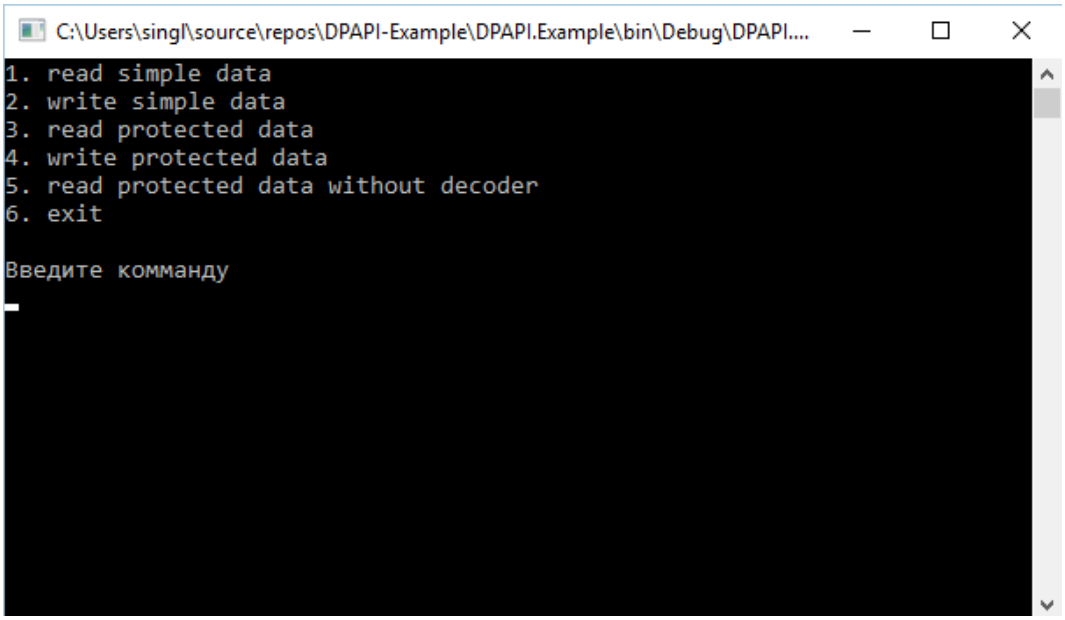

Рисунок 2 – Интерфейс демонстрационного консольного приложения

Проверим работу первых двух пар команд чтения/записи. На рисунке 3-4 можно увидеть, что запись и чтение файлов прошло успешно.

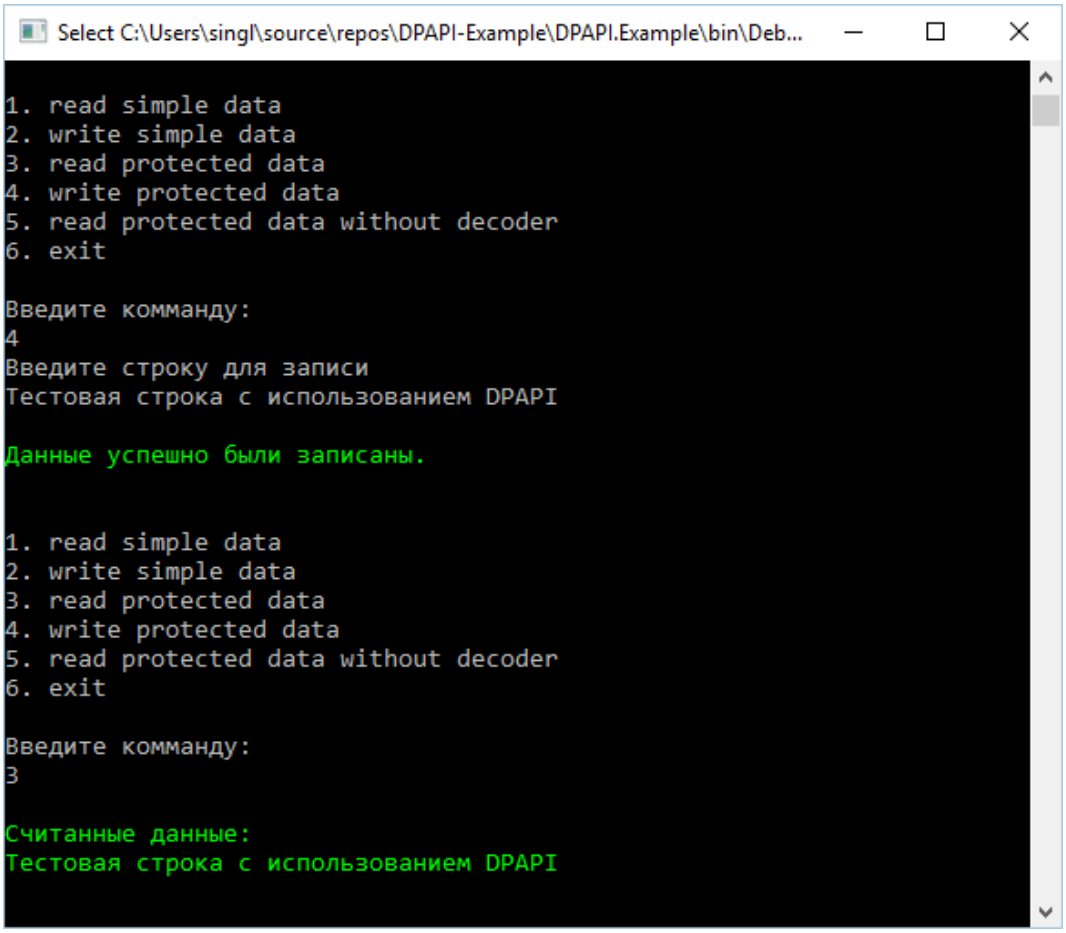

Рисунок 3 – Вывод консоли при использовании записи, а затем чтения строки без использования DPAPI

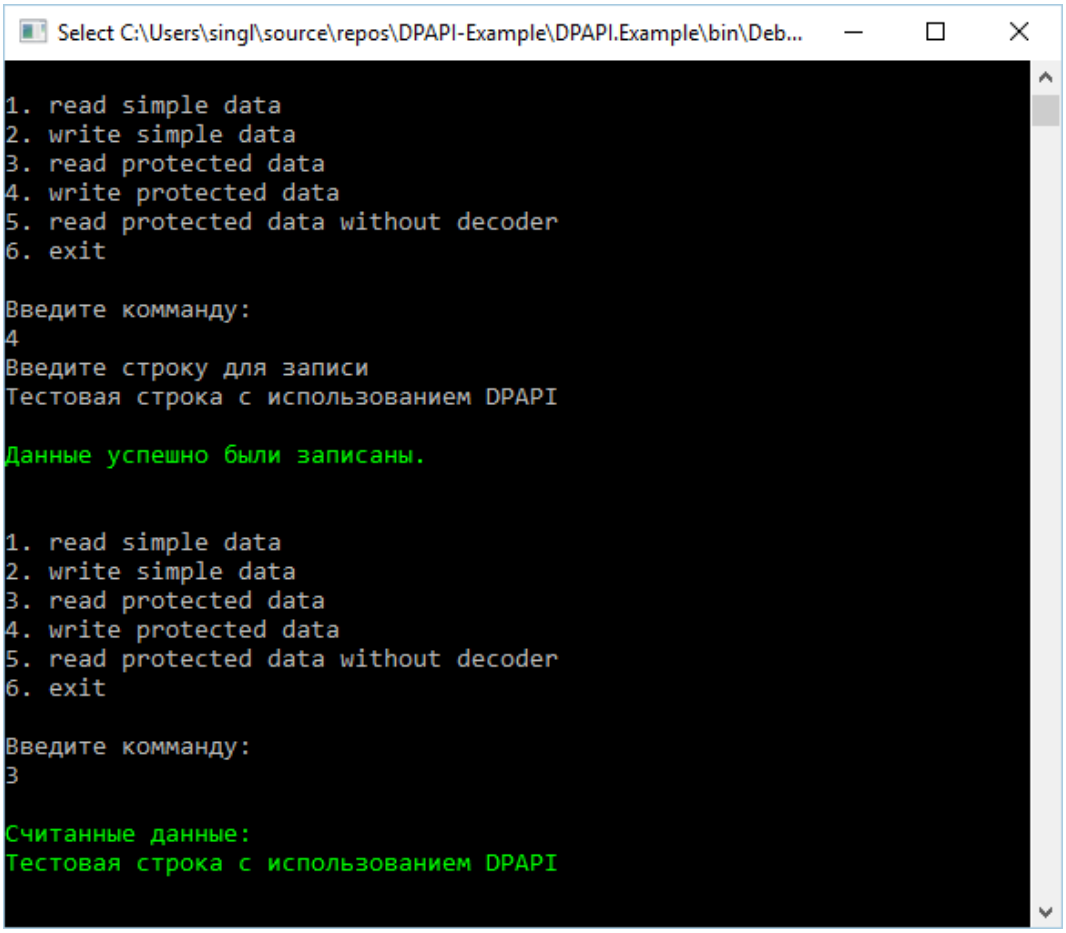

Рисунок 4 – Вывод консоли при использовании записи, а затем чтения строки с использованием DPAPI

Сравнивая результаты никаких видимых отличий нет. И там, и там вводится, а затем выводится текстовое сообщение. Для того чтобы увидеть различия, воспользуемся командой под номером 5 (рисунок 5).

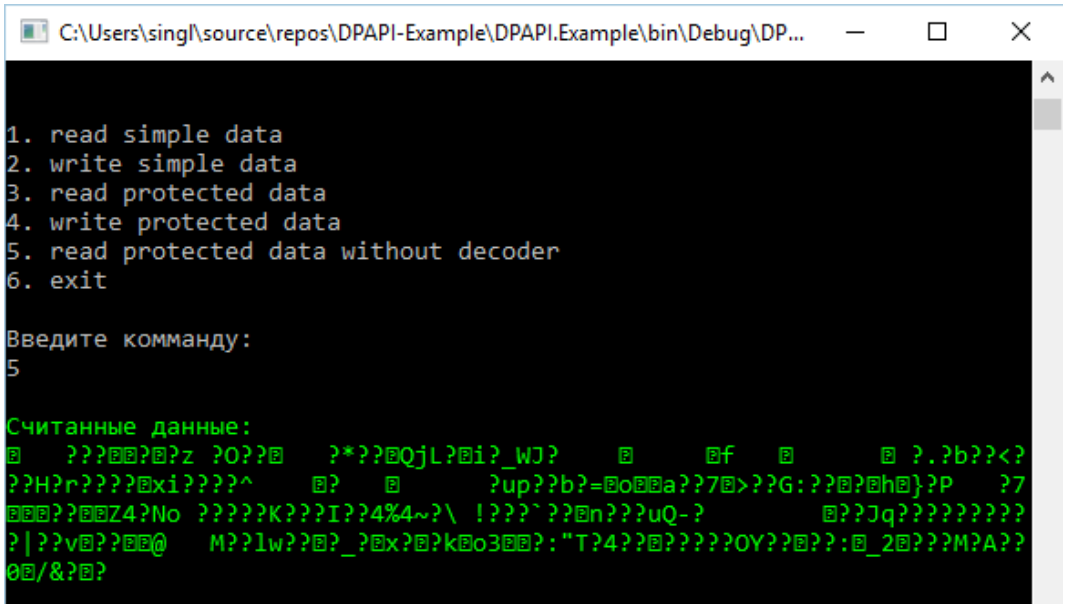

Рисунок 5 – Считывание без использования DPAPI зашифрованной ранее строки с помощью DPAPI

Данная команда использует для чтения алгоритм 1 команды (без использования DPAPI), для чтения файла, записанного с помощью 4 команды, которая для записи использует DPAPI. Результат можно увидеть на рисунке 5. Как видим, строка имеет нечитабельный вид, хотя изначально там записано «Тестовая строка с использованием DPAPI».

Следующий этап будет попытка запуска команд от имени другого пользователя. Для этого необходимо сменить пользователя в операционной системе Windows или запустить программу с другого компьютера. У нас уже имеется записанная ранее информация, поэтому достаточно протестировать команды 1 и 3. На рисунке 6, продемонстрирован результат выполнения команд 1 и 3.

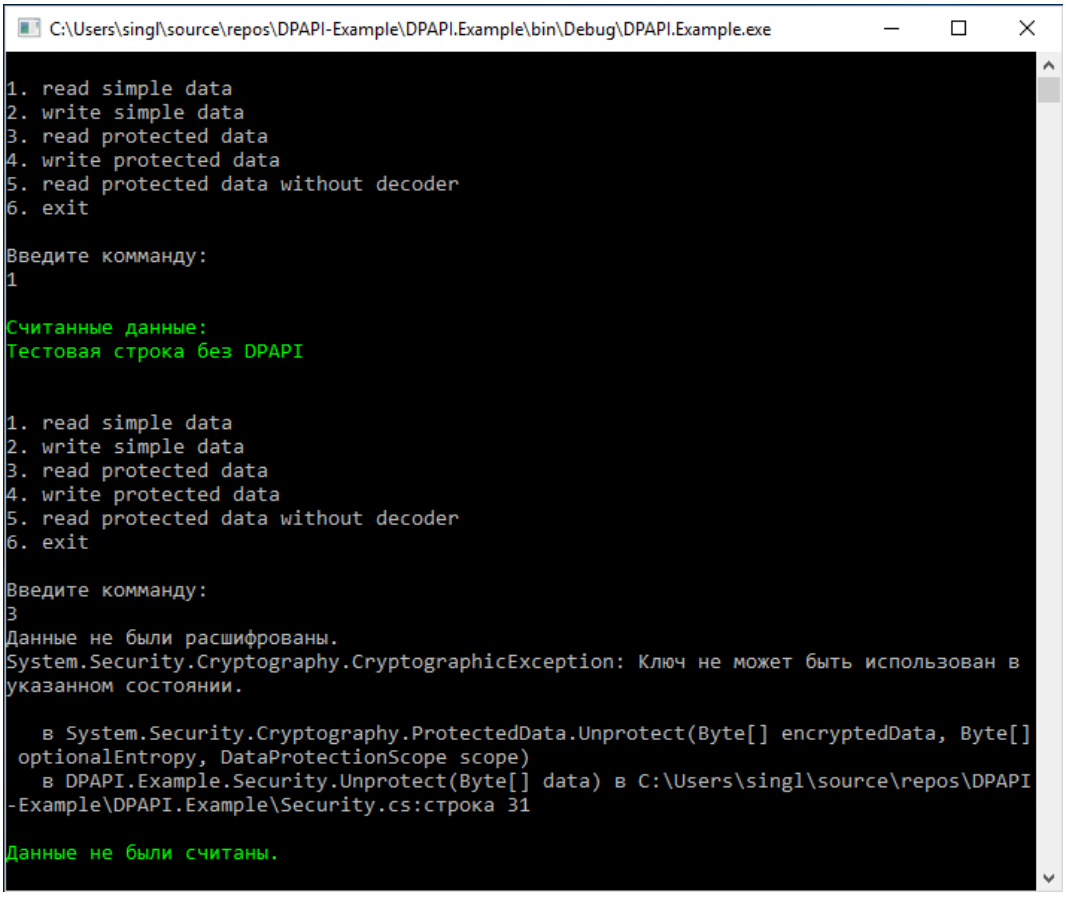

Рисунок 6 – Последовательный запуск считывания файла с помощью команд 1 и 3

Как видим незашифрованный файл считался без проблем, но а попытка считывания зашифрованного файл от имени другой учетной записи завершилось ошибкой «ключ не может быть использован в данном состоянии». Если же заново провести цикл запись/чтение с помощью команд 4 и 3, то результат будет идентичен результату на рисунке 4.

Таким образом, поставленная цель достигнута, данные защищены и прочитать их может только пользователь, от имени которого записан файл. Для более подробного изучения данного демонстрационного приложения рекомендуется ознакомиться с исходным кодом [2]. Также стоит учитывать,

что данное приложение написано исключительно для демонстрации и не совсем удобна для использования в повседневной жизни. К примеру, в коде «жестко» прописаны пути к файлам, а для большинства пользователей консольный интерфейс не совсем дружелюбен.

Библиографический список

1. Алферов А.П., Зубов А.Ю., Кузьмин А.С., Черемушкин А.В. Основы криптографии. М.: Гелиос АРВ, 2001. 479 с.
2. Исходный код разработанного проекта // github.com URL: https://github.com/single1992/DPAPI-Example (дата обращения: 28.01.2018).
3. Класс ProtectedData // msdn.microsoft.com URL: https://msdn.microsoft.com/ru-ru/library/92f9ye3s(v=vs.110).aspx (дата обращения: 28.01.2018).
4. Мао В. Современная криптография. Теория и практика. М.: Вильямс, 2005. 763 с.
5. Рихтер Д. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.0 на языке C#. М.: Питер, 2013. 928 с.
6. Николаев С.В., Пасюков П.А., Якимов А.С., Баженов Р.И. Пример использования информационных технологий для решения повседневных задач // Постулат. 2017. №12. URL: http://e-postulat.ru/index.php/Postulat/article/view/934/960
7. Службы криптографии // msdn.microsoft.com URL: https://msdn.microsoft.com/ru-ru/library/92f9ye3s(v=vs.110).aspx (дата обращения: 28.01.2018).
8. Троелсен Э. Язык программирования C# 5.0 и платформа .NET 4.5 М.: Вильямс, 2015. 486 с.
9. Шилдт Г. C# 4.0: полное руководство. Пер. с англ.. М.: Вильямс, 2011. 1056 с.
10. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. М.: Триумф, 2003. 806 с.