УДК 004.42

Особенности использования алгоритмов DES и 3DES в языке C#

Ибрагимов Тимур Рашидович
Смоленский государственный университет
магистрант

Козлов Сергей Валерьевич
Смоленский государственный университет
кандидат педагогических наук, доцент, доцент кафедры информатики

Аннотация
В статье рассмотрены особенности использования алгоритмов DES и 3DES в языке C#. Автором объясняется, почему программная реализация криптографических алгоритмов является наиболее эффективной. На примере приложения «Симметричная криптография» выявлены отличительные признаки и преимущества использования данных криптографических алгоритмов в языке C#. Подробно рассмотрены инструменты языка, позволяющие выполнять настройки шифрования и обработку результатов. Актуальность статьи состоит в необходимости простых в использовании и гибких методов для обеспечения надежной защиты информации.
Ключевые слова: информатика, программирование, информационно-коммуникационные технологии, информация, криптография.

The features of the algorithms for DES and 3DES in C#

Ibragimov Timur Rashidovich
Smolensk State University
Undergraduate

Kozlov Sergey Valerevich
Smolensk State University
Candidate of pedagogical sciences, associate professor, associate Professor of the Department of Informatics

Abstract
In this article is considered and explained features of using algorithms such as DES and 3DES in C# programming language as well as the reason why program realization of cryptographic algorithms is the most effective way to secure the data. Using the example program “Symmetric-key algorithm” there some features and advantages of using C# were identified and examined such as instruments of programming language to set up encryption settings and results analyzing. The relevance of this article lies in need of simple and adjustable ways and methods to provide high-level security of information.

Keywords: informatics, programming, information and communication technologies, information, cryptography.

Наше время – это время информационных технологий [1, 2, 3]. Сегодня информация играет первостепенную роль. Следовательно, наиболее важными становятся вопросы ее защиты. Эту задачу с древнейших времен решает криптография [4].

Термин «криптография» происходит от двух греческих слов: κρυπτός – скрытый и γράφω – пишу, и означает тайнопись. Этот термин ввел английский математик XVII века Д. Валлис (*John Wallis).*

Криптография – это наука о методах изменения информации с *целью* обеспечения конфиденциальности (невозможности прочтения информации незаконным лицам), целостности (обеспечение того, что при пересылке или чтении информация не была изменена лицом, не имеющим на это права) и аутентификации (проверка подлинности сторон при обмене информацией, автора документов, получателя и т.д.)

Поскольку сегодня информация хранится и обрабатывается в цифровом (двоичном) формате, то на данный момент программная реализация криптографических методов является наиболее эффективной и используемой [5]. Современные языки программирования предоставляют широкий выбор инструментов, позволяющих реализовывать и использовать мощные алгоритмы шифрования [6].

Алгоритм DES – является современным, быстрым алгоритмом шифрования [7, 8]. Однако обладает относительно низкой криптостойкостью. Поэтому он находит активное применение в простых приложениях, чувствительных к быстродействию, но не требующих серьезного уровня защиты. Алгоритм DES – это симметричный, блочный шифр, использующий ключ длиной 56 бит.

Алгоритм Triple DES является усовершенствованием алгоритма DES [9]. 3DES был создан с целью устранения главной проблемы своего предшественника – низкой криптостойкойсти. Triple DES в три раза медленнее, но во много раз лучше противостоит криптоанализу.

Одним из самых наглядных способов демонстрации особенностей реализации DES и 3DES на языке C# является приложение типа Windows Form [10]. В качестве примера можно привести приложение «Симметричная криптография» (рис. 1).

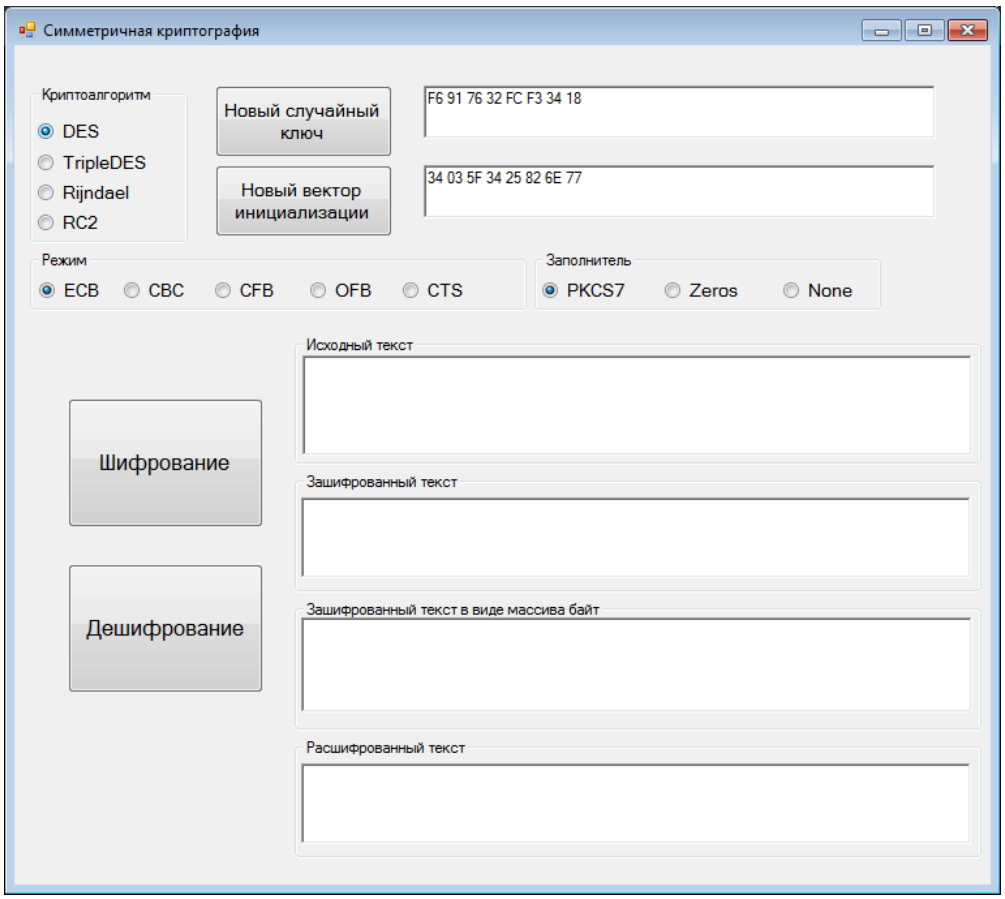

Рисунок 1 – Форма приложения «Симметричная криптография»

Данное приложение реализовано в интегрированной среде разработки MS Visual Studio при использовании бесплатной студенческой подписки и представляет собой проект типа Windows Form Application.

Реализация симметричных криптографических алгоритмов на языке C# (в среде .NET) делает работу программистов намного проще. Используя эти технологии, нет необходимости изучать тонкости работы каждого алгоритма, т.к. все современные алгоритмы шифрования реализованы и инкапсулированы в наборе классов пространства имен System.Security.Cryptography. Необходимые для использования DES и 3DES классы являются производными от класса симметричной криптографии, который расположен в библиотеке System.Security.Cryptography.SymmetricAlgorithm.

В тоже время можно на основании имеющихся классов криптографии и инструментов работы с ними разработать собственные классы криптографии множественного наследования для обработки и хранения данных в информационных системах [11, 12, 13].

Первым шагом при разработке приложения необходимо подключить пространство имен System.Security.Cryptography. Для этого в программный код следует включить команду using System.Security.Cryptography.

Слева на форме находится набор радиокнопок (radioButton), объединенных в группу при помощи элемента groupBox. Данная конструкция позволяет легко и удобно выбирать необходимый алгоритм (рис. 2).

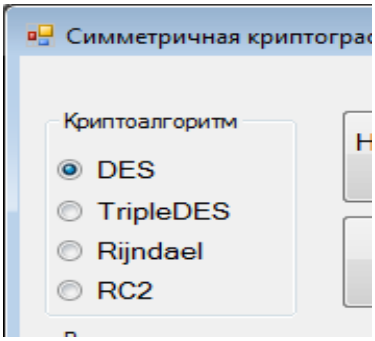

Рисунок 2 – Панель выбора используемого криптоалгоритма

Ниже расположена панель выбора режима шифрования. Для DES и 3DES рекомендуется использовать режимы ECB, CBC, CFB, OFB и CTS (рис. 3).

Рисунок 3 – панель выбора режима шифрования

При старте приложения генерируется случайный ключ, используемый для шифрования и дешифрования. Этот ключ подходит для DES, однако 3DES требует ключ большей длины, поэтому перед его использованием нужно сгенерировать новый ключ, выбрав необходимый алгоритм и нажав на соответствующую кнопку. Генерация нового ключа осуществляется вызовом метода GenKey().

```
private void GenKey()
{
   //Генерация нового случайного ключа
   SymmetricAlgorithm sa =CreateSymmetricAlgorithm();
   sa.GenerateKey();
   Key = sa.Key;
   //Обслуживание пользовательского интерфейса
   UpdateKeyTextBox();
   ClearOutputFields();
}
```

Метод создает объект типа SymmetricAlgorithm вызовом метода CreateSymmetricAlgorithm(), который ориентируясь на активный radioButton на форме, инициализирует соответствующий объект. Это осуществляется путем простых проверок.

```
if (radioButtonDES.Checked == true)
   return DES.Create();
if (radioButtonTripleDES.Checked == true)
   return TripleDES.Create();
```

Сгенерированный ключ вставляется в первый textBox посредством функции UpdateKeyTextBox(). Функция ClearOutputFields() очищает поля

вывода. Далее, когда ключ сгенерирован, необходимо ввести в поле «исходный текст» нужную информацию и нажать на кнопку «Шифрование». При нажатии на кнопку, происходит обработка введенной информации и шифрование. На первом шаге создается объект типа SymmetricAlgorithm командой:

```
SymmetricAlgorithm symmetricAlgorithm =  CreateSymmetricAlgorithm();
```

Затем заполняются его свойства:

```
symmetricAlgorithm.Key = Key;
symmetricAlgorithm.Mode = Mode;
symmetricAlgorithm.Padding = Padding;
```

Свойство Padding используется для выбора режима заполнения. В данном приложении используются режимы PKCS7, Zeros и None, которые также можно выбрать с помощью радиокнопок на специальной панели формы. Заполнение нужно для тех случаев, когда при разбиении на блоки открытого текста в последнем блоке недостает символов.

Далее создается поток шифрования. Для этого служит команда MemoryStream ms = new MemoryStream();

Класс System.Security.Cryptography.CryptoStream соединяет исходный поток данных с выбранным криптоалгоритмом.

```
CryptoStream cs =
new CryptoStream (ms, symmetricAlgorithm.CreateEncryptor(), CryptoStreamMode.Write);
```

Происходит записать байтов открытого текста в поток шифрования.

```
byte [] plainbytes =  Encoding.UTF8.GetBytes(textPlaintext.Text);
cs.Write(plainbytes, 0, plainbytes.Length);
cipherbytes = ms.ToArray();
```

После того, как потоки отработали их необходимо закрыть.

```
cs.Close();
ms.Close();
```

Зашифрованный текст отображается в виде символов и выводится в элемент textBox формы с меткой «Зашифрованный текст».

```
textCiphertext.Text =  Encoding.UTF8.GetString(cipherbytes) ,
```

Он же отображается также в виде байт и выводится в поле «Зашифрованный текст в виде массива байт». Результат шифрования представлен на рисунке 4.

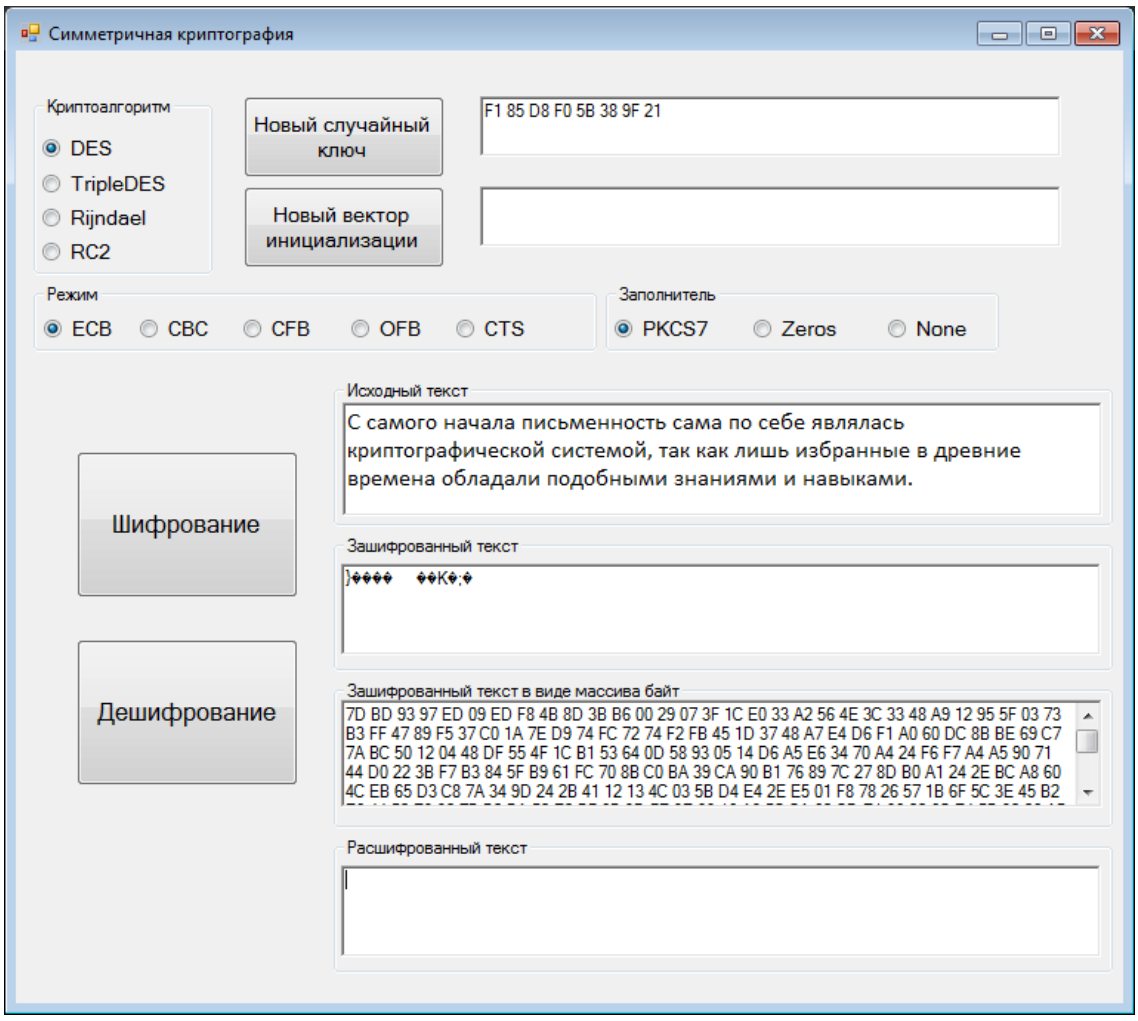

Рисунок 4 – Результат шифрования данных

Дешифрование происходит аналогичным образом. При нажатии на кнопку «дешифрование» происходит создание нового объекта SymmetricAlgorithm.

Аналогично заполняются свойства объекта. Задается поток шифрования.

```
MemoryStream ms = new MemoryStream(cipherbytes);
CryptoStream cs = new CryptoStream(ms, sa.CreateDecryptor(), ryptoStreamMode.Read);
```

Потом считываются шифрованные байты из потока и выполняется дешифрование.

```
byte [] plainbytes =  new Byte[cipherbytes.Length];
cs.Read(plainbytes, 0, cipherbytes.Length);
```

Потоки закрываются.

```
cs.Close();
ms.Close();
```

Восстановленный текст отображается в поле «Расшифрованный текст».

```
textRecoveredPlaintext.Text = Encoding.UTF8.GetString(plainbytes);
```

Результат дешифрования изображен на рисунке 5.

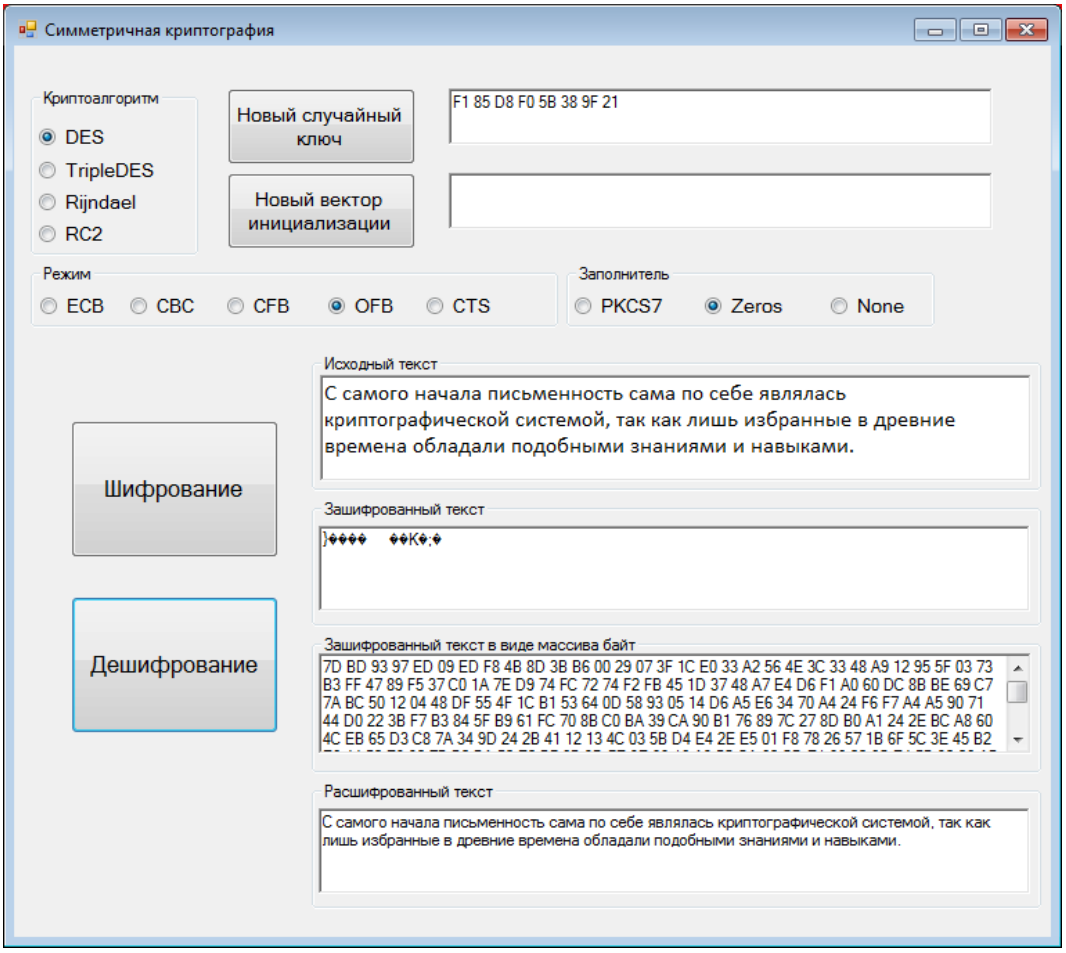

Рисунок 5 – Результат дешифрования

Подводя итоги, можно говорить о следующих преимуществах применения криптографических алгоритмов, при использовании языка C#:

1. Простота и легкость встраивания шифрования в приложения.
2. Объектно-ориентированный подход [14, 15], который обеспечивает инкапсуляцию и понятный, имеющий подробную документацию, программный интерфейс.

В описанном выше приложении «Симметричная криптография» перечисленные преимущества ярко продемонстрированы. Пространство имен System.Security.Cryptography и инструменты языка позволяют использовать необходимые, сложные криптографические алгоритмы; не вдаваясь в детали и в сложность их реализации, выполнять подробные настройки шифрования (автоматическая генерация ключа, режим шифрования, заполнитель и т.д.) и получать результат в требуемом виде.

Библиографический список

1. Размахнина А.Н., Баженов Р.И. О применении экспертных систем в различных областях // Постулат. 2017. № 1 (15). С. 38.
2. Козлов С.В. Применение соответствия Галуа для анализа данных в информационных системах // Траектория науки. 2016. Т. 2. № 3 (8). С. 18.
3. Журавлёва У.С., Баженов Р.И. Нейронные сети в Scilab // Постулат. 2017. № 1 (15). С. 25.
4. Яблонский С. В. Введение в дискретную математику М.: Высшая школа, 2008. 384 с.
5. Логачев О. А., Сальников А. А., Ященко В. В. Булевы функции в теории кодирования и криптологии. М.: МЦНМО, 2004. 470 с.
6. Козлов С. В. Использование соответствия Галуа как инварианта отбора контента при проектировании информационных систем // Современные информационные технологии и ИТ-образование. 2015. Т. 2. № 11. С. 220-225.
7. Гомес Ж. Математики, шпионы и хакеры. Кодирование и криптография // Мир Математики. М.: Де Агостини, 2014. Т.34. 144 с.
8. Мао В. Современная криптография: Теория и практика. М.: Вильямс, 2005. 247 с.
9. Гатчин Ю. А. Основы криптографических алгоритмов. Учебное пособие / Гатчин Ю. А., Коробейников А. Г. – СПб: ГИТМО (ТУ), 2002. 29 с.
10. Суин И. А., Козлов С. В. Особенности реализации графических решений при разработке пользовательских приложений в объектно-ориентированных средах программирования // Международный студенческий научный вестник. 2017. № 5. С. 53.
11. Козлов С. В. Интерпретация инвариантов теории графов в контексте применения соответствия Галуа при создании и сопровождении информационных систем // International Journal of Open Information Technologies. 2016. Т. 4. № 7. С. 38-44.
12. Козлов С.В. Функциональные назначения и возможности информационно-образовательного ресурса «ADVANCED TESTER» // Горизонты науки. 2011. № 2. С. 9.
13. Козлов С.В. Теория графов и соответствия Галуа как инструменты проектирования информационных систем // NovaInfo.Ru. 2016. Т. 3. № 48. С. 144-149.
14. Гатченко Н. А., Исаев А. С., Яковлев А. Д Криптографическая защита информации. СПб: НИУ ИТМО, 2012. 142 с.
15. Козлов С.В. Анализ результатов экспериментальной деятельности по изучению основ объектно-ориентированного программирования в школьном курсе информатики // Современные научные исследования и инновации. 2014. № 6-3 (38). С. 16.