**­Министерство образования и науки РФ**

**БРЯНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Кафедра «Информатика и программное обеспечение»**

**программная Реализация режима сцепления блоков шифра «Cast-256»**

**КУРСОВАЯ РАБОТА**

по дисциплине «Методы и средства защиты информации»

**Студент группы 10ПО2**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Калевко В.В.

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2014 г.

**Руководитель**

\_\_\_\_\_\_\_\_\_\_к.т.н. доц. Дергачев К.В.

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2014 г.

**БРЯНСК 2014**

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc402307976)

[1. АНАЛИТИЧЕСКАЯ ЧАСТЬ 4](#_Toc402307977)

[1.1. CAST-256 4](#_Toc402307978)

[1.2. IDEA 8](#_Toc402307979)

[1.3. ГОСТ 28147-89 9](#_Toc402307980)

[1.4. Режимы работы симметричных шифров 10](#_Toc402307981)

[1.4.1 Режим электронной кодовой таблицы (ECB) 10](#_Toc402307982)

[1.4.2 Режим сцепления блоков (CBC) 11](#_Toc402307983)

[1.4.3 Режим обратной связи по шифр-тексту (CFB) 12](#_Toc402307984)

[1.4.4 Режим обратной связи по выходу (OFB) 12](#_Toc402307985)

[1.5. Вывод 13](#_Toc402307986)

[2. КОНСТРУКТОРСКАЯ ЧАСТЬ 14](#_Toc402307987)

[2.1 Программная реализация CAST-256 14](#_Toc402307988)

[2.2 Описание классов и основных методов 15](#_Toc402307989)

[3. ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ 17](#_Toc402307990)

[3.1 Руководство пользователя 17](#_Toc402307991)

[3.2 Тестирование работы шифра 18](#_Toc402307992)

[СПИСОК ЛИТЕРАТУРЫ 22](#_Toc402307993)

# ВВЕДЕНИЕ

 CAST-256 – блочный алгоритм симметричного шифрования на основе сети Фейстеля, опубликованный в июне 1998 года в качестве кандидата на участие в конкурсе AES. Алгоритм разработан специалистами канадской компании Entrust Technologies Карлайслом Адамсом (Carlisle Adams) и Стаффордом Таваресом (Starrord Tavares). Они утверждают, что название обусловлено ходом разработки и должно напоминать о вероятностном характере процесса, а не об инициалах авторов.

CAST-256 построен из тех же элементов, что и CAST-128, включая S-боксы, но размер блока увеличен вдвое и равен 128 битам. Это влияет на диффузионные свойства и защиту шифра.

CAST имеет знакомую структуру. Алгоритм использует 6 S-блоков с 8-битовым входом и 32-битовым выходом. Работа этих S-блоков сложна и зависит от реализации.

# АНАЛИТИЧЕСКАЯ ЧАСТЬ

## 1.1. CAST-256

Алгоритм CAST-256 шифрует информацию 128-битными блоками и использует несколько фиксированных размеров ключа шифрования: 128, 160, 192, 224 или 256 бит.

В общем случае алгоритм состоит из двух этапов — расширение ключа и шифрация/дешифрация исходных данных (Рис.1).

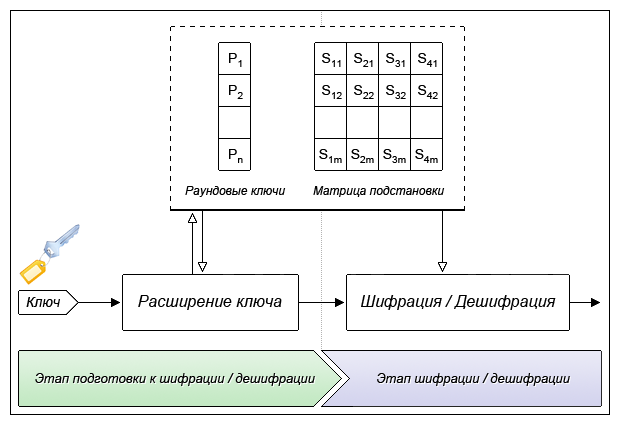
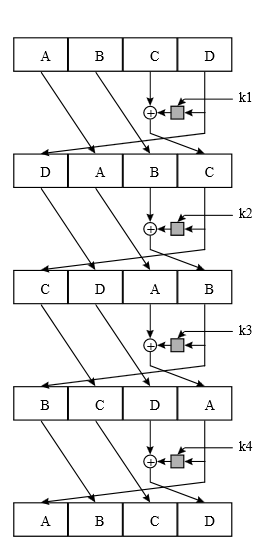


Рис. 1.1. Общий алгоритм

Шифрация данных, а также расширение раундовых ключей и подстановки, происходит через использование сети Фейстеля (Рис.1.2).



**Рис. 1.2. Измененная сеть Фейстеля**

При шифровании 32-битного блока (ABCD) по 6 раз выполняются операции Forward Quad-Round (рис 1.3) и Reverse Quad-Round (рис 1.4).

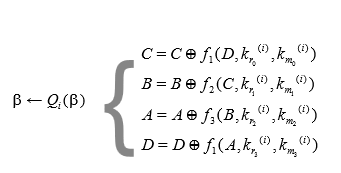


Рис. 1.3 Forward Quad-Round

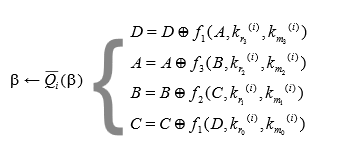
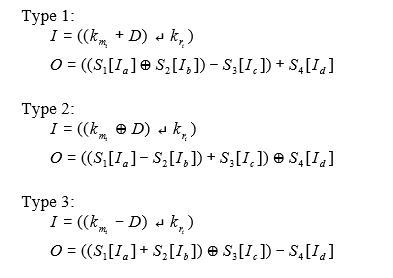


Рис. 1.4 Reverse Quad-Round

При этом используются 3 раундовые функции (рис. 1.5):

*Function 1:*



*Function 3:*

*Function 2:*

Рис. 1.5. Раундовые функции CAST-256

Здесь D – входные данные, байты блока данных, – i-й S-box, О – результат функции, «+» и «–» сложение и вычитание по модулю , - исключающее ИЛИ, «←» цикличный битовый сдвиг влево.

Процедура расширения ключа сложнее чем само шифрование данных. Пусть массив ключей k = (ABCDEFGH) представляющий собой 256-битный блок, где фрагменты A, B,…,H каждый длиной по 32 бита. Пусть «k ← wi(k)», где w(⋅) есть функция расширения и представима в следующем виде:

*G*=*G*⊕*f*1(*H*,*tr*0,*tm*0)

*F*=*F*⊕*f*2(*G*,*tr*1,*tm*1)

*E*=*E*⊕*f*3(*F*,*tr*2,*tm*2)

*D*=*D*⊕*f*1(*E*,*tr*3,*tm*3)

*C*=*C*⊕*f*2(*D*,*tr*4,*tm*4)

*B*=*B*⊕*f*3(*C*,*tr*5,*tm*5)

*A*=*A*⊕*f*1(*B*,*tr*6,*tm*6)

*H*=*H*⊕*f*2(*A*,*tr*7,*tm*7)

В результате 4 преобразований *k*(*i*)*r*←*k* по 5 младших бита образуются подключи сдвига:

*k*(*i*)*r*0=5*LSB*(*A*),*k*(*i*)*r*1=5*LSB*(*C*),*k*(*i*)*r*2=5*LSB*(*E*),*k*(*i*)*r*3=5*LSB*(*G*)

где 5LSB(x)означает образование 5 младших бит в результате операции x. Аналогично *k*(*i*)*m*←*k* образуются маскирующие подключи:

*k*(*i*)*m*0=*H*,*k*(*i*)*m*1=*F*,*k*(*i*)*m*2=*D*,*k*(*i*)*m*3=*B*

**Реализация процедуры расширения ключа**

В каждом раунде функции расширения ключа используются по 8 дополнительных переменных *T*(*i*)*mj* и *T*(*i*)*rj*.

***1. Инициализация***

*Cm*=2302√=5*A*82799916

*Mm*=2303√=6*ED*9*EBA*116

*Cr*=19

*Mr*=17

Пусть *T*(*i*)*mj* = Tmij, *T*(*i*)*rj* = Trij.

for(i=0; i<24; i++)

for(j=0; j<8; j++){

Tmij = Cm

Cm = (Cm + Mm)mod2^32

Trij = Cr

Cr = (Cr + Mm)mod32

}

***2. Ключ расширения:***

k = ABCDEFGH = 256 бит исходного ключа K. Пусть «*k*←*w*2*i*(*k*)»  ∼  «*W*2*i*(*k*)», «*k*←*w*2*i*+1(*k*)»  ∼ «*W*2*i*+1(*k*)», «*kir*←*k*»  ∼ «*kr*», «*kim*←*k*» ∼ «*km*»

for(j=0; j<12; j++){

W2i(k)

W2i+1(k)

kr

km

}

## 1.2. IDEA

Симметричный блочный шифр с длиной ключа 128 бит и размером блока 64 бита.

Процесс шифрования состоит из восьми одинаковых раундов шифрования и одного выходного преобразования. Исходный незашифрованный текст делится на блоки по 64 бита. Каждый такой блок делится на четыре подблока по 16 бит каждый. В каждом раунде используются свои подключи согласно таблице подключей. Над 16-битными подключами и подблоками незашифрованного текста производятся следующие операции:

* умножение по модулю 216 + 1 = 65537, причем вместо нуля используется 216
* сложение по модулю 216
* побитовое исключающее ИЛИ

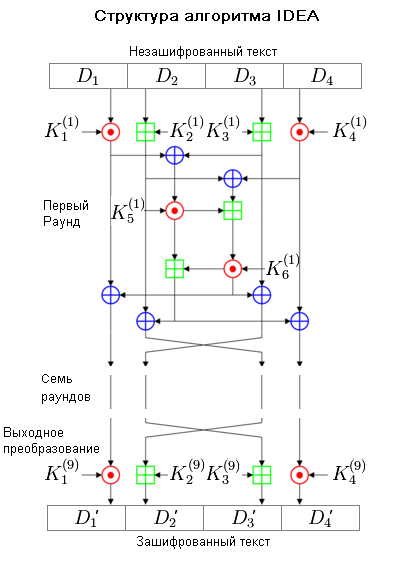


Рис. . Алгоритм IDEA

После выполнения выходного преобразования конкатенация подблоков *D*1', *D*2', *D*3' и *D*4' представляет собой зашифрованный текст. Затем берется следующий 64-битный блок незашифрованного текста и алгоритм шифрования повторяется. Так продолжается до тех пор, пока не зашифруются все 64-битные блоки исходного текста (рис. 1.6).

## 1.3. ГОСТ 28147-89

Блочный шифр с 256-[битным](http://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%82) [ключом](http://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D1%8E%D1%87_%28%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F%29) и 32 циклами преобразования, оперирующий 64-битными блоками. Основа [алгоритма](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) шифра — [сеть Фейстеля](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D1%8C_%D0%A4%D0%B5%D0%B9%D1%81%D1%82%D0%B5%D0%BB%D1%8F).

Для зашифровывания в этом режиме 64-битный блок [открытого текста](http://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%B9_%D1%82%D0%B5%D0%BA%D1%81%D1%82) сначала разбивается на две половины (младшие биты — A, старшие биты — B). На i-ом цикле используется подключ Ki:

Ai+1 = Bi⊕f(Ai,Ki)

Bi+1 = Ai

Для генерации подключей исходный 256-битный ключ разбивается на восемь 32-битных блоков: K1…K8.

Ключи K9…K24 являются циклическим повторением ключей K1…K8 (нумеруются от младших битов к старшим). Ключи K25…K32 являются ключами K8…K1.

После выполнения всех 32 раундов алгоритма, блоки A33 и B33 склеиваются (обратите внимание, что старшим блоком становится A33, а младшим — B33) — результат есть результат работы алгоритма.

Основные проблемы ГОСТа связаны с неполнотой стандарта в части генерации ключей и таблиц замен.

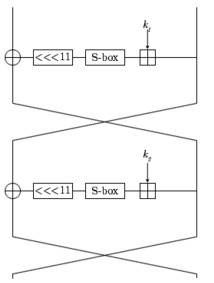


Рис. . Шифрация ГОСТ 28147-89

## 1.4. Режимы работы симметричных шифров

### 1.4.1 Режим электронной кодовой таблицы (ECB)

В этом режиме каждый из блоков кодируется независимо с использованием одного и того же ключа (рис.1.8) . Основное достоинство этого алгоритма - простота реализации. Недостаток - относительно слабая устойчивость против криптоанализа.

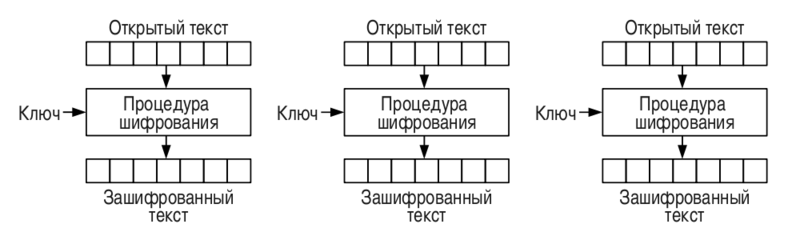


Рис. . Режим ECB

### 1.4.2 Режим сцепления блоков (CBC)

В этом режиме каждый последующий блок исходного текста складывается по модулю 2 с предыдущим блоком шифр-текста. Самый первый блок складывается со случайно сгенерированным блоком – вектором инициализации IV, который добавляется в начало шифр-текста (рис.1.9).

Режим CBC является самовосстанавливающимся. Ошибка в шифр-тексте влияет только на текущий и следующий за ним блок. Последний блок шифр-текста может служить кодом аутентификации сообщения.

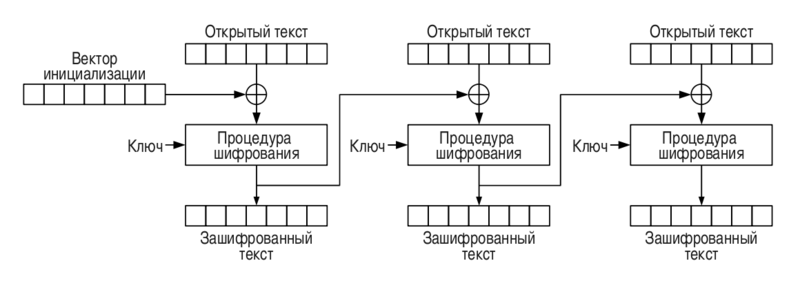


Рис. . Режим CBC

### 1.4.3 Режим обратной связи по шифр-тексту (CFB)

Исходный файл считывается последовательными t-битовыми блоками (t <= 64). 64-битовый сдвиговый регистр вначале содержит вектор инициализации IV, выравненный по правому краю.

Для всех i = 1...n блок шифртекста C(i) определяется следующим образом:

C(i) = M(i) XOR P(i-1) ,

где P(i-1) - старшие t битов операции Crypt(С(i-1)).

Обновление сдвигового регистра осуществляется путем удаления его старших t битов и дописывания справа C(i)(рис. 1.10).

Этот режим позволяет шифровать текст произвольной длины, не дополняя его до размеров полного блока.

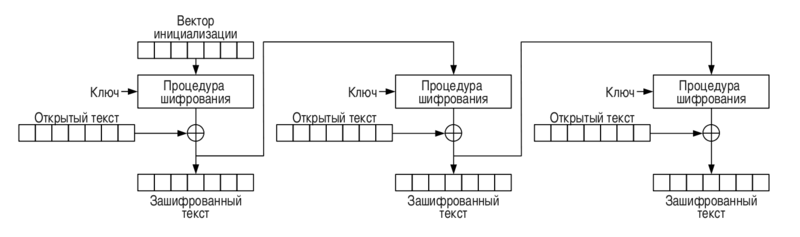


Рис. . Режим CFB

### 1.4.4 Режим обратной связи по выходу (OFB)

Данный режим отличается от CFB только способом формирования сдвигового регистра (рис. 1.11).

Режим OFB имеет следующее преимущество по сравнению с режимом CFB: ошибки, возникающие в результате передачи по каналу с шумом, при дешифровании не «размазываются» по всему шифротексту, а локализуются в пределах одного блока. Однако это свойство дает дополнительные параметры для криптоанализа, поэтому данный режим используется только при необходимости (высокий уровень помех).

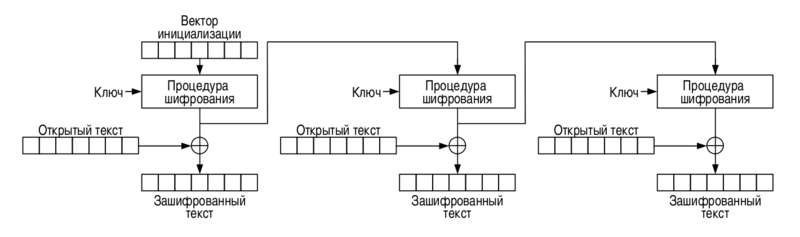


Рис. . Режим OFB

## 1.5. Вывод

Сила алгоритма заключена в его S-блоках. У CAST нет фиксированных S-блоков, для каждого приложения они конструируются заново. Изогнутыми функциями являются столбцы S-блоков, обеспечивающие необходимые свойства. Созданные для данной реализации CAST S-блоки уже больше никогда не меняются.

Основными достоинствами CAST являются:

* отсутствие уязвимостей.
* возможность быстрого выполнения расширения ключа, то есть в процессе операции кодирования, но не декодирования.

CAST устойчив к линейному и дифференциальному криптоанализу. На данный момент неизвестно ничего, кроме грубой силы, что может его взломать.

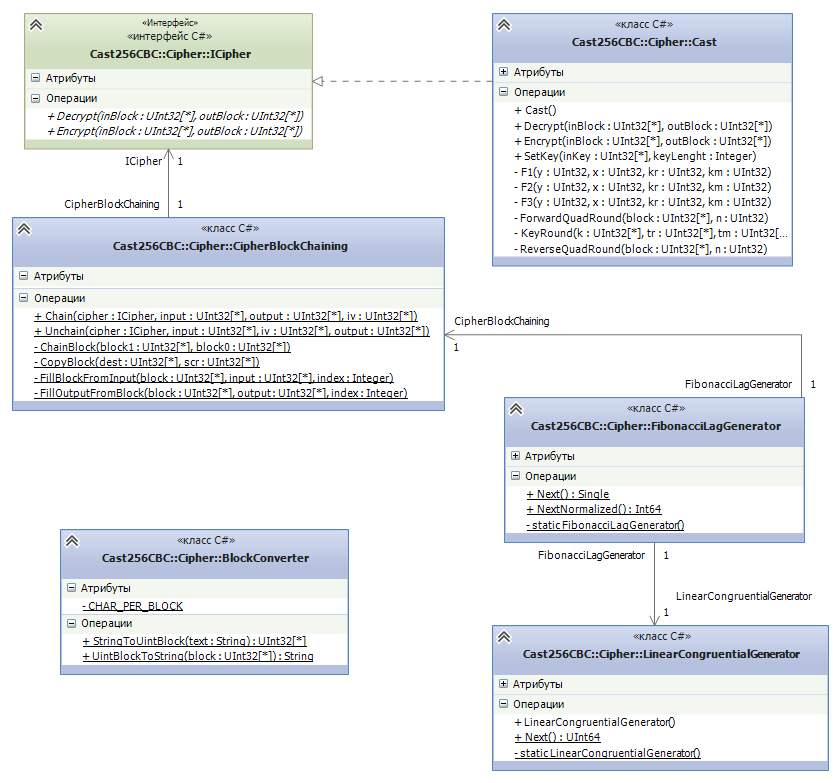
Среди режимов шифрования наиболее предпочтительными являются CBC и CFB, за счет высокой криптостойкости. Выбор зависит от способа применения шифра – блочный или поточный.

# КОНСТРУКТОРСКАЯ ЧАСТЬ

## Программная реализация CAST-256

Для программной реализации был выбран язык программирования C#, при помощи которого можно просто создавать GUI приложения для Windows. В качестве среды разработки была выбрана MS Visual Studio 2013, поскольку только эта IDE очень удобна.

В процессе реализации задачи курсовой работы были созданы несколько классов, с помощью которых можно выполнять CAST-256 шифрование с помощью сцепления блоков.



**Рис. 2.1. UML диаграмма классов проекта**

Реализация алгоритма CAST-256 осуществляется классом Cast. Данный класс позволяет выполнять шифрование и дешифрование 128-битных блоков данных. Данный класс можно использовать отдельно, т.е. без какого-то конкретного режима шифрования. Перед использованием шифрования или дешифрования необходимо выполнить операцию расширения ключа с помощью метода SetKey(…).

Режим сцепления блоков реализуется статическим классом ChipherBlockChaining. В открытом доступе есть методы Chain(…) и Unchain(…). Для выполнения действий в методы необходимо передавать объект класса, реализующего интерфейс ICipher, например Cast.

В результате выполнения метода Chain(…) создается вектор инициализации. Вектор инициализации при прямом шифровании генерируется случайным образом с помощью генератора случайных чисел Фибоначчи с запаздываниями. Данный вектор необходимо сохранить для возможности дешифровки.

Для обработки данных необходимо предоставить данные в виде uint32-массива. Для преобразования текста в uint32[] и обратно используется статический класс BlockConverter.

## Описание классов и основных методов

Класс Cast:

* Encrypt – шифрует блок данных, заданный uint32[4] массивом;
* Decrypt – дешифрует блок данных, заданный uint32[4] массивом;
* SetKey – выполняет процедуру расширения ключа;
* F1 – реализация раундовой функции *f1;*
* F2 – реализация раундовой функции *f2;*
* F3 – реализация раундовой функции *f3;*
* KeyRound – реализация раундовой функции для расширения ключа;

Класс CipherBlockChaining:

* Chain – реализует сцепление блоков для заданного uint32 массива. Размер массива должен быть кратен 4, иначе будут добавлены необходимые элементы и увеличен размер. В ходе работы метода создается вектор инициализации, который потом нужно хранить для возможности расшифровки текста
* ChainBlock – выполняет XOR для двух блоков
* Unchain – выполняет расцепление блоков. Входные параметры аналогичны методу Chain за исключением вектора инициализации IV.

Класс Converter:

* StringToUintBlock – трансформирует строку в массив uint32 чисел
* UintBlockToString – трансформирует массив uint32 в строку

# ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ

## 3.1 Руководство пользователя

Интерфейс пользователя состоит из двух вкладок. При выборе вкладки «Шифрование» пользователю нужно выбрать пути к файлу с оригинальным текстом и файлу с ключом.

По нажатию на кнопку «Зашифровать текст» программа выполняет шифрование данных и сохраняет шифротекст, вектор инициализации и ключ в отдельные файлы. Результат работы программы отображается в специальном поле (рис 3.1).

Для расшифровки текста необходимо выбрать файлы с шифротекстом, вектором инициализации и ключом. После расшифровки получившийся текст будет сохранен в отдельный файл (рис 3.2).

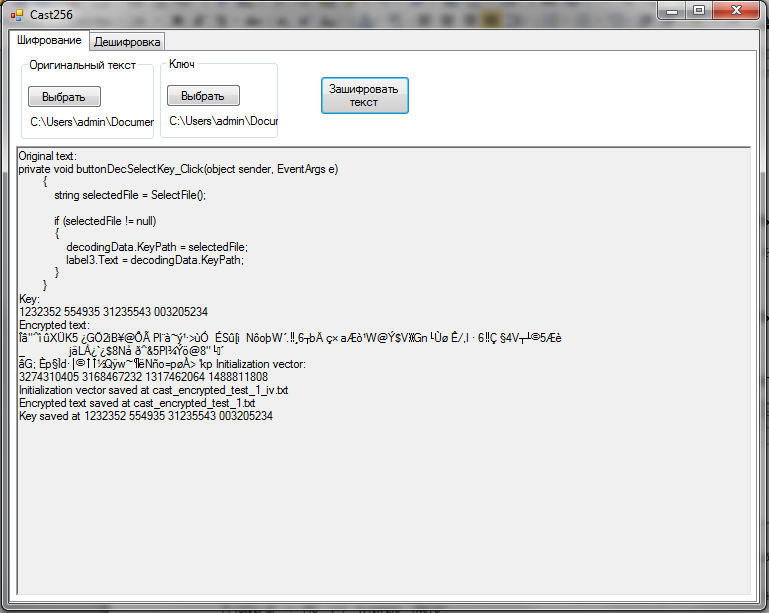
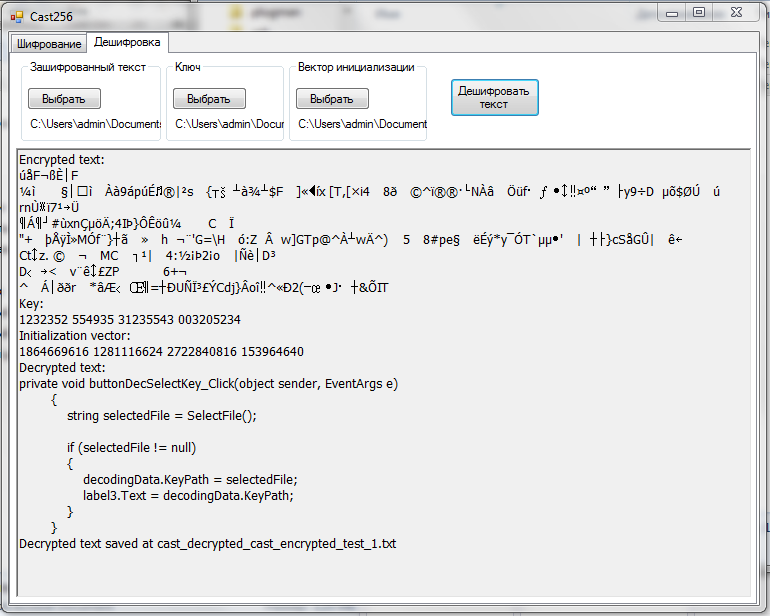


Рис. 3.1. Интерфейс программы. Вкладка "Шифрование"



**Рис. 3.2. Интерфейс программы. Вкладка "Дешифровка"**

## 3.2 Тестирование работы шифра

В рамках экспериментальной части будут рассмотрены несколько тестов для проверки корректности работы программы.

* 1. Стандартный сценарий.

|  |  |
| --- | --- |
| Оригинальный текст | private void buttonDecSelectKey\_Click(object sender, EventArgs e)  {  string selectedFile = SelectFile();  if (selectedFile != null)  {  decodingData.KeyPath = selectedFile;  label3.Text = decodingData.KeyPath;  }  } |
| Ключ шифрования | 1232352 554935 31235543 3205234 |
| Зашифрованный текст | úåF¬ßÈF¼ì§ìÀà9ápúÉ®|²s{à¾$F]«íx [T,[×i48ð©^ï®®·NÀâÖüf¤ºy9÷D­µõ$ØÚú rnÙï7¹Ü  Á#ùxnÇµöÄ;4IÞ}ÔÊöû¼CÏ  "+þÅÿÌ»MÓf¨}ã»h‑¬¨'G=\Hó­:Z­Â‑w]GTp@^ÀwÄ^)58#pe§ëÉý\*y¯ÓT`µµ'|}cSåGÛ|ê  Ctz. ©¬MC¹|4:½¡Þ2¡o|ÑèD³  D<v¨ê£ZP 6+¬^Áððr\*âÆ=ÐUÑÏ³£ÝCdj}Âoî^«Ð2(J&ÕIT |
| Ключ для расшифрования | 1232352 554935 31235543 3205234 |
| Вектор инициализации | 1864669616 1281116624 2722840816 153964640 |
| Расшифрованный текст | private void buttonDecSelectKey\_Click(object sender, EventArgs e)  {  string selectedFile = SelectFile();  if (selectedFile != null)  {  decodingData.KeyPath = selectedFile;  label3.Text = decodingData.KeyPath;  }  } |

* 1. Проверка использования инициализирующего вектора

|  |  |
| --- | --- |
| Исходный текст | 11 22 33 44 55 66 5 |
| Ключ шифрования | 45345 554935 31235543 003205234 |
| Инициализирующий вектор (результат шифрования) | 1927319360 1351663968 740894192 1223461888 |
| Зашифрованный текст | ÿÃ#MW#8d'×A6#Ê¼Eù¹¢~XLÖq |
| Инициализирующий вектор с ошибкой | 1427319365 1351663968 740894192 1223461888 |
| Ключ расшифрования | 45345 554935 31235543 003205234 |
| Расшифрованный текст | Â72 33 44 55 66 5 |

* 1. Проверка распространения ошибки

|  |  |
| --- | --- |
| Исходный текст | 11 22 33 44 55 66 5 |
| Ключ шифрования | 45345 554935 31235543 003205234 |
| Инициализирующий вектор | 505223824 1930461920 2842651104 1450950224 |
| Зашифрованный текст | ßbí¸QÃ%øë{ë`Gôÿíçé}Ø‑íùÂU|, |
| Зашифрованный текст с внесенной ошибкой | ß~~b~~ í¸QÃ%øë{ë`Gôÿíçé}Ø‑íùÂU|, |
| Ключ расшифрования | 45345 554935 31235543 003205234 |
| Расшифрованный текст | 6/»þåQ7VÆúÐõr6b5 |

* 1. Проверка на потерю части сообщения

|  |  |
| --- | --- |
| Исходный текст | 11 22 33 44 55 66 5 |
| Ключ шифрования | 45345 554935 31235543 003205234 |
| Инициализирующий вектор | 1657301616 467659184 1454415936 1162831192 |
| Зашифрованный текст | «îlâ==Ëp¢ñü¡YË:ÛüàU`o·ç"°k |
| Зашифрованный текст с потерянным символом | «îlâ==Ëp~~¢~~ñü¡YË:ÛüàU`o·ç"°k |
| Ключ расшифрования | 45345 554935 31235543 003205234 |
| Расшифрованный текст | ÏoúÝå+ÿ«]çÛ[²»n[×¨»µ»ò/ |

# СПИСОК ЛИТЕРАТУРЫ

1. Шнайер, Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си / Б. Шнайдер. – М.: Триумф, 2003. – 815 с.
2. М. Масленников, Практическая криптография / Масленников М. – Спб.: БХВ-Петербург, 2003. - 458 с.
3. С. Панасенко, Алгоритмы шифрования / Панасенко С. – СПб.: БХВ-Петербург, 2009. – 576 с.
4. И. Ященко, Основные математические понятия криптографии / Ященко И.В. – М.: Москва, 2003.