МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра защиты информации

**

**КУРСОВАЯ РАБОТА**

**по дисциплине: «Программирование»**

**на тему: «***Разработка программного обеспечения для шифрования “Алгоритм AES”***»**

|  |  |
| --- | --- |
| Выполнила: | Проверил: |
| Студентка гр. АБ-121, АВТФ | *доцент кафедры ЗИ* |
| Вяткина Таисия Владимировна | *Архипова А. Б* |
| «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2022 г. | «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2022 г. |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| (подпись) | (подпись) |

Новосибирск 2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

*Защита информации*

Кафедра \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(полное название кафедры)

**ЗАДАНИЕ  
НА КУРСОВУЮ РАБОТУ**

##### УТВЕРЖДАЮ

Зав. кафедрой Иванов А. В.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата)

*Вяткиной Таисии Владимировне*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Горячеву Данилу Сергеевичу*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Горячеву Данилу Сергеевичу*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Горячеву Данилу Сергеевичу*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

*Кисель Анастасия Владимировна*

студенту*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(фамилия, имя, отчество)

*10.03.01*

*Информационная безопасность*

Направление подготовки\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(код и наименование направления подготовки бакалавра)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Факультет автоматики и вычислительной техники*\_\_\_\_\_\_\_

(полное название факультета)

*Разработка программного обеспечения для шифрования «Алгоритм AES»*

Тема\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Календарный план

|  |  |  |
| --- | --- | --- |
| Наименование задач  (мероприятий),  составляющих задание | Дата выполнения задачи  (мероприятия) | Подпись руководителя |
| 1 | 2 | 3 |
| Развернутая постановка задачи, изучение необходимой научно-технической литературы | 12.09.2022-25.10.2022 |  |
| Разработка структуры данных и алгоритма решения задачи | 19.10.2022-8.11.2022 |  |
| Написание текста задачи | 17.10.2022-21.11.2022 |  |
| Тестирование и отладка программного продукта | 22.11.2022-28.11.2022 |  |
| Оформление отчета о проделанной работе | 15.11.2022-13.12.2022 |  |
| Сдача работы руководителю и ее защита | 15.12.2022г. |  |

Задание согласовано и принято к исполнению.

Руководитель

от НГТУ

Архипова Анастасия Борисовна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия, имя, отчество)

к.т.н.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ученая степень, ученое звание)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата)

Студент

Вяткина Таисия Владимировна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия, имя, отчество)

*АВТФ, АБ-121*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(факультет, группа)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата)

СОДЕРЖАНИЕ

[Введение 5](#_Toc122339473)

[1 Современные блочные алгоритмы шифрования 7](#_Toc122339474)

[1.1 Математические основы 7](#_Toc122339475)

[1.2 Основные методы проектирования современных блочных шифров 7](#_Toc122339476)

[1.3 Сеть Файстеля 9](#_Toc122339477)

[1.4 Методы взлома современных блочных шифров 10](#_Toc122339478)

[2 Алгоритм AES 12](#_Toc122339479)

[2.1 Описание алгоритма 12](#_Toc122339480)

[2.2 Математический аппарат 15](#_Toc122339481)

[2.3 Методы взлома 15](#_Toc122339482)

[2.4 Модификация алгоритма AES 18](#_Toc122339483)

[3 Описание программного обеспечения 19](#_Toc122339484)

[3.1 Состав и структура программного обеспечения 19](#_Toc122339485)

[3.2 Программная реализация функций 19](#_Toc122339486)

[3.2.1 SubBytes и InvSubBytes 19](#_Toc122339487)

[3.2.2 ShiftRows и InvShiftRows 20](#_Toc122339488)

[3.2.3 MixColumns и InvMixColumns 20](#_Toc122339489)

[3.2.4 AddRoundKey 20](#_Toc122339490)

[3.3 Руководство пользователя 21](#_Toc122339491)

[3.3.1 Введение 21](#_Toc122339492)

[3.3.2 Назначение и условия применения 21](#_Toc122339493)

[3.3.3 Подготовка к работе 21](#_Toc122339494)

[3.3.4 Описание операций 22](#_Toc122339495)

[3.3.5 Сообщения пользователю 24](#_Toc122339496)

[3.3.6 Аварийные ситуации 25](#_Toc122339497)

[4 Руководство системного программиста 25](#_Toc122339498)

[4.1 Общие сведения 25](#_Toc122339499)

[4.2 Структура программы 25](#_Toc122339500)

[5 Контрольный пример 27](#_Toc122339501)

[Заключение 30](#_Toc122339502)

[Список использованных источников 31](#_Toc122339503)

[Приложение 33](#_Toc122339504)

# Введение

В последнее время довольно остро стоит вопрос о том, как обеспечивается безопасность в Интернете, особенно с учетом того, что существует множество злоумышленников в лице и простых граждан, и целых государств, которые постоянно пытаются взломать хранилища данных. Криптография — это один из ответов, который лучше всего подходит для этого вопроса.

Это технология, которая постоянно совершенствуется, чтобы гарантировать, что стандарты, на которые она опирается, всегда находятся на самом высоком уровне.

Примером такого стандарта является AES, аббревиатура, с которой многие, вероятно, сталкивались раньше, особенно если постоянно используют приложения для связи, такие как WhatsApp, Signal или Telegram, или программное обеспечение VPN. Целью данной работы является самостоятельная реализация программного обеспечения для шифровки и дешифровки текста. Решение данной задачи призвано обеспечить засекречивание информации, рассекречивание зашифрованной информации, защиту от третьих лиц.

AES существует с 2001 года, и с момента его появления было предпринято множество попыток взломать его, что позволяет с уверенностью сказать, что он выдержал испытание временем и по-прежнему является надежным выбором, когда речь идет об алгоритмах шифрования.

Количество шагов, которые проходит алгоритмический процесс, чтобы сделать данные нечитаемыми, огромно, и для взлома AES потребуются буквально миллиарды лет, даже с сегодняшними вычислительными мощностями и технологическими достижениями. Именно этими фактами обусловлен выбор AES в качестве реализуемого стандарта.

Теоретической основой написания курсовой работы явились книги авторов: Daemen J., Rijmen V[1]; Баричев С. Г[2]; Б. Я. Рябко, А. Н. Фионов[4];

Практической основой написания курсовой работы явились учебные пособия и интернет-источники AES related home page[3]; Конструктор и деструктор классов в С#[5]; Введение в WindowsForms[6].

Курсовая работа написана в среде разработки Visual Studio 2019, на языке программирования C# с использованием Windows Forms.

Современные блочные алгоритмы шифрования

## Математические основы[3]

Общая алгебра — раздел математики, изучающий алгебраические системы, такие как группы, кольца, поля, модули, решётки, а также отображения между такими структурами. Для изучения структур используются общие методы и сходные понятия. В раздел общей алгебры входит понятие конечного поля (поля Галуа). Это поле, состоящее из конечного числа элементов. С точностью до изоморфизма конечное поле полностью определяется его порядком, который всегда является степенью какого-нибудь простого числа. Для поля Галуа определены особые методы полиномиального сложения и умножения.

Алгебра логики — раздел математической логики, в котором изучаются логические операции над высказываниями. Чаще всего предполагается, что высказывания могут быть только истинными или ложными, то есть используется так называемая бинарная или двоичная логика. К числу двоичных операций относится сложение по модулю 2 (XOR). Это логическая и битовая операция, в случае двух переменных результат выполнения операции истинен тогда и только тогда, когда один из аргументов истинен, а другой — ложен. Такая операция естественным образом возникает в кольце вычетов по модулю 2.

Линейная алгебра — раздел алгебры, изучающий объекты линейной природы: векторные или линейные пространства, линейные отображения, системы линейных уравнений. Среди основных инструментов, используемых в линейной алгебре — определители, матрицы, сопряжение. Матрицы — это математический объект, записываемый в виде прямоугольной таблицы элементов кольца или [поля](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B5_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) (например, целых, действительных или комплексных чисел), который представляет собой совокупность [строк](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D1%82%D1%80%D0%BE%D0%BA%D0%B0_%D0%BC%D0%B0%D1%82%D1%80%D0%B8%D1%86%D1%8B&action=edit&redlink=1) и [столбцов](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D1%82%D0%BE%D0%BB%D0%B1%D0%B5%D1%86_%D0%BC%D0%B0%D1%82%D1%80%D0%B8%D1%86%D1%8B&action=edit&redlink=1), на пересечении которых находятся его элементы.

## Основные методы проектирования современных блочных шифров

Клод Шеннон определил основные принципы[7], которым должны соответствовать надежные шифры, доказал существование абсолютно стойких, невскрываемых шифров, и сформулировал условия, необходимые для этого. Кроме того, он ввел в рассмотрение понятия перемешивания и рассеивания, и предложил строить стойкие криптографические системы из относительно несложных преобразований. Рассеивание скрывает отношения между зашифрованным текстом и исходным текстом. Перемешивание скрывает отношения между зашифрованным текстом и ключом

Перемешивание маскирует взаимосвязи между открытым текстом, шифртекстом и ключом. Другими словами, если единственный бит в ключе изменен, все биты в зашифрованном тексте будут также изменены. Даже незначительная зависимость между этими тремя составляющими может быть использована в дифференциальном и линейном криптоанализе. Пусть имеется пространство с мерой (или вероятностное пространство) Ω и некоторое сохраняющее меру отображение F этого пространства в само себя, то есть такое отображение, что мера отображенной области FR равна мере исходной области R. Отображение называется перемешиванием, если для любой функции, определенной на пространстве, и для любой области R интеграл от этой функции по области FnR стремится при к интегралу от функции по всему пространству Ω, умноженному на объем области R.

Рассеивание распространяет влияние отдельных битов открытого текста на возможно больший объем шифртекста. Рассеивание подразумевает, что каждый символ (символ или бит) в зашифрованном тексте зависит от одного или всех символов в исходном тексте. Другими словами, если единственный символ в исходном тексте изменен, несколько или все символы в зашифрованном тексте будут также изменены. Если данное условие выполняется, то при шифровании двух блоков данных с минимальными отличиями между ними должны получаться совершенно непохожие друг на друга блоки шифротекста.

## Сеть Файстеля

Сеть Фейстеля — это метод блочного шифрования, разработанный Хорстом Фейстелем в лаборатории IBM в 1971 году. Сегодня сеть Фейстеля лежит в основе большого количества криптографических протоколов. Сеть Фейстеля оперирует блоками открытого текста[8].

Алгоритм работы (Рисунок 1.1):

1. Блок разбивается на две равные части — левую (L) и правую (R).
2. После разбиения левый подблок изменяется функцией с использованием ключа K: . В качестве функции можно представить любое преобразование — например, шифр сдвига с ключом K.
3. Полученный подблок складывается по модулю 2 с правым подблоком R: .
4. Далее полученные части меняются местами и соединяются.
5. При переходе от одной ячейки к другой меняется ключ, причём выбор ключа зависит от конкретного алгоритма.

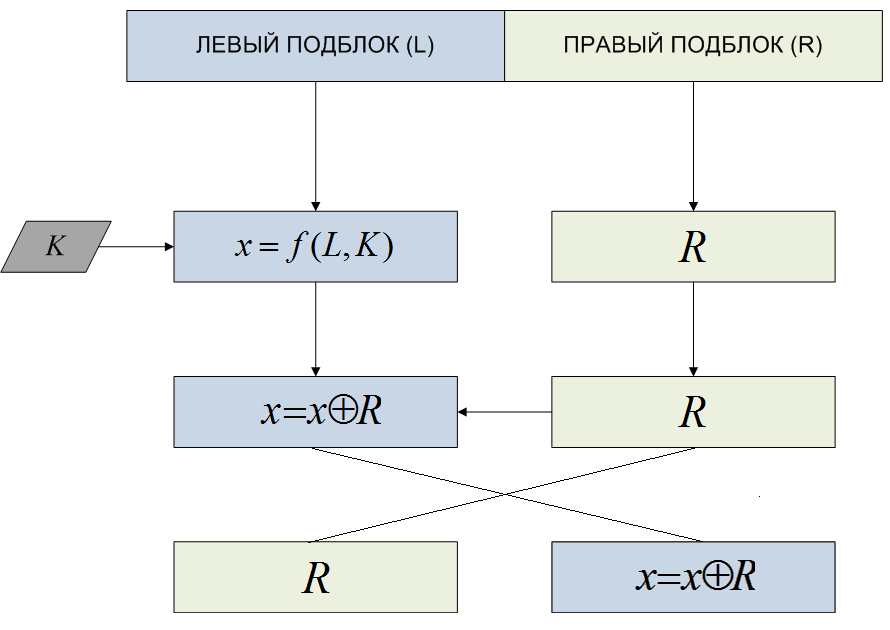


Рисунок 1.1 – Алгоритм работы

Приведённая схема называется ячейкой Фейстеля. Сама сеть Фейстеля состоит из нескольких ячеек. Полученные на выходе первой ячейки подблоки поступают на вход второй ячейки, результирующие подблоки из второй ячейки попадают на вход третьей ячейки и так далее в зависимости от количества раундов сети Фейстеля. В каждом таком раунде применяется заранее определенный раундовый ключ. Чаще всего раундовые ключи выработаны из основного секретного ключа K. Когда все раунды будут пройдены, подблоки текста «склеиваются», и получается шифротекст.

Операции шифрования и расшифрования на каждом этапе очень просты, и при определённой доработке совпадают, требуя только обратного порядка используемых ключей.

Шифрование при помощи данной конструкции легко реализуется как на программном уровне, так и на аппаратном, что обеспечивает широкие возможности применения. Большинство современных блочных шифров используют сеть Фейстеля в качестве основы.

## Методы взлома современных блочных шифров[2]

Атаки традиционных шифров могут также использоваться для современных блочных шифров, но сегодняшние блочные шифры успешно противостоят большинству атак. Например, грубая силовая атака ключа, как правило, неосуществима, потому что ключи обычно имеют очень большую длину. Однако недавно были изобретены некоторые новые виды атак блочных шифров, которые основаны на структуре современных блочных шифров. Эти атаки используют методы дифференциального, и линейного анализа.

#### Дифференциальный криптоанализ

Идею относительно дифференциального криптоанализа предложили Эли Бихам и Ади Шамир. Это — атака с выборкой исходного текста. Злоумышленник может каким-либо образом получить доступ к компьютеру одного из абонентов и завладеть выборочно частью исходного текста и соответствующего зашифрованного текста. Цель состоит в том, чтобы найти ключ шифра.

Перед тем как злоумышленник предпримет атаку с выборкой исходного текста, она должна проанализировать алгоритм шифрования, чтобы собрать некоторую информацию об отношениях зашифрованного и исходного текстов. Очевидно, он не знает ключ шифра. Однако некоторые шифры имеют слабости в структурах, которые могут позволить найти различия исходного текста и различия зашифрованного текста, не зная ключ.

После того как анализ однажды сделан, он может быть сохранен для будущего использования, пока структура шифра не изменится. Злоумышленник может выбрать для атак исходные тексты.

После запуска некоторых атак с соответствующей выборкой исходного текста злоумышленник может найти некоторую пару "исходный текст —зашифрованный текст", которая позволяет ему предположить некоторое значение ключа.

#### Линейный криптоанализ

Линейный криптоанализ был представлен Митцури Мацуи (Mitsuru Matsui) в 1993 году. Анализ использует атаки знания исходного текста (в отличии от атак с выборкой исходного текста в дифференциальном криптоанализе). Полное обсуждение этой атаки базируется на некоторых понятиях теории вероятностей. Линейный криптоанализ является одним из самых распространенных подходов.

Анализ основывается на построении соотношений между открытым текстом, ключом и зашифрованным сообщением, а затем их использовании для нахождения ключа.

В некоторых современных блочных шифрах может случиться, что некоторые блоки не полностью нелинейные; тогда они могут быть в вероятностном смысле аппроксимированы некоторыми линейными функциями.

Алгоритм AES

## Описание алгоритма

AES является стандартом, основанным на алгоритме Rijndael[1]. Для AES длина матрицы (блока) состояния постоянна и равна 128 бит, а длина шифроключа может составлять 128, 192, или 256 бит. При этом исходный алгоритм Rijndael допускает и длину ключа, и размер блока 128, 192, 256 бит.

AES использует несколько раундов, каждый раунд состоит из нескольких преобразований. Количество раундов зависит от размера ключа. 10, 12 и 14 раундов соответствуют ключам размеров 128, 192 и 256 бит. В начале и в конце шифра AES применяется термин блок данных; до и после каждого преобразования блок данных называется матрицей состояний State.

Структура каждого раунда ([Рисунок 2.1)](https://intuit.ru/studies/courses/552/408/lecture/9363?page=2#image.9.5) на стороне шифрования подразумевает, что каждый раунд, кроме последнего, использует четыре преобразования, которые являются обратимыми. Последний раунд имеет только три преобразования. Каждое преобразование принимает матрицу состояний и создает другую матрицу состояний, которая применяется для следующего преобразования или следующего раунда. Секция, предваряющая раунд, использует только одно преобразование (AddRoundKey); последний раунд использует только три преобразования (MixColumns — преобразование отсутствует).

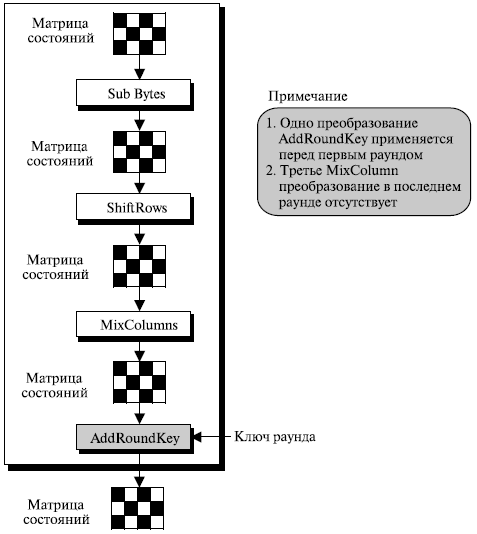


Рисунок 2.1 — Структура раунда AES

AES использует следующие преобразования[3]:

#### SubBytes()

SubBytes() обрабатывает каждый байт состояния, независимо производя нелинейную замену байтов, используя таблицу замен S-box (Рисунок 2.2). Чтобы применить подстановку к байту, байт интерпретируется как две шестнадцатеричные цифры. Левая цифра определяет строку, а правая — колонку в таблице перестановки. На пересечении строки и колонки, обозначенных этими шестнадцатеричными цифрами, находится новый байт.

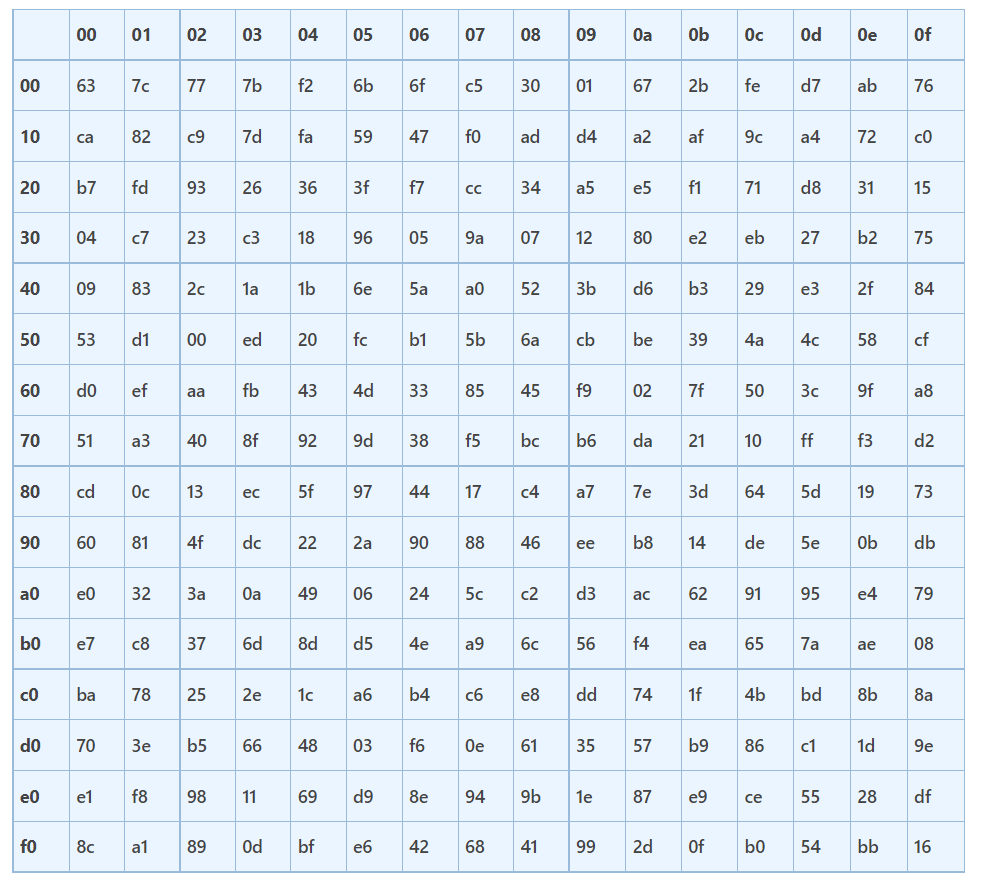


Рисунок 2.2 — S-Box

#### ShiftRows()

ShiftRows работает со строками матрицы состояния State. При шифровании применяется преобразование со смещением влево. Число сдвигов зависит от номера строки (0, 1, 2 или 3) матрицы состояний. Это означает, что строка 0 не сдвигается и последняя строка сдвигается на три байта.

#### MixColumns()

MixColumns работает на уровне столбца; оно преобразовывает каждый столбец матрицы состояний в новый столбец. Это преобразование — фактически матричное умножение столбца матрицы состояний и квадратной матрицы констант (Рисунок 2.3). Байты в столбце матрицы состояний и в матрице констант интерпретируются как слова по 8 битов (или полиномы) с коэффициентами в GF (поле Галуа). Умножение байтов выполняется по модулю (10001101) или (x8 + x4 + x3 + x + 1). Сложение — это применение операции XOR к словам по 8 бит.



Рисунок 2.3 — Матрицы для MixColumns

#### AddRoundKey()

Процедура производит побитовый XOR каждого байта сообщения с каждым байтом ключа RoundKey. AddRoundKey обрабатывает в один момент времени один столбец. Он подобен MixColumns. MixColumns умножает квадратную матрицу констант на каждый столбец матрицы состояний. AddRoundKey складывает ключевое слово раунда с каждым столбцом матрицы состояний. В MixColumns применяется матричное умножение; в AddRoundKey — операции сложения и вычитания. Так как сложение и вычитание в этом поле одни и те же, AddRoundKey инверсен сам себе.

## Математический аппарат

#### Процедура побитового исключающего ИЛИ (XOR) – алгебра логики;

#### Матричный сдвиг – линейная алгебра;

#### Полиномиальное сложение и умножение – общая алгебра;

## Методы взлома[2]

* Атака с распознаванием известного ключа

Атака с использованием известного ключа была разработана исследователями криптографии и наблюдалась в действии, и она показала многообещающие результаты. Атака была введена в 2007 году Винсентом Райменом и Ларсом Кнудсеном.

Атака имела два недостатка, из-за которых маловероятно ее использование в реальном сценарии: тот факт, что она использовала известный ключ и что она была проверена на версии из семи раундов. AES-128.

Учитывая, что в большинстве случаев ключ неизвестен злоумышленнику (если только он случайно не наткнется на него с точными инструкциями относительно его назначения) и что AES-128 имеет 10 раундов шифрования, атака на стандартный AES-128 вряд ли произойдет.

* Атака по связанным ключам

Атаки со связанными ключами — скорее категория, поскольку они представляют собой все формы криптоанализа, когда злоумышленники могут анализировать, как шифр работает с несколькими ключами. В этом типе атаки злоумышленник не знает значения ключей, используемых в шифре.

Однако несмотря на то, что ключи неизвестны, они также имеют какое-то математическое отношение друг к другу, которое известно злоумышленнику. Одним из самых простых примеров может быть то, что злоумышленник знает, что первые 16 бит ключей всегда одинаковы, но понятия не имеет, что это за биты на самом деле.

Хотя атаки с использованием связанных ключей могут быть успешными при правильном применении, они не представляют реальной опасности в реальном мире. Чтобы такая атака сработала, злоумышленнику необходимо успешно убедить криптографа зашифровать данные с использованием различных секретных ключей, которые математически связаны друг с другом, а затем поделиться секретом со злоумышленником.

Таким образом, этот тип атаки используется просто как напоминание о том, что человеческая ошибка может внести некоторые изменения в броню AES из-за плохой реализации этого стандарта шифрования.

* Атака по сторонним каналам

Этот тип атаки сочетает в себе несколько типов утечек данных для извлечения достаточного количества данных из алгоритма, чтобы его можно было взломать.

Например, злоумышленники могут сосредоточиться на энергопотреблении, электромагнитном излучении, времени, необходимом для выполнения различных вычислений, и даже на звуках, издаваемых во время вычислений, чтобы получить больше информации об алгоритме.

Существует несколько классов побочных каналов, в том числе:

* Атаки по времени

Атаки по времени вращаются вокруг измерения времени, необходимого для выполнения определенных вычислений (например, сравнение известного пароля с неизвестным).

* Кэш-атаки

Эти атаки возможны только в том случае, если злоумышленник может получить доступ к некоторым кэшам жертвы в различных средах, будь то среды виртуализации (виртуальные машины), общие физические системы или облачные сервисы, о которых мы говорим.

* Электромагнитные атаки

Электромагнитные атаки основаны на утечках электромагнитного излучения, которые могут быть использованы для извлечения различных битов информации. Измерение утечек электромагнитного излучения иногда может привести к обнаружению криптографических ключей.

* Атаки с мониторингом мощности

Злоумышленники отслеживают энергопотребление оборудования на определенных этапах вычислений, чтобы извлечь информацию об алгоритме.

* Акустический криптоанализ

Подобно атакам с контролем энергопотребления, но злоумышленники слушают и анализируют звуки, издаваемые аппаратным обеспечением при выполнении определенных шагов вычислений, и извлекают информацию из этих результатов.

* Сохранение данных

Этот тип атаки вращается вокруг конфиденциальных данных, к которым можно получить доступ и прочитать даже после удаления.

* Оптический анализ

В этом сложном типе атаки камеры с высоким разрешением используются для захвата секретов или конфиденциальных данных (таких как анализ индикаторов активности жесткого диска или состояний переключения транзисторов).

* Дифференциальный анализ неисправностей

Этот тип атаки основан на раскрытии секретов путем ввода ошибок в вычислениях, чтобы вызвать ошибку.

* Атаки с восстановлением ключа

Атаки с восстановлением ключа являются серьезным конкурентом, когда речь идет об эффективных способах взлома шифрования AES. В 2011 году была предпринята попытка такой атаки, чтобы доказать ее эффективность против механизма шифрования AES.

Этот тип атаки маловероятен, поскольку он предполагает, что злоумышленник получает по крайней мере пару зашифрованных и расшифрованных сообщений из системы, которая скоро будет скомпрометирована.

Атака показала многообещающие результаты, так как была примерно в четыре раза быстрее, чем обычный перебор пароля. Однако, учитывая, что атака методом перебора займет буквально миллиарды лет, атаки с восстановлением ключа все еще далеки от идеала, что делает маловероятным их использование в атаке.

Кроме того, даже если это занимало меньше времени, смена ключа безопасности часто могла сделать пару зашифрованных и расшифрованных сообщений бесполезной.

## Модификация алгоритма AES

Изменение раундового ключа с помощью основного ключа или вспомогательной таблицы. Это позволяет иметь для каждого раунда разный ключ шифрования.

Описание программного обеспечения

## Состав и структура программного обеспечения

В программу «AES encoder-decoder» включены 2 класса.

Класс «AES.cs» отвечает за математическую реализацию процессов шифрования и содержит поля «Key» и «Input» для работы с преобразованными в байтовые массивы входными данными. Методы «AESEncrypt» и «AESDecrypt» содержат вызовы необходимых функций шифровки и дешифровки в количестве, соответствующем размерности ключа. В данной реализации происходит 10 этапов шифрования и дешифрования с помощью функций SubBytes, ShiftRows, MixColumns, AddRoundKey и им обратных.

Класс «Form1.cs» отвечает за графический интерфейс, взаимодействие с пользователем и первичное преобразование входных данных в виде ключа и сообщения. Методы отвечают за файловый ввод и вывод ключей и сообщений, генерацию ключей, сохранение текстов из поля вывода и т.д.

## Программная реализация функций

### SubBytes и InvSubBytes

На вход и выход подаётся квадратная матрица из 16 десятичных чисел в диапазоне от 0 до 255. Каждое из чисел представляется в шестнадцатеричном виде в виде 2 символов. Левый отвечает за строки, а правый за колонки в константной матрице S-Box. Исходное значение матрицы заменяется на соответствующее значение из S-Box.

Обратное преобразование выполняется путём линейного поиска совпадающих значений из матрицы состояния и S-Box. В качестве нового значения берутся строка и столбец найденного элемента, составляющие в паре шестнадцатеричное число, которое записывается в матрицу состояния в виде десятичного.

### ShiftRows и InvShiftRows

На вход и выход подаётся квадратная матрица из 16 десятичных чисел в диапазоне от 0 до 255. Производится построчный циклический сдвиг влево на n символов. n находится в диапазоне от 0 до 3.

Обратная операция аналогична, за исключением изменения направления сдвига: вправо.

### MixColumns и InvMixColumns

На вход и выход подаётся квадратная матрица из 16 десятичных чисел в диапазоне от 0 до 255. Операция подразумевает умножение исходных столбцов матрицы на константнцю матрицу C (Рисунок 3.1) согласно правилам линейной алгебры и получения нового столбца. Функция может быть вызвана алгоритмом четыре раза. За сам процесс вычисления отвечает функция «GMul». Полиномиальное умножение 2 чисел происходит путём обработки чисел в двоичном представлении. Умножение на 1 даёт исходное число. Умножение эквивалентно сдвигу числа слева на единицу, а затем XOR с константным значением 0x1B, если старший бит был равен единице.

Обратная функция производит аналогичные операции с вызовом «GMul», но умножение происходит на обратную С матрицу.

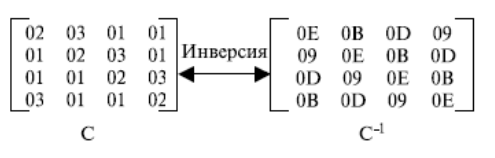


Рисунок 3.1 — матрицы констант для MixColumns

### AddRoundKey

На вход и выход подаётся квадратная матрица из 16 десятичных чисел в диапазоне от 0 до 255. Производится операция XOR между значениями матрицы состояний и раундового ключа, имеющими одинаковые индексы. Обратная функция аналогична, т.к. операция XOR обратна самой себе.

## Руководство пользователя

### Введение

Программа «AES encoder-decoder» предназначена для шифрования и дешифрования сообщений.

Программа предоставляет пользователю следующие возможности:

* Зашифровать сообщение;
* Расшифровывать сообщение;
* Ввести ключ вручную, из файла, сгенерированный случайно;
* Ввести сообщение вручную, из файла;
* Сохранить зашифрованный и расшифрованный текст;
* Очищать поля;
* Менять местами блоки зашифрованного и расшифрованного текста;

### Назначение и условия применения

Программа «AES encoder-decoder» предназначена для шифрования и расшифрования сообщений.

Решение данной задачи призвано обеспечить:

* Засекречивание информации;
* Рассекречивание зашифрованной информации;
* Защита от третьих лиц;

Программа реализована на языке C#. При написании курсовой работы была использована кроссплатформенная IDE – Microsoft Visual Studio 2019 без подключения сторонних библиотек с готовыми реализациями. Работает на операционной системе Windows. Дисковой памяти для запуска программы требуется не менее 1 МБ.

### Подготовка к работе

Система состоит из исполнительного файла «AES encoder-decoder.exe». Все файлы программы хранятся в папке «AES encoder-decoder».

Запуск программы осуществляется двойным щелчком по ярлыку на рабочем столе.

### Описание операций

Основные функции системы соответствуют шифрованию и дешифрованию сообщения. Для удобства пользователя и более легкого изучения, есть только одно главное окно. Далее описаны все функции системы, а также форма главного окна.

После запуска системы на экран выводится окно, в котором представлен весь интерфейс программы. Вид окна приведен на рисунке 3.2.

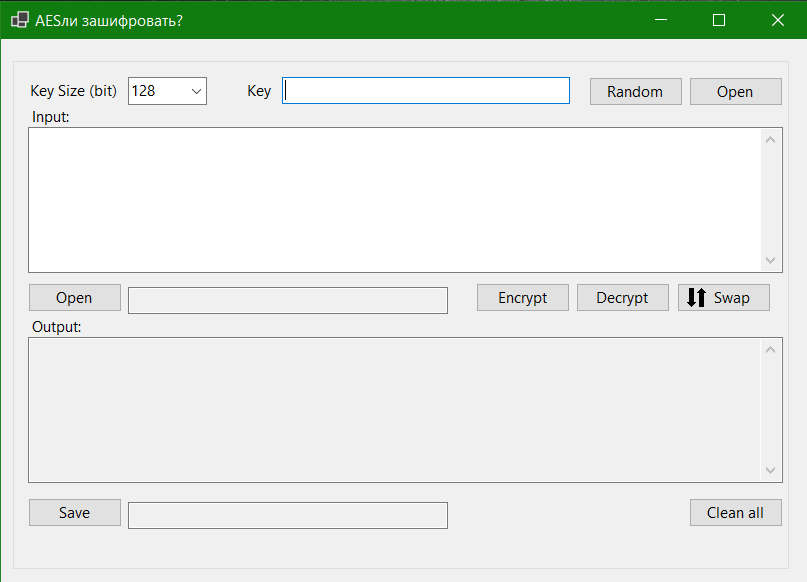


Рисунок 3.2 – Главное окно программы

Верхняя область, отвечающая за шифроключ (Рисунок 3.3) позволяет выбрать размерность ключа, сгенерировать его автоматически, вывести из файла и вручную. Причём ключи большей длины, чем выбрано в «Key Size» автоматически обрезаются, а ключи меньшей длины – достраиваются.



Рисунок 3.3 – Ключ шифрования

Область для ввода «Input», отвечающая за шифруемый (дешифруемый) текст (Рисунок 3.4) позволяет выбрать ввести текст сообщения из файла и вручную.

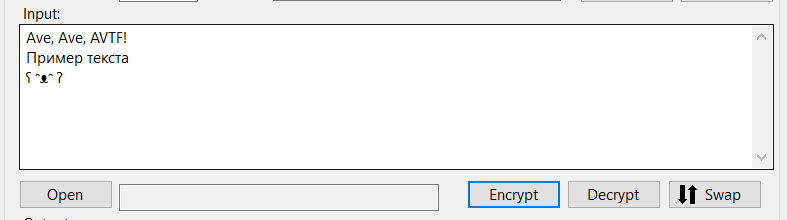


Рисунок 3.4 – Поле ввода

Кнопки «Encrypt» и «Decrypt» вызывают соответствующие функции обработки текста из «Input», а в поле «Output» появляется выходное сообщение, которое можно сохранить кнопкой «Save» или вернуть обратно в поле ввода для повторной операции кнопкой «Swap» (Рисунок 3.5)

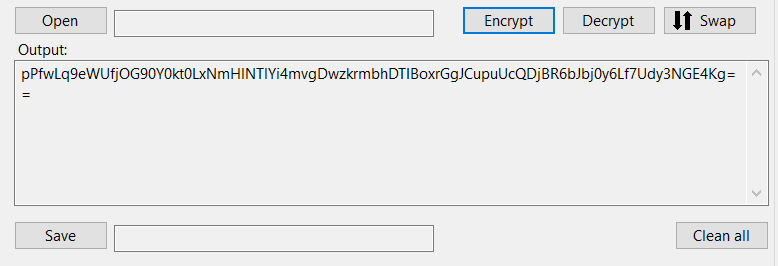


Рисунок 3.5 – Поле вывода

Описание функциональных кнопок содержится в таблице 3.1

Таблица 3.1 – Описание функциональных кнопок

|  |  |  |
| --- | --- | --- |
| Кнопка | Назначение | Действия системы |
|  | Вывод буквы или слова | Шифруется сообщение, с помощью ключа и выводится шифротекст |
|  | Вывод буквы или слова | Расшифруется шифротекст, с помощью ключа и выводится расшифрованное сообщение |
|  | Генерация случайного ключа | В поле для ключа вписывается случайный набор символов, общей длиной 128 бит |
|  | Открытие файла | Открывается диалоговое окно выбора файлов. После выбора содержимое отображается в окне ввода |
|  | Сохранение текста | Открывается диалоговое окно выбора места сохранения файла на диске. |
|  | Обмен блоками текста | Содержимое поля вывода переносится в поле ввода, а его собственное содержимое очищается |
|  | Очистка полей | Все поля ввода очищаются |

### Сообщения пользователю

В случае пустого ввода в «Input» пользователь получит соответствующее уведомление системы с предложением ввести текст (Рисунок 3.6) или ключ (Рисунок 3.7).

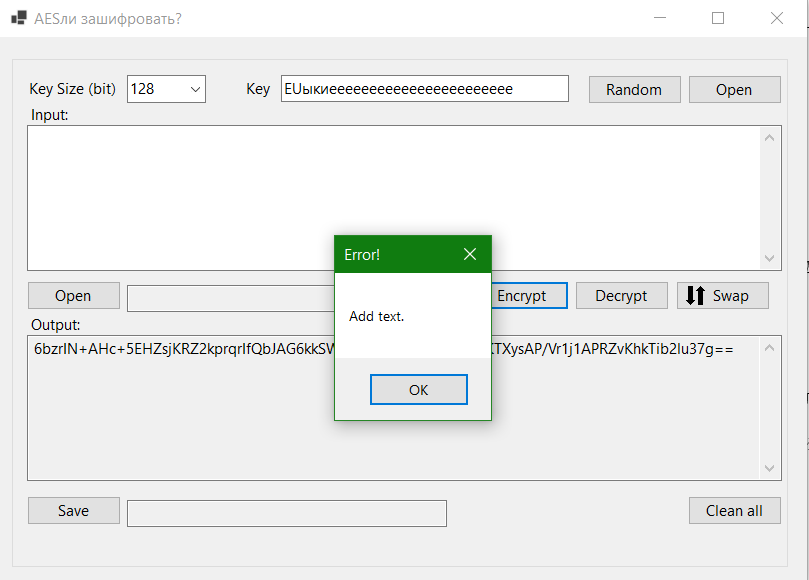


Рисунок 3.6 – Сообщение об отсутствии сообщения

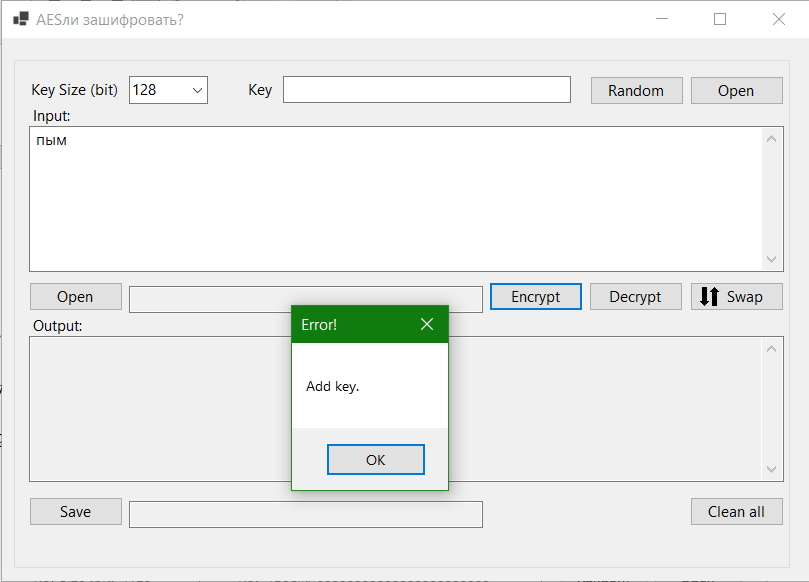


Рисунок 3.7 – Сообщение об отсутствии ключа

### Аварийные ситуации

К аварийным ситуациям относятся: нехватка оперативной памяти для создания окна программы. В данном случае работа программы прекращается.

Если исполняемый модуль программы не запускается, либо не выполнены требования условия работы программы, либо один из файлов поврежден, необходимо обратиться к разработчику программы или переустановить программу.

Руководство системного программиста

## Общие сведения

Данная программа предназначена шифрования и расшифрования сообщения алгоритмом Advanced Encryption Standard (AES).

Результатом работы программы является видеограмма.

Программа реализована на языке C#. При написании курсовой работы была использована кроссплатформенная свободная IDE для разработки – VisualStudio 2019. совместимой с Windows.

Работает в любой среде, совместимой с Windows. Памяти для запуска программы требуется не менее 1 МБ. Перед эксплуатацией системы необходимость установки дополнительных пакетов программного обеспечения отсутствует, за исключением .NET Core.

## Структура программы

Программа реализована на языке C# в исполняющей среде Visual Studio 2019, основанном на визуальном построении приложений (помещение компонентов на формы и изменение их свойств и методов), поэтому некоторые функции формирования окон невозможно описать в списке функций и листинге программы.

Модули программы:

*AES.cs* – файл с реализацией шифрования и расшифрования.

*Form1.cs* – файл с реализацией методов, необходимых для работы с графический интерфейсом программы.

Процедуры, методы и функции программы:

* *private void Encrypter\_Click(object sender, EventArgs e)* – метод при нажатии кнопки «Encrypt», отвечающий за получение данных с полей «Key», «Input» и шифрование сообщения и в конце вывода шифра в поле «Output».
* *private void Decrypter\_Click(object sender, EventArgs e)* – метод при нажатии кнопки «Decrypt», отвечающий за получение данных с полей «Key», «Input» и дешифрование сообщения и в конце вывода шифра в поле «Output».
* *private void Random\_Click(object sender, EventArgs e)* – метод, при нажатии кнопки «Random», отвечающий за генерацию ключа.
* *private void FromFile\_Click(object sender, EventArgs e)* и *private void button2\_Click(object sender, EventArgs e)* – метод, при нажатии «Open», отвечающий за считывание информации в виде массива байтов из файлов.
* *private void Cleaner\_Click(object sender, EventArgs e)* – метод, при нажатии кнопки «Clean», очищающий все поля.
* *private void SaveText\_Click(object sender, EventArgs e)* – метод при нажатии кнопки «Save», сохраняющий содержимое поля вывода в виде текстового файла.
* *private void Swap\_Click(object sender, EventArgs e)* – метод при нажатии кнопки «Swap», обменивающий местами поля ввода и вывода.
* *private byte[] Get\_textbox\_input\_byte\_representation(string textbox\_message)* – метод, преобразующий текстовые значения ввода.
* *private string Get\_textbox\_base64\_representation\_of\_byte\_buffer(byte[] byte\_buffer)* и *private string Get\_textbox\_unicode\_representation\_of\_byte\_buffer(byte[] byte\_buffer)* – метод, преобразующий байтовый массив в Base64 представление и Unicode соответственно.

Связь модулей программы между собой представлена на схеме 4.1.

Program.cs

Form1.cs

\_\_\_\_\_

Схема 4.1 – Связь модулей программы

Контрольный пример

После запуска на экран выводится главное окно программы, вид которого представлен на рисунке 5.1.

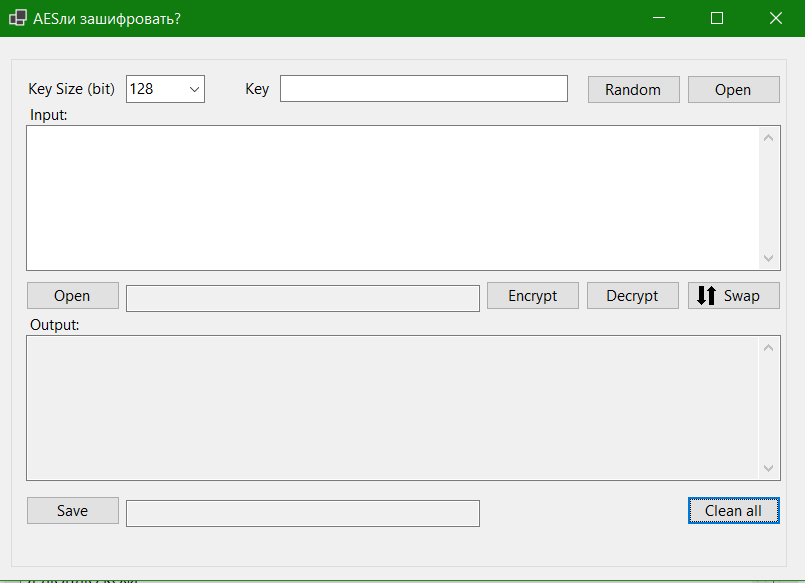


Рисунок 5.1 – Главное окно программы

Вводим ключ и сообщение вручную и нажимаем кнопку «Encrypt». Вид окна представлен на рисунке 5.2.

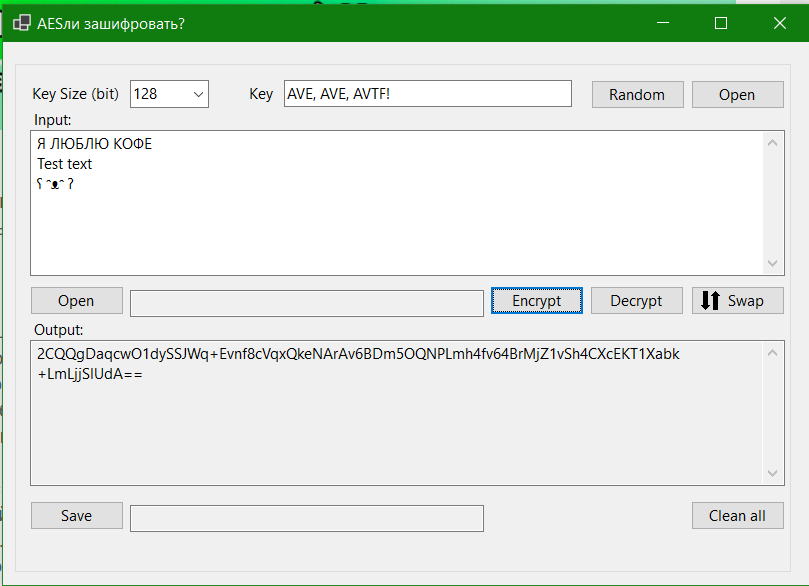


Рисунок 5.2 – Работа шифрования

Вводим ключ и сообщение в поля и нажимаем кнопку «Decrypt». Вид окна представлен на рисунке 5.3.

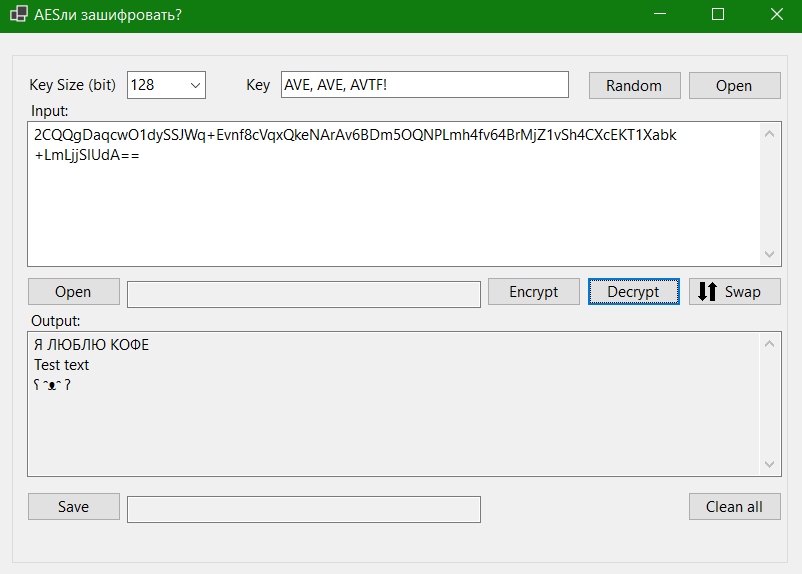


Рисунок 5.3 – Работа расшифрования

Для очистки полей нажимаем кнопку «Clean». Вид окна представлен на рисунке 5.4.

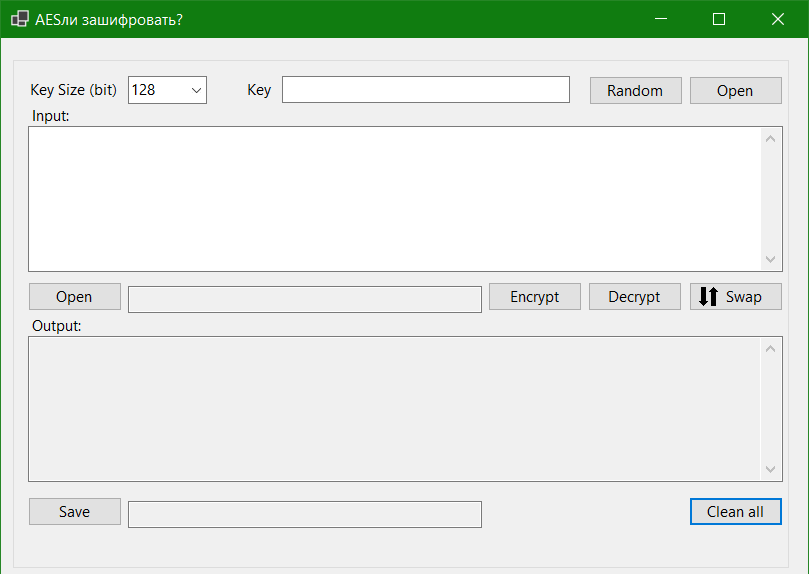


Рисунок 5.4 – Очистка окна

Генерируем ключ кнопкой «Random» и выбираем файл любого расширения кнопкой «Open», нажимаем «Encrypt» (Рисунок 5.5).

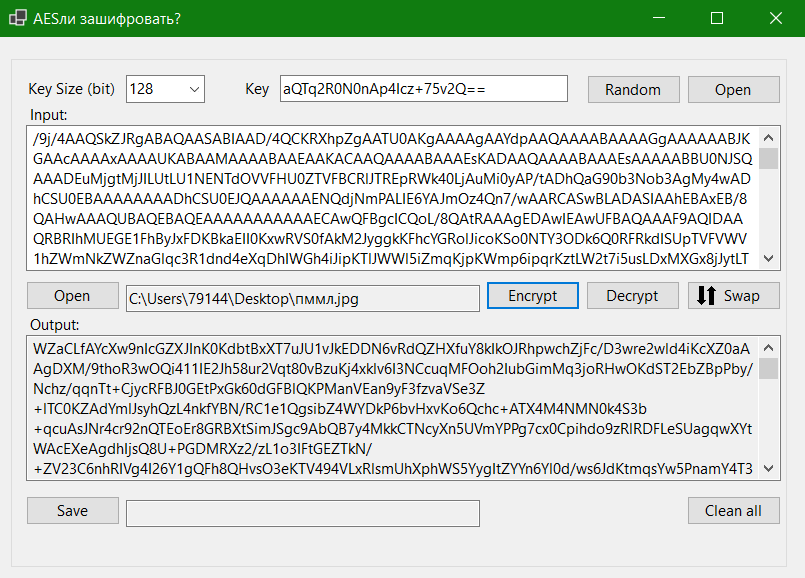


Рисунок 5.5 – Файловый и автоматический ввод

Сохраняем файл кнопкой «Save». Автоматический выставляется расширение «.enc». Вид окна представлен на рисунке 5.6. Дешифровка производится аналогично примеру с ручным вводом текста и сохранением.

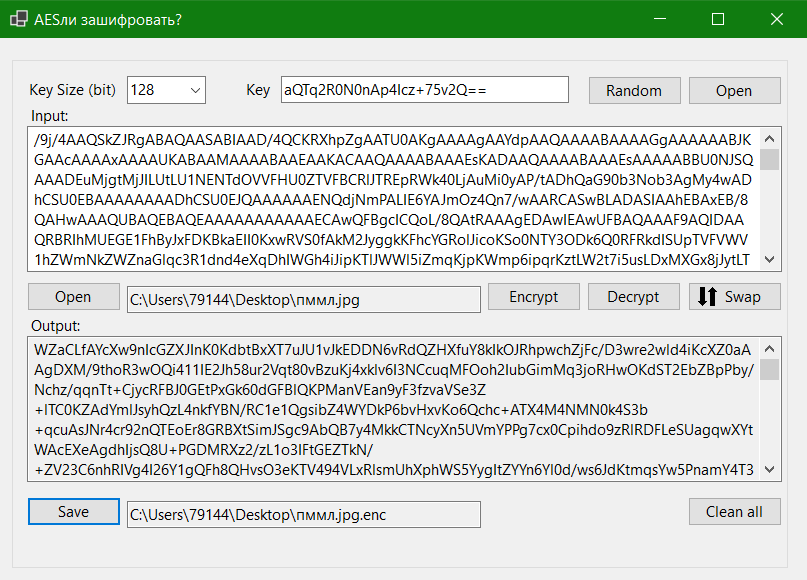


Рисунок 5.6 – Сохранение файла

# Заключение

В ходе выполнения курсовой работы были приобретены навыки по программированию на языке C# и разработке приложений с графическим интерфейсом Windows Form. Как результат, была создана программа «AES encoder-decoder», которая позволяет шифровать и дешифровывать информацию.

Для реализации поставленной цели был решён ряд задач: изучен криптографический метод защиты информации, его обоснование и математические доказательства; изучена и реализована работа программы с текстовыми и бинарными файлами.

Перспективами дальнейшей работы могут стать: повышение универсальности программы на этапе ввода исходного текста, увеличение длины используемого ключа, организация многопоточности.

Программа для использования подойдёт людям всех возрастов с любым знанием компьютера. Программа может быть использована, как для шифрования информации, так и дешифрования.

# Список использованных источников

1. J. Daemen and V. Rijmen, The block cipher Rijndael, Smart Card research and Applications, LNCS 1820, Springer-Verlag, J. Daemen and V. Rijmen, стр. 288-296. (дата обращения 19.10.2022)
2. Стандарт AES. Алгоритм Rijdael [учебное пособие] Баричев С. Г., Гончаров В. В., Серов Р. Е. 2.4.2. (дата обращения 20.10.2022)
3. B. Gladman’s AES related home page [Электронный ресурс]. – Режим доступа: <http://fp.gladman.plus.com/cryptography_technology/>(дата обращения 10.11.2022)
4. Б. Я. Рябко, А. Н. Фионов, Криптографические методы защиты информации: учебное пособие. — М.: Горячая линия—Телеком, 2005. — стр.167-177. (дата обращения 12.11.2022).
5. Конструктор и деструктор классов в С# [Электронный ресурс]. – Режим доступа: <https://csharp.net-tutorials.com/ru/> (дата обращения: 18.11.2022)
6. Введение в Windows Forms [Электронный ресурс]. —Режим доступа: <https://metanit.com/sharp/windowsforms/1.1.php/> , (дата обращения 21.11.2022).
7. К. Шеннон. "Работы по теории информации и кибернетике", М., ИЛ, 1963, с. 333-369 (Перевод В.Ф.Писаренко).Шеннон, Клод Эльвуд.  
   Работы по теории информации и кибернетике [Текст] : [Сборник статей] : Пер. с англ. / С предисл. А. Н. Колмогорова ; Под ред. Р. Л. Добрушина и О. Б. Лупанова. —Москва : Изд-во иностранной литературы, 1963. —829 с. : черт.; 23 см (дата обращения 10.11.2022).
8. Введение в основы современных шифров с симметричным ключом [Электронный ресурс]. —Режим доступа: <https://intuit.ru/studies/courses/552/408/lecture/9360?page=6>, (дата обращения 21.11.2022).
9. Span<T> Структура [Электронный ресурс]. —Режим доступа: <https://learn.microsoft.com/en-us/dotnet/api/system.span-1?view=net-7.0>(дата обращения 1.12.2022).
10. Convert.TryFromBase64String(String, Span<Byte>, Int32) Method [Электронный ресурс]. —Режим доступа: <https://learn.microsoft.com/en-us/dotnet/api/system.convert.tryfrombase64string?view=net-7.0>(дата обращения 10.12.2022).

# Приложение

using System; using System.Collections.Generic; using System.Text; using System.Data; using System.Diagnostics; namespace AES\_encoder\_decoder { partial class AES { private int[,] sBox = new int[16, 16] { { 0x63 ,0x7c ,0x77 ,0x7b ,0xf2 ,0x6b ,0x6f ,0xc5 ,0x30 ,0x01 ,0x67 ,0x2b ,0xfe ,0xd7 ,0xab ,0x76}, { 0xca ,0x82 ,0xc9 ,0x7d ,0xfa ,0x59 ,0x47 ,0xf0 ,0xad ,0xd4 ,0xa2 ,0xaf ,0x9c ,0xa4 ,0x72 ,0xc0}, { 0xb7 ,0xfd ,0x93 ,0x26 ,0x36 ,0x3f ,0xf7 ,0xcc ,0x34 ,0xa5 ,0xe5 ,0xf1 ,0x71 ,0xd8 ,0x31 ,0x15}, { 0x04 ,0xc7 ,0x23 ,0xc3 ,0x18 ,0x96 ,0x05 ,0x9a ,0x07 ,0x12 ,0x80 ,0xe2 ,0xeb ,0x27 ,0xb2 ,0x75}, { 0x09 ,0x83 ,0x2c ,0x1a ,0x1b ,0x6e ,0x5a ,0xa0 ,0x52 ,0x3b ,0xd6 ,0xb3 ,0x29 ,0xe3 ,0x2f ,0x84}, { 0x53 ,0xd1 ,0x00 ,0xed ,0x20 ,0xfc ,0xb1 ,0x5b ,0x6a ,0xcb ,0xbe ,0x39 ,0x4a ,0x4c ,0x58 ,0xcf}, { 0xd0 ,0xef ,0xaa ,0xfb ,0x43 ,0x4d ,0x33 ,0x85 ,0x45 ,0xf9 ,0x02 ,0x7f ,0x50 ,0x3c ,0x9f ,0xa8}, { 0x51 ,0xa3 ,0x40 ,0x8f ,0x92 ,0x9d ,0x38 ,0xf5 ,0xbc ,0xb6 ,0xda ,0x21 ,0x10 ,0xff ,0xf3 ,0xd2}, { 0xcd ,0x0c ,0x13 ,0xec ,0x5f ,0x97 ,0x44 ,0x17 ,0xc4 ,0xa7 ,0x7e ,0x3d ,0x64 ,0x5d ,0x19 ,0x73}, { 0x60 ,0x81 ,0x4f ,0xdc ,0x22 ,0x2a ,0x90 ,0x88 ,0x46 ,0xee ,0xb8 ,0x14 ,0xde ,0x5e ,0x0b ,0xdb}, { 0xe0 ,0x32 ,0x3a ,0x0a ,0x49 ,0x06 ,0x24 ,0x5c ,0xc2 ,0xd3 ,0xac ,0x62 ,0x91 ,0x95 ,0xe4 ,0x79}, { 0xe7 ,0xc8 ,0x37 ,0x6d ,0x8d ,0xd5 ,0x4e ,0xa9 ,0x6c ,0x56 ,0xf4 ,0xea ,0x65 ,0x7a ,0xae ,0x08}, { 0xba ,0x78 ,0x25 ,0x2e ,0x1c ,0xa6 ,0xb4 ,0xc6 ,0xe8 ,0xdd ,0x74 ,0x1f ,0x4b ,0xbd ,0x8b ,0x8a}, { 0x70 ,0x3e ,0xb5 ,0x66 ,0x48 ,0x03 ,0xf6 ,0x0e ,0x61 ,0x35 ,0x57 ,0xb9 ,0x86 ,0xc1 ,0x1d ,0x9e}, { 0xe1 ,0xf8 ,0x98 ,0x11 ,0x69 ,0xd9 ,0x8e ,0x94 ,0x9b ,0x1e ,0x87 ,0xe9 ,0xce ,0x55 ,0x28 ,0xdf}, { 0x8c ,0xa1 ,0x89 ,0x0d ,0xbf ,0xe6 ,0x42 ,0x68 ,0x41 ,0x99 ,0x2d ,0x0f ,0xb0 ,0x54 ,0xbb ,0x16} }; private byte[] key; private byte[] message; public AES(byte[] key, byte[] message) { this.key = key; this.message = message; } private void SubBytes(ref int[,] state) { for (int j = 0; j <= 3; j++) { for (int i = 0; i <= 3; i++) { int row = state[i, j] / 16;//строка в sBox int col = state[i, j] % 16;//столбец в sBox state[i, j] = sBox[row, col];// замена через sBox } } } private void InvSubBytes(ref int[,] state) { for (int i = 0; i < 4; i++) { for (int j = 0; j < 4; j++) { int subbyte = state[i, j];// текущий элемент матрицы for (int m = 0; m < 16; m++)// поиск текущего эл-та в sBox { for (int n = 0; n < 16; n++) { if (sBox[m, n] == subbyte) { state[i, j] = n + m \* 16; } } } } } } private void ShiftRows(ref int[,] state) { for (int i = 0, depth = 0; i <= 3; i++, depth++)// задаётся кол-во шагов сдвига в соответствии со строкой { for (int d = 0; d < depth; d++) { int save = state[i, 0];//сохраним первый эл-т строки int j; for (j = 0; j < 3; j++) { state[i, j] = state[i, j + 1];//сдвиг влево } state[i, j] = save; } } } private void InvShiftRows(ref int[,] state) { for (int i = 0, depth = 0; i <= 3; i++, depth++)// задаётся кол-во шагов сдвига в соответствии со строкой { for (int d = 0; d < depth; d++) { int save = state[i, 3];//сохраним последний эл-т строки int j; for (j = 3; j > 0; j--)// сдвиг вправо { state[i, j] = state[i, j —1]; } state[i, j] = save; } } } private int GMul(int a, int b) {// Умножение 2 байтов в поле Галуа int p = 0; for (int counter = 0; counter < 8; counter++) { if ((b & 1) != 0) { p = p ^ a; } bool HighBit = (a & 0x80) != 0; a <<= 1; if (HighBit)// проверка установки старшего (8) бита { a = a ^ 0x1B; //mod x^8 + x^4 + x^3 + x + 1 } b >>= 1; } return p; } private void MixColumns(ref int[,] state) { int[,] ss = new int[4, 4]; for (int c = 0; c < 4; c++) { ss[0, c] = (byte)(GMul(0x02, state[0, c]) ^ GMul(0x03, state[1, c]) ^ state[2, c] ^ state[3, c]); ss[1, c] = (byte)(state[0, c] ^ GMul(0x02, state[1, c]) ^ GMul(0x03, state[2, c]) ^ state[3, c]); ss[2, c] = (byte)(state[0, c] ^ state[1, c] ^ GMul(0x02, state[2, c]) ^ GMul(0x03, state[3, c])); ss[3, c] = (byte)(GMul(0x03, state[0, c]) ^ state[1, c] ^ state[2, c] ^ GMul(0x02, state[3, c])); state[0, c] = ss[0, c]; state[1, c] = ss[1, c]; state[2, c] = ss[2, c]; state[3, c] = ss[3, c]; } } private void InvMixColumns(ref int[,] state) { int[,] ss = new int[4, 4]; for (int c = 0; c < 4; c++) { ss[0, c] = (byte)(GMul(0x0E, state[0, c]) ^ GMul(0x0B, state[1, c]) ^ GMul(0x0D, state[2, c]) ^ GMul(0x09, state[3, c])); ss[1, c] = (byte)(GMul(0x09, state[0, c]) ^ GMul(0x0E, state[1, c]) ^ GMul(0x0B, state[2, c]) ^ GMul(0x0D, state[3, c])); ss[2, c] = (byte)(GMul(0x0D, state[0, c]) ^ GMul(0x09, state[1, c]) ^ GMul(0x0E, state[2, c]) ^ GMul(0x0B, state[3, c])); ss[3, c] = (byte)(GMul(0x0B, state[0, c]) ^ GMul(0x0D, state[1, c]) ^ GMul(0x09, state[2, c]) ^ GMul(0x0E, state[3, c])); state[0, c] = ss[0, c]; state[1, c] = ss[1, c]; state[2, c] = ss[2, c]; state[3, c] = ss[3, c]; } } private void AddRoundKey(ref int[,] state, byte[] key) { for (int i = 0, z = 0; i <= 3; i++) { for (int j = 0; j <= 3; j++, z++) { state[i, j] = state[i, j] ^ key[z]; } } } public byte[] AESEncrypt() { int[,] state = new int[4, 4];// матрица состояний List<byte> encoded\_symbol\_sequence = new List<byte>(); int block\_counter = 0; Debug.WriteLine("Current message byte length is: {0}", message.Length); for (int array\_index = 0; array\_index < message.Length; array\_index += 16) { Debug.WriteLine("Current block number is: {0}", block\_counter); int current\_item\_index = 0; for (int i = 0; i <= 3; i++)//положить в матрицу блок сообщения { for (int j = 0; j <= 3; j++) { int calculated\_byte\_index = current\_item\_index + array\_index; if (calculated\_byte\_index >= message.Length) { state[j, i] = 0; } else { state[j, i] = (int)message[calculated\_byte\_index]; current\_item\_index += 1; } } } print\_current\_block\_state(state, "Unmodified block: "); // шифруем AddRoundKey(ref state, key); print\_current\_block\_state(state, "Added round key: "); for (int round = 1; round <= 9; round++) { Debug.WriteLine("Current round is: {0}", round); SubBytes(ref state); print\_current\_block\_state(state, "SubBytes: "); ShiftRows(ref state); print\_current\_block\_state(state, "ShiftRows: "); MixColumns(ref state); print\_current\_block\_state(state, "MixColumns: "); AddRoundKey(ref state, key); print\_current\_block\_state(state, "AddRoundKey: "); }; SubBytes(ref state); print\_current\_block\_state(state, "SubBytes: "); ShiftRows(ref state); print\_current\_block\_state(state, "ShiftRows: "); AddRoundKey(ref state, key); print\_current\_block\_state(state, "AddRoundKey: "); void print\_current\_block\_state(int[,] state, string debug\_message) { Debug.WriteLine(debug\_message); for (int i = 0; i <= 3; i++)//положить в матрицу блок сообщения { for (int j = 0; j <= 3; j++) { Debug.Write(state[i, j]); Debug.Write(" "); } Debug.WriteLine(""); } Debug.WriteLine("------------------"); } print\_current\_block\_state(state,"Encrypted block: "); block\_counter += 1; for (int j = 0; j <= 3; j++) { for (int i = 0; i <= 3; i++) { encoded\_symbol\_sequence.Add((byte)state[i, j]); } } } byte[] finalized\_encoded\_symbol\_sequence = encoded\_symbol\_sequence.ToArray(); return finalized\_encoded\_symbol\_sequence; } public byte[] AESDecrypt() { int[,] state = new int[4, 4];// матрица состояний List<byte> decrypted\_symbol\_sequence = new List<byte>(); Debug.WriteLine("Decrypted blocks: "); for (int array\_index = 0; array\_index < message.Length; array\_index += 16) { int current\_item\_index = 0; for (int i = 0; i <= 3; i++)//положить в матрицу блок сообщения { for (int j = 0; j <= 3; j++) { int calculated\_byte\_index = current\_item\_index + array\_index; if (calculated\_byte\_index >= message.Length) { state[j, i] = 0; } else { state[j, i] = (int)message[calculated\_byte\_index]; current\_item\_index += 1; } } } void print\_current\_block\_state() { for (int i = 0; i <= 3; i++)//положить в матрицу блок сообщения { for (int j = 0; j <= 3; j++) { Debug.Write(state[i, j]); Debug.Write(" "); } Debug.WriteLine(""); } Debug.WriteLine("------------------"); } print\_current\_block\_state(); // дешифруем AddRoundKey(ref state, key); InvShiftRows(ref state); InvSubBytes(ref state); for (int round = 1; round <= 9; round++) { AddRoundKey(ref state, key); InvMixColumns(ref state); InvShiftRows(ref state); InvSubBytes(ref state); }; AddRoundKey(ref state, key); for (int j = 0; j <= 3; j++) { for (int i = 0; i <= 3; i++) { decrypted\_symbol\_sequence.Add((byte)state[i, j]); } } } return decrypted\_symbol\_sequence.ToArray(); } } }