**Министерство науки и высшего образования Российской Федерации**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Факультет безопасности информационных технологий**

**Дисциплина:**

«Криптографические методы обеспечения информационной безопасности»

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №3**

«Основные структурные элементы алгоритма AES»

**Выполнил:**

Изображение выглядит как зарисовка, Штриховая графика, Шрифт, рукописный текст

Автоматически созданное описаниеПолевцов Артем Сергеевич, студент группы N34511

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(подпись)

**Проверил:**

Волков Александр Григорьевич, инженер ФБИТ

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(отметка о выполнении)

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(подпись)

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc154059299)

[1 ОСНОВНЫЕ СТРУКТУРНЫЕ ЭЛЕМЕНТЫ АЛГОРИТМА AES 4](#_Toc154059300)

[1.1 Ход работы 4](#_Toc154059301)

[1.1.1 Изучение алгоритма шифрования AES при помощи AES Visualization в Cryptool 4](#_Toc154059302)

[1.1.2 Демонстрация лавинного эффекта: 9](#_Toc154059303)

[1.1.3 Проведение атаки на основе известного открытого текста 10](#_Toc154059304)

[1.1.4 Проведение дифференциального криптоанализа 11](#_Toc154059305)

[1.1.5 Ручное выполнение 1 цикла раундовой функции 14](#_Toc154059306)

[1.1.6 Шифрование файла с помощью библиотеки OpenSSL 18](#_Toc154059307)

[ЗАКЛЮЧЕНИЕ 21](#_Toc154059308)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 22](#_Toc154059309)

# ВВЕДЕНИЕ

Цель работы - изучить основные принципы работы алгоритмы AES.

В задачи данной лабораторной работы:

* Проанализировать эмуляцию алгоритма AES и примитивных атак на шифр, используя Cryptool 2. Выделить основные необходимые настройки шифра и требуемые ограничения на параметры;
* Выполненить 1 цикла раундовой функции алгоритма AES вручную. Отразить промежуточные результаты шифрования после всех этапов алгоритма AES. Дать математическое обоснование для каждой операции. Также для подробного изучения шифра может быть использована программная реализация 1 раунда (или полной системы) AES в режиме отладки с выводом промежуточных значений шифрования;
* Проанализировать принципы использования криптосистемы в современных приложениях на примере библиотеки openssl;

# ОСНОВНЫЕ СТРУКТУРНЫЕ ЭЛЕМЕНТЫ АЛГОРИТМА AES

## Ход работы

### Изучение алгоритма шифрования AES при помощи AES Visualization в Cryptool

Я оставил ключ и открытый текст по умолчанию:

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, Операционная система

Автоматически созданное описание

Рисунок 1 - AES Visualization в Cryptool 2

Расширение ключа:

Из исходных байтов ключа была построена матрица 4 на 4

Изображение выглядит как число, снимок экрана, прямоугольный, текст

Автоматически созданное описание

Рисунок 2 – State matrix

Здесь мы взяли последний столбец матрицы и сдвинули все байты в этом столбце на один вверх так, чтобы последний стал первым, предпоследний – последним и так далее.

Изображение выглядит как текст, снимок экрана, число, диаграмма

Автоматически созданное описание

Рисунок 3 – State matrix

Далее по матрице s-box, в которой название столбца – это левая часть байта, а строки – правая.

Изображение выглядит как текст, снимок экрана, линия, Параллельный

Автоматически созданное описание

Рисунок 4 – Преобразование столбца матрицы

Таким образом, получаем новый столбец из байтов исходного:

Изображение выглядит как снимок экрана, число, Прямоугольник, линия

Автоматически созданное описание

Рисунок 5 – Новый столбец

Далее к полученному столбцу применяется операция сложения с константой, которая соответствует номеру этапа расширения ключа.

Изображение выглядит как прямоугольный, диаграмма, Прямоугольник, снимок экрана

Автоматически созданное описание

Рисунок 6 – Формирование результирующей матрицы

Далее мы берем первый столбец матрицы и применяем операцию xor с получившимся в предыдущем шаге столбцом.

Изображение выглядит как текст, снимок экрана, диаграмма, Прямоугольник

Автоматически созданное описание

Рисунок 7 – Формирование результирующей матрицы

Далее применяем операцию xor для следующего столбца исходной матрицы и крайнего полученного столбца результирующей матрицы.

Изображение выглядит как текст, снимок экрана, диаграмма, Прямоугольник

Автоматически созданное описание

Рисунок 8 – Формирование результирующей матрицы

И так повторяем для всех столбцов, в итоге получаем новую матрицу 4 на 4:

Изображение выглядит как текст, снимок экрана, диаграмма, число

Автоматически созданное описание

Рисунок 9 – Формирование результирующей матрицы

На следующем этапе проделываем те же самые операции, только полученная на предыдущем этапе матрица становится исходной:

Изображение выглядит как текст, снимок экрана, Параллельный, линия

Автоматически созданное описание

Рисунок 10 – Начало второго раунда

На первом этапе шифрования мы построчно слева-направо сверху-вниз применяем операцию xor для каждого байта исходного сообщения и ключа.

Изображение выглядит как текст, снимок экрана, диаграмма, Прямоугольник

Автоматически созданное описание

Рисунок 11 – Операция xor

Далее на этапе Sub Bytes мы так же идем слева-направо и сверху-вниз по строкам получившейся на предыдущем шаге матрицы и заменяем каждый байт по матрице S-Box

Изображение выглядит как текст, снимок экрана, число, диаграмма

Автоматически созданное описание

Рисунок 12 – Этап Sub Bytes

На этапе Shift Row сначала вторая строка сдвигается один раз влево. Затем третья строка дважды сдвигается влево и, наконец, четвертая строка сдвигается три раза влево. Перекрывающиеся байты переносятся вправо, образуя матрицу 4 x 4.

Изображение выглядит как текст, диаграмма, снимок экрана, линия

Автоматически созданное описание

Рисунок 13 - Этап Shift Row

Изображение выглядит как текст, снимок экрана, число, прямоугольный

Автоматически созданное описание

Рисунок 14 - Этап Shift Row

На этапе Mix Col мы выполняем умножение каждого столбца, полученной на предыдущем этапе матрицы на специальную матрицу:

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Рисунок 15 - Этап Mix Col

Далее на этапе Add key мы меняем в полученной на прошлом этапе матрице столбцы и строки местами и производим операцию xor для каждого байта этой матрицы с каждым байтом ключа:

Изображение выглядит как текст, снимок экрана, Прямоугольник, число

Автоматически созданное описание

Рисунок 16 - Этап Add Key

### Демонстрация лавинного эффекта:

Для демонстрации лавинного эффекта я изменил случайный бит входного сообщения:

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 17 – Демонстрация лавинного эффекта

Далее, пройдя десять раундов, мы можем видеть, что на третьем раунде произошел скачок с 7% искаженных байтов до 49,2% и далее это значение остается примерно одинаковым, максимальный процент искаженных бит мы можем видеть на восьмом раунде 57%.

Можем сделать выводы, что такое значение является очень хорошим так как сильно искажает большое количество конечных данных при минимальном количестве изменений во входных.

Изображение выглядит как текст, снимок экрана, компьютер, программное обеспечение

Автоматически созданное описание

Рисунок 18 – Результат лавинного эффекта

### Проведение атаки на основе известного открытого текста

Криптоанализ алгоритма AES с использованием компонента KeySearcher и известного фрагмента открытого текста (шпаргалки). KeySearcher получает на вход зашифрованный текст и пробует все ключи данного пространства ключей. Полученные открытые тексты затем сопоставляются с шпаргалкой. На выход отправляется открытый текст, который лучше всего соответствует шпаргалке.

Укажем в настройках анализа 13 байт ключа для более быстрого получения результата из 16 байт.

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, веб-страница

Автоматически созданное описание

Рисунок 19 – Алгоритм KeySearcher

На 12 ядрах процесс подбора ключа занял около двух минут.

### Проведение дифференциального криптоанализа

Если операция XOR используется для шифрования сообщения m с ключом k, зашифрованный текст c получается следующим образом:

Изображение выглядит как Шрифт, текст, каллиграфия, символ

Автоматически созданное описание

Рисунок 20 – Операция XOR

Основная идея дифференциального криптоанализа заключается в использовании эффекта свойства операции XOR путем двукратного использования ключа k. Если добавление ключа выполняется дважды, применяется следующее:

Изображение выглядит как Шрифт, текст, рукописный текст, каллиграфия

Автоматически созданное описание

Рисунок 21 – Эффект свойства операции XOR

Мы получаем обратно исходное сообщение m.

Оператор XOR часто называют оператором разности. Это имя также используется в этом уроке.

Дифференциальный криптоанализ — это так называемая «атака с выбранным открытым текстом», которая означает, что криптоаналитик может выбирать сообщения с открытым текстом и шифровать их, чтобы иметь совпадающие пары открытый текст/зашифрованный текст.

Мы воспользуемся идеей использования ключа дважды, используя пары сообщений m1 и m2. Если теперь разница между зашифрованными текстами образовалась, справедливо следующее:

Изображение выглядит как текст, Шрифт, диаграмма, дизайн

Автоматически созданное описание

Рисунок 22 – Разница между зашифрованными текстами

Следует:

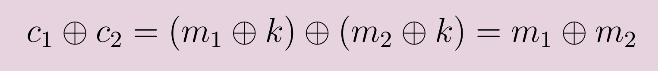


Рисунок 23 – Операция XOR

Мы получаем разницу открытых текстов, вычисляя разницу зашифрованных текстов. Это можно использовать для получения информации о раундовых ключах в DCA.

Здесь мы применяем идею DCA к Chiffre 1. Структура шифра обычно общедоступна, поэтому известны все компоненты и их функциональность. Безопасность шифра никогда не должна основываться на секретности конструкции, а на секретности ключа. Это требование еще называют принципом Керкгофа.

Здесь мы применяем идею DCA к Chiffre 1. Структура шифра обычно общедоступна, поэтому показаны все компоненты и их функциональность. Безопасность шифра никогда не должна основываться на секретности конструкции, а также на секретности переключателя. Это требование по-прежнему называют принципом Керкгофа.

Изображение выглядит как Шрифт, типография, текст, белый

Автоматически созданное описание

Рисунок 24 – Принцип Керкгофа

Промежуточный результат u затем заменяется S-блоками:

Изображение выглядит как Шрифт, текст, Графика, белый

Автоматически созданное описание

Рисунок 25 – Замена S-блоками

Наконец, снова происходит добавление ключей (промежуточный результат v указан с добавлением k1):

Изображение выглядит как Шрифт, символ, каллиграфия, белый

Автоматически созданное описание

Рисунок 26 – Добавление ключей

Внутренние переменные u и v шифра имеют большое значение в дальнейшем, поскольку мы используем эти промежуточные результаты для восстановления неизвестных ключей k1 и k0.

На следующем рисунке обобщен процесс шифрования, описанный выше на слайдах 11–16.

Изображение выглядит как текст, диаграмма, План, Технический чертеж

Автоматически созданное описание

Рисунок 27 – Обобщенный процесс шифрования

Криптоаналитику известны параметры m и c, но не внутренние промежуточные значения u и v, поскольку k0 и k1 неизвестны. Однако криптоаналитик может вычислить разницу между двумя внутренними промежуточными значениями u на основе разницы между двумя открытыми текстами. Далее мы рассмотрим два сообщения m1 и m2 и построим их разницу:



Рисунок 28 – Разница двух сообщений

Эти знания можно использовать для расчета k1. Для этого мы рассмотрим две пары открытого текста и зашифрованного текста (m1, c1) и (m2, c2). Согласно уравнению (1), разность значений u равна разнице m1 и m2. Далее «советуется» k1 так, чтобы из



Рисунок 29 – Расчет c

Поскольку функциональность S-Box является общедоступной и, следовательно, обратимой, криптоаналитик может выполнять вычисления с помощью

Изображение выглядит как Шрифт, каллиграфия, белый, текст

Автоматически созданное описание

Рисунок 30 – Вычисления

Изображение выглядит как Шрифт, текст, каллиграфия, белый

Автоматически созданное описание

Рисунок 31 – Значения

Однако эти значения нельзя напрямую сравнивать с внутренними значениями u1 и u2, поскольку они неизвестны.

Из уравнения (1) следует, что если ключ раунда k1 угадан правильно, справедливо следующее:

Изображение выглядит как Шрифт, текст, рукописный текст, каллиграфия

Автоматически созданное описание

Рисунок 32 – Уравнение

Изображение выглядит как текст, Шрифт, символ, белый

Автоматически созданное описание

Рисунок 33 – Операция XOR

уже известно. Как указано на предыдущей странице, значения k1 угаданы. Здесь объясняется, как ограничить набор возможных правильных значений: Криптоаналитик пробует все значения t из k1, и если

Изображение выглядит как текст, Шрифт, рукописный текст, каллиграфия

Автоматически созданное описание

Рисунок 34 – Ограничение набора возможных правильных значений

верно, t считается кандидатом на k1. Если в конце остается более одного кандидата, атака повторяется с новыми парами открытого текста и зашифрованного текста.

После успешного восстановления k1 можно сразу вычислить k0. Это можно сделать, рассчитав уравнение

Изображение выглядит как Шрифт, рукописный текст, текст, белый

Автоматически созданное описание

Рисунок 35 – Вычисление k0

Будет определено.

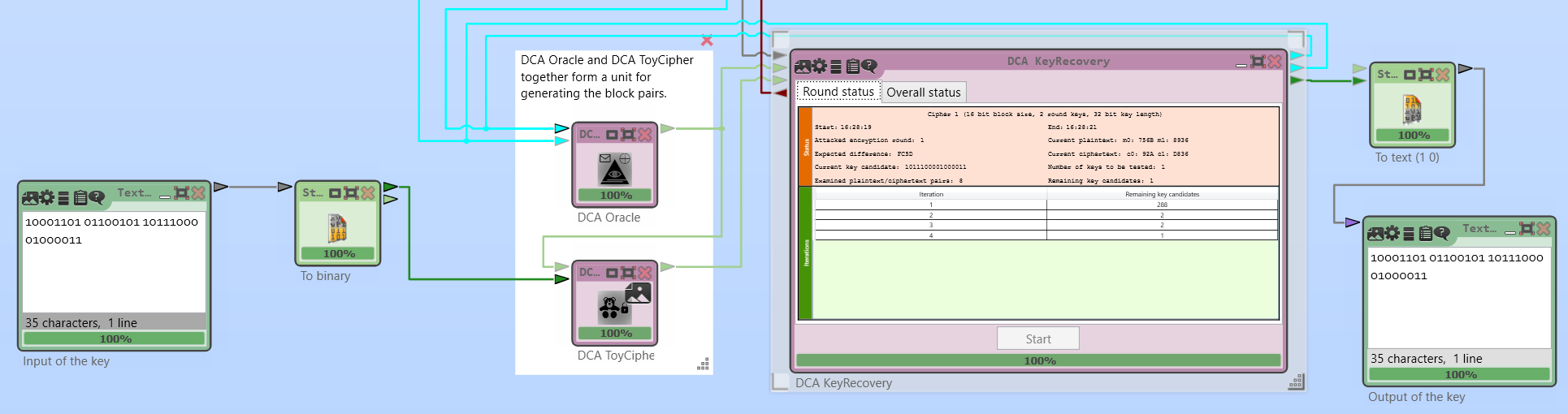


Рисунок 36 – Результат криптоанализа

### Ручное выполнение 1 цикла раундовой функции

Входное сообщение: 81 e4 2a f6 a4 0b 5b 05 f2 cf 0a 96 f2 e2 33 e0

Ключ: df 9b 26 79 74 05 60 62 1d 9e 2d 39 05 a4 67 59

Изображение выглядит как снимок экрана, число, линия, Шрифт

Автоматически созданное описание  
Рисунок 37 - Входное сообщение

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание  
Рисунок 38 - Ключ

Расширение ключа:

Берем последний столбец исходной матрицы, сдвигаем каждый байт на один вверх и заменяем через матрицу S-box:

Изображение выглядит как снимок экрана, линия, число, текст

Автоматически созданное описание

Рисунок 40 – Промежуточные вычисления

Далее производим операцию xor с предыдущим столбцом, в данном случае с константой:

Изображение выглядит как снимок экрана, текст, линия, число

Автоматически созданное описание

Рисунок 41 – Промежуточные вычисления

Далее берем первый столбец исходной матрицы и производим xor его с получившимся на прошлом шаге:

Изображение выглядит как снимок экрана, текст, линия, число

Автоматически созданное описание

Рисунок 42 – Промежуточные вычисления

Проводим аналогичные действия, только со вторым столбцом исходной матрицы и крайним непустым столбцом результирующей матрицы:

Изображение выглядит как снимок экрана, текст, линия, число

Автоматически созданное описание

Рисунок 43 – Промежуточные вычисления

И так со всеми оставшимися столбцами исходной матрицы:

Изображение выглядит как текст, снимок экрана, линия, число

Автоматически созданное описание

Рисунок 44 – Промежуточные вычисления

Изображение выглядит как снимок экрана, текст, число, линия

Автоматически созданное описание

Рисунок 45 – Промежуточные вычисления

В итоге получаем результирующую матрицу:

Изображение выглядит как снимок экрана, текст, Шрифт, число

Автоматически созданное описание

Рисунок 46– Промежуточные вычисления

Побитовое сложение с ключом:

Применим операцию xor для каждого байта открытого текста и каждого байта ключа:

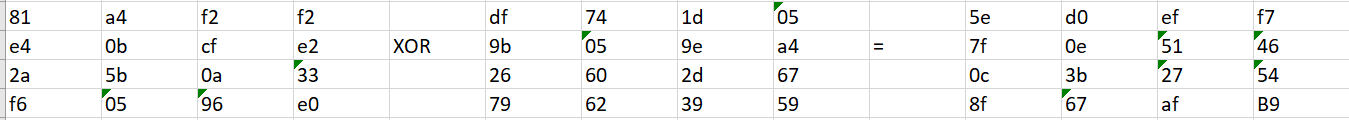


Рисунок 47 – Промежуточные вычисления

Операция SubBytes:

В результате операции получаем матрицу:

Изображение выглядит как снимок экрана, текст, число, Шрифт

Автоматически созданное описание

Рисунок 48 – Промежуточные вычисления

Операция ShiftRows:

Первая строка остается на месте, вторую сдвигаем на один элемент, третью -на два, четвертую на три:

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 49 – Промежуточные вычисления

Получаем:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 50 – Промежуточные вычисления

Операция MixColumns:

Изображение выглядит как снимок экрана, линия, Шрифт, текст

Автоматически созданное описание

Рисунок 51 – Промежуточные вычисления

Произведем операцию для каждой строки:

Изображение выглядит как снимок экрана, текст, линия, число

Автоматически созданное описание

Рисунок 52 – Промежуточные вычисления

Изображение выглядит как снимок экрана, линия, текст, число

Автоматически созданное описание

Рисунок 53 – Промежуточные вычисления

Изображение выглядит как снимок экрана, линия, График, число

Автоматически созданное описание

Рисунок 54 – Промежуточные вычисления

Изображение выглядит как снимок экрана, линия, число, Шрифт

Автоматически созданное описание

Рисунок 55 – Промежуточные вычисления

В итоге получаем следующую последовательность байт:

Изображение выглядит как снимок экрана, текст, Шрифт, число

Автоматически созданное описание

Рисунок 56 – Промежуточные вычисления

### Шифрование файла с помощью библиотеки OpenSSL

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 57 – Список всех алгоритмов, включающих aes

-aes-128-ofb:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 58 – АЕS-128-ofb

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 59 – АЕS-128-ofb

-aes-256-ecb:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 60 – АЕS-256-ecb

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 61 – АЕS-256-ecb

-aes-192-cfb:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 62 – АЕS-192-cfb

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 63 – АЕS-192-cfb

-salt Use salt in the KDF (default)

-a Base64 encode/decode, depending on encryption flag

-pbkdf2 Use password-based key derivation function 2 (PBKDF2)

-k val Passphrase

-iter +int Specify the iteration count and force the use of PBKDF2

# ЗАКЛЮЧЕНИЕ

По результатам выполнения данной лабораторной работы я изучил, проанализировал и реализовал один цикл раундовой функции алгоритма AES. Также был произведен криптоанализ данного алгоритма. Были проанализированы принципы использования криптосистемы в современных приложениях на примере библиотеки openssl. В реализации одного цикла раундовой функции были отражены промежуточные результаты шифрования после всех этапов алгоритма AES.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Бабенко, Л. К. Современные алгоритмы блочного шифрования и методы их анализа / Л.К. Бабенко, Е.А. Ищукова. - М.: Гелиос АРВ, 2015. - 376 c.
2. Бабенко, Л.К. Современные интеллектуальные пластиковые карты / Л.К. Бабенко. - М.: Гелиос АРВ, 2015. - 921 c.
3. Болотов, А. А. Элементарное введение в эллиптическую криптографию. Протоколы криптографии на эллиптических кривых / А.А. Болотов, С.Б. Гашков, А.Б. Фролов. - М.: КомКнига, 2012. - 306 c.
4. Бузов, Геннадий Алексеевич Защита информации ограниченного доступа от утечки по техническим каналам / Бузов Геннадий Алексеевич. - М.: Горячая линия - Телеком, 2016. - 186 c.