**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Факультет безопасности информационных технологий**

**Дисциплина:**

«Криптографические методы обеспечения информационной безопасности»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3**

«Основные структурные элементы алгоритма AES»

**Выполнил:**

Молитвин Илья Алексеевич, студент группы N3Изображение выглядит как текст, соединитель

Контент, сгенерированный ИИ, может содержать ошибки.345

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

(подпись)

**Проверил:**

Таранов Сергей Владимирович

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

(отметка о выполнении)

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

(подпись)

**Оглавление**

[Введение 3](#_Toc162728157)

[Ход работы 4](#_Toc162728158)

[1. Визуализация алгоритма AES 4](#_Toc162728159)

[2. Визуализация «лавинного эффекта» AES 11](#_Toc162728160)

[3. Атака на основе известного открытого текста 12](#_Toc162728161)

[4. Дифференциальный анализ на блочные симметричные криптосистемы 12](#_Toc162728162)

[5. Цикл раундовой функции вручную 16](#_Toc162728163)

[Шифрование файла с помощью криптографической библиотеки OpenSSL 21](#_Toc162728164)

[Вывод 25](#_Toc162728165)

# Введение

Цель работы – изучить основные принципы работы алгоритмы AES.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Проанализировать эмуляцию алгоритма AES и примитивных атак на шифр,

используя Cryptool 2. Выделить основные необходимые настройки шифра и

требуемые ограничения на параметры.

1. Выполнить 1 цикл 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.

1. Проанализировать принципы использования криптосистемы в современных

приложениях на примере библиотеки openssl.

# Ход работы

## Визуализация алгоритма AES

Для визуализации работы алгоритма воспользуемся шаблоном в CrypTool 2, куда вставим входные данные и ключ.

Входные данные: 697420697320736563726574206b6579

128-битный ключ: 7468697320697320616573206c616261

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 1 - Общий вид визуализации AES

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

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2 - Схема работы AES

Сначала происходит этап расширения ключа - генерация подключей, которые далее будут использоваться на каждом из раундов шифрования (их может быть 10, 12 или 14 - в нашем случае 10). Также стоит отметить, что ключей используется на 1 больше, чем раундов - мастер-ключ нужен для процедуры “отбеливания” (0-й раунд).

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

Автоматически созданное описание

Рисунок 3 - Ключ в виде матрицы состояний

Введенный ключ записывается в матрицу состояний размера, в нашем случае, 4 \* 4. Берем последние 4 байта (последний столбец) - верхний байт перемещается вниз (остальные 3, соответственно, сдвигаются).

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

Автоматически созданное описание

Рисунок 4 - Изменение матрицы состояний

Далее каждый байт из матрицы состояний заменяется в соответствии с S-boxом (таблицей замены): ячейка со значением выбирается по значениям 1й (строка) и 2й (столбец) половины каждого байта.

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 5 - Замена значения из S-box

Далее происходит добавление раундовой константы к каждому значению полученных 4х байтов. С полученными значениями XOR-ятся (побайтово) соответствующие значения первых 4х байтов исходного ключа - результат записывается в матрицу.

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

Автоматически созданное описание

Рисунок 6 - Финальные значения

Для дальнейшего заполнения матрицы необходимо для каждого нового столбца произвести побайтовый XOR значений предыдущего столбца со значениями столбца с таким же номером из матрицы состояний.

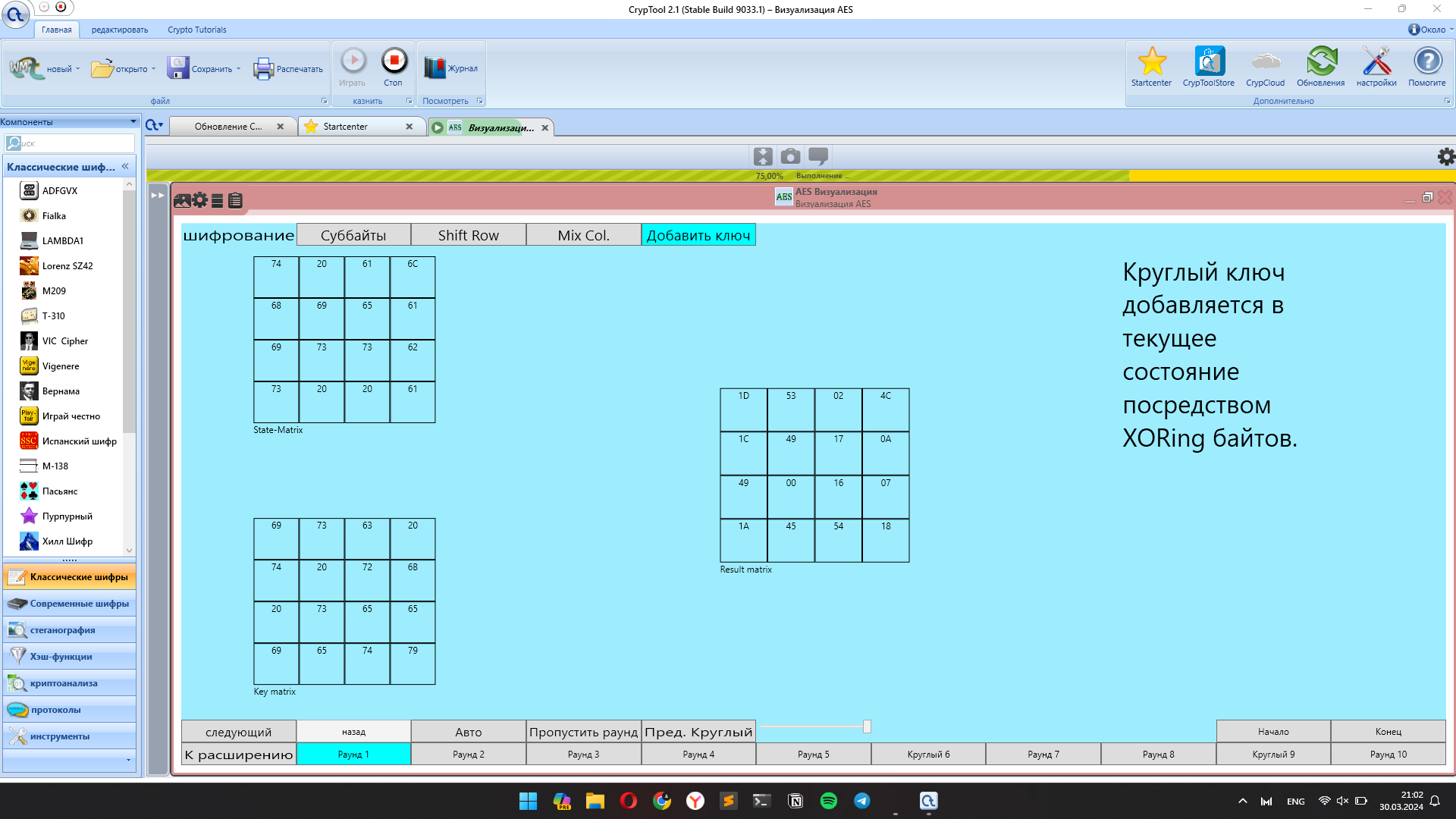


Рисунок 7 - Начальные и конечные значения

Переходим к этапу шифрования. В начале происходит - XOR мастер-ключа с входной последовательностью (текстом).

Далее идет операция SubBytes, главная задача которой - обеспечение свойства confusion. Аналогично этапу расширения ключа, она представляет собой замену по S-box.

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 8 - Sub Bytes

Переходим к операции ShiftRows: 1я строка остается без изменений, 2я сдвигается на одну ячейку, 3я - на две и 4я - на три. “Выдвинутые” байты затем встраиваются обратно в матрицу, дополняя ее до размера 4 \* 4.

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 9 - Shiftrow

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 10 - Shiftrow

Следующая операция - MixColumn. Последовательно берутся столбцы, каждый из которых перемножается с каждой строкой матрицы умножения, результат записывается в матрицу.

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 11 - Mix Columns

Заключительная операция раунда - AddKey - представляет собой XOR полученных значений с соответствующим раундовым ключом.

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 12 - AddKey

Далее полученная матрица подается на вход следующего раунда, и повторяется аналогичная последовательность действий. Все раунды одинаковые, за исключением последнего - там отсутствует операция MixColumn.

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 13 - Результат шифрования

## Визуализация «лавинного эффекта» AES

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

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 14 - Лавинный эффект при искажении одного бита

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

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 15 - Лавинный эффект при искажение нескольких битов

На Рисунках 14 и 15 видно, что к концу выполнения алгоритма лавинный эффект достигает в среднем 50% даже от изменения всего одно бита. При одном искаженном бите явные изменения достигаются на втором раунде, а при нескольких уже на первом.

## Атака на основе известного открытого текста

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

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 16 - Known-plaintext attack

На Рисунке 16 можно увидеть процесс выполнения атаки на основе известного открытого текста. Данная атака требует перебора возможного ключа по известным шифротекстам и отрытым текстам и частотного анализа для понимания насколько текст по подобранному ключу соответствует языку исходного текста, но данная атака не очень эффективна для современных алгоритмов, потому что требует большое количество пар шифротекстов и открытых текстов. Иначе перебор за нормальное время просто невозможен.

Я использовал часть текста и часть ключа, чтобы это заняло меньше времени

## Дифференциальный анализ на блочные симметричные криптосистемы

Дифференциальный криптоанализ (ДК) — метод криптоанализа симметричных блочных шифров (и других криптографических примитивов, в частности, хеш-функций и поточных шифров).

ДКз основан на изучении преобразования разностей между шифруемыми значениями на различных раундах шифрования. В качестве разности, как правило, применяется операция побитового суммирования по модулю 2, хотя существуют атаки и с вычислением разности по модулю 2^32. Является статистической атакой. Применим для взлома DES, FEAL и некоторых других шифров, как правило, разработанных ранее начала 90-х. Количество раундов современных шифров (AES, Camellia и др.) рассчитывалось с учётом обеспечения стойкости, в том числе и к дифференциальному криптоанализу.

Анализу подвергаются зависимости между парами блоков данных до и после применения шифра. ДК-атаки становятся возможны тогда, когда можно установить проникновение таких зависимостей сквозь почти все ( за исключением 2-3) раунды криптопреобразования.

Изображение выглядит как текст, программное обеспечение, диаграмма, Значок на компьютере

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 17 - Выбор S-боксов для проведение дальнейшего анализа

Рассмотрим изменение разностей в S-боксах.

Чем больше ключевых битов выбрано, тем сложнее поиск дифференциала. Однако может иметь смысл атаковать несколько S-боксов одновременно. В зависимости от характера шифра могут быть найдены лучшие или худшие различия.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 18 - Информация о возможных характеристиках после анализа

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

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 19 - Результаты анализа

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

Автоматически созданное описание

Рисунок 19 - Визуализация возможного пути и различий в значениях

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 20 - Список возможных ключей

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

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 21 - Второй выбор S-боксов

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 22 - Количество возможных ключей на каждой итерации

Спустя три итерации получилось успешной найти ключ.

## Цикл раундовой функции вручную

Ключ:

|  |  |  |  |
| --- | --- | --- | --- |
| 04 | 52 | a3 | 38 |
| 82 | e8 | 74 | ea |
| cd | 6a | b4 | 00 |
| 36 | 35 | 56 | 00 |

**Key Expansion.**

Сперва делаем сдвиг последнего столбца:

|  |
| --- |
| ea |
| 00 |
| 00 |
| 38 |

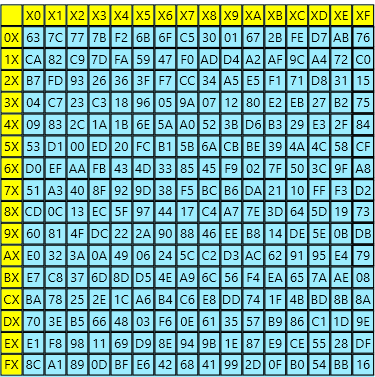


Рисунок 23 – Таблица S-Box

Далее с помощью S-box, получаем новое 4-байтовое слово:

|  |
| --- |
| 87 |
| 63 |
| 63 |
| 07 |

Далее выполняем битовое сложение этого слова с раундовой константой и получаем то, что содержится в третьем столбце:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 01 | ⊕ | 87 | = | 86 |
| 00 | 63 | 63 |
| 00 | 63 | 63 |
| 00 | 07 | 07 |

Далее битовое сложение получившегося слова с первым столбцом матрицы ключа:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 04 | ⊕ | 86 | = | 82 |
| 82 | 63 | e1 |
| cd | 63 | ae |
| 36 | 07 | 91 |

Далее со получившийся столбец со вторым столбцом матрицы:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 52 | ⊕ | 82 | = | d0 |
| e8 | e1 | 09 |
| 6a | ae | c4 |
| 35 | 91 | 04 |

Далее со получившийся столбец с третьим столбцом матрицы:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a3 | ⊕ | d0 | = | 73 |
| 74 | 09 | 7d |
| b4 | c4 | 70 |
| 56 | 04 | 52 |

Далее со получившийся столбец с четвертым столбцом матрицы:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 38 | ⊕ | 73 | = | 4b |
| ea | 7d | 97 |
| 00 | 70 | 70 |
| 00 | 52 | 52 |

Получившийся раундовый ключ:

|  |  |  |  |
| --- | --- | --- | --- |
| 82 | d0 | 73 | 4b |
| e1 | 09 | 7d | 97 |
| ae | c4 | 70 | 70 |
| 91 | 04 | 52 | 52 |

State перед циклом раундовой функции:

|  |  |  |  |
| --- | --- | --- | --- |
| 33 | ea | 7b | 00 |
| 00 | 78 | ea | 7d |
| f7 | aa | 5e | 00 |
| af | 3c | fa | 00 |

**SubBytes.**

С помощью таблицы S-Box на Рисунке 18 получаем новую матрицу state:

|  |  |  |  |
| --- | --- | --- | --- |
| с3 | 87 | 21 | 63 |
| 63 | 21 | 87 | ff |
| 68 | ac | 58 | 63 |
| 79 | de | 2d | 63 |

**ShiftRow**.

В результате сдвига получаем новую матрицу state:

|  |  |  |  |
| --- | --- | --- | --- |
| с3 | 87 | 21 | 63 |
| 21 | 87 | ff | 63 |
| 58 | 63 | 68 | ac |
| 63 | 79 | de | 2d |

**MixColumns.**

Умножим мультипликативную матрицу и матрицу состояний

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 1 | 1 | \* | с3 | 87 | 21 | 63 | = | c5 | e7 | 4d | 97 |
| 1 | 2 | 3 | 1 | 21 | 87 | ff | 63 | 96 | c6 | c5 | 07 |
| 1 | 1 | 2 | 3 | 58 | 63 | 68 | ac | 37 | 43 | 96 | 6b |
| 3 | 1 | 1 | 2 | 63 | 79 | de | 2d | e2 | 67 | 34 | 30 |

Пример вычисления первой ячейки матрицы: 2\*с3+3\*21+1\*58+1\*63

Представим все числа в полиномиальном представлении GF(2^8):

1 = 1

2 = x

3 = x+1

c3 = x^7+x^6+x+1

21 = x^5+1

58 = x^6+x^4+x^3

63 = x^6+x^5+x+1

2\*c3 = x(x^7+x^6+x+1) = x^8+x^7+x^2+x = x^4+x^3+**x**+1+x^7+x^2+**x** = x^7+x^4+x^3+x^2+1

3\*21=(x+1)\*(x^5+1) = x^6+x^5+x+1

Тогда итоговый полином: x^7+**x^4**+**x^3**+x^2+**1**+x^6+**x^5**+**x**+**1**+**x^6**+**x^4**+**x^3**+**x^6**+**x^5**+**x**+1 = x^7+x^6+x^2+1 = c516

**AddRoundKey.**

Выполним битовое сложение матрицы state и раундового ключа

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| c5 | e7 | 4d | 97 | ⊕ | 82 | d0 | 73 | 4b | = | 47 | 37 | 3e | dc |
| 96 | c6 | c5 | 07 | e1 | 09 | 7d | 97 | 30 | cf | 3f | f0 |
| 37 | 43 | 96 | 6b | ae | c4 | 70 | 70 | 82 | 01 | e6 | 44 |
| e2 | 67 | 34 | 30 | 91 | 04 | 52 | 52 | 06 | 03 | 39 | 62 |

Цикл раундовой функции окончен.

Операция InversionMixColumn является обратной к MixColumn и используется для расшифровки. Для проведения данного этапа берется мультипликативная матрица и выводится обратная ей.

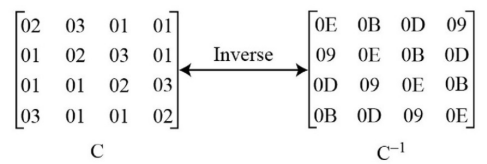


Рисунок 24 – Матрица обратная мультипликативной

## Шифрование файла с помощью криптографической библиотеки OpenSSL

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 25 – Исходный текст

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 26 – Шифрование AES CBC

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 27 – Результат шифрование AES CBC

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 28 – Дешифрование AES CBC

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

Рисунок 29 – Результат дешифрования AES CBC

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 30 – Шифрование AES CFB

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 31 – Дешифрование AES CFB

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 32 – Шифрование и дешифрование AES OFB

Дополнительные параметры шифрования AES в OpenSSL:

-salt – соль, которая добавляется к исходным данных. Она должна быть представлена в виде строки из шестнадцатеричных цифр. Если этот параметр используется при шифровании, то при расшифровке снова потребуется то же самое точное значение.

-a – если выполняется шифрование, то данные после шифрования кодируются в формате base64. Если задано дешифрование, то входные данные перед дешифрованием декодируются в формате base64.

-k – Пароль, из которого можно получить ключ. Это сделано для совместимости с предыдущими версиями OpenSSL.

-iter – использует указанное число итераций для пароля при получении ключа шифрования. Высокие значения увеличивают время, необходимое для взлома пароля брут-форсом зашифрованного файла.

# Вывод

В ходе выполнения данной лабораторной работы был подробно изучен алгоритм AES: как с точки зрения теории, так и на практике (с обзором в Cryptool 2 и ручным исполнением 1 раунда).

Были рассмотрены лавинный эффект и его визуализация, и проведены атака на основе известного текста и дифференциальный криптоанализ.

Наконец было протестировано шифрование файла в Openssl алгоритмами, включающими DES, и различные его вариации, достигаемые благодаря добавлению определенных параметров в команды.