Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчёт по лабораторной работе №2 по курсу «Защита информации»

Тема Реализация DES

Студент Волков Г.В.

Группа ИУ7-71Б

Оценка (баллы)

Преподаватели Чиж И. С.

Москва — 2023 г.

Введение

Шифрование информации — занятие, которым человек занимался ещё до начала первого тысячелетия, занятие, позволяющее защитить информацию от посторонних лиц.

Существует множество методов шифрования. Одним из них является блочное шифрование. Блочные шифры оперируют группами бит фиксированной длины — блоками, характерный размер которых меняется в пределах 64–256 бит. Одним из самых распространённых шифров является AES. Это симметричный алгоритм блочного шифрования (размер блока 128 бит, ключ 128/192/256 бит), принятый в качестве стандарта шифрования правительством США по результатам конкурса AES. Этот алгоритм хорошо проанализирован и сейчас широко используется, как это было с его предшественником DES.

Цель — реализация программы шифрования симметричным алгоритмом AES с применением режима шифрования CFB.

Для достижения поставленной цели необходимо выполнить следующие задачи:

— изучить алгоритм работы алгоритма AES;

— изучить режим шифрования CFB;

— реализовать в виде программы алогритм шифрования AES с применением режима шифрования CFB;

— обеспечить шифрование и расшифровку произвольного файла с использованием разработанной программы;

— предусмотреть работу программы с пустым, однобайтовым файлом и с файлами архива (rar, zip или др.).

1 Аналитическая часть

1.1 Алгоритм AES

AES — симметричный итеративный блочный алгоритм шифрования, базирующийся на SP–сети. В отличии от сетей Фейстеля, в SP–сети шифруется весь текст. Особенностями алгоритма является: представление шифруемого блока в виде двумерного байтового массива, шифрование за один раунд всего блока данных (байт-ориентированная структура), выполнение криптографических преобразований, как над отдельными байтами массива, так и над его строками и столбцами. Последний пункт обеспечивает рассеивание.

AES позволяет использовать три различных ключа длиной 128,192 или 256 бит. В зависимости от длинны ключа варьируется количество раундов. Для 128 битного ключа используется 10 раундов. Перед началом раундов производится расширение ключа от одного 128 битного до одиннадцати 128 битных ключей. Дополнительный ключ используется для начальной подстановки, которая нужна для единообразия алгоритмов кодирования и декодирования. Общая схема алгоритма представлена на рисунке 1.1.

Каждый раунд шифрования состоит из 4 этапов: SubBytes, ShiftRows, MixColumns, AddRoundKey. Последний раунд не включает операцию MixColumns, для избежания бесполезных вычислений, поскольку MixColumns линейная, ее воздействие в самом последнем раунде можно аннулировать, скомбинировав биты способом, не зависящим ни от их значения, ни от ключа.

Для описания алгоритма используется конечное поле Галуа $GF(2^8)$, построенное как расширение поля$GF(2)$ по модулю неприводимого многочлена $m(x) = x^8 + x^4 + x^3 + x + 1$, аналог простого числа. Элементами поля являются многочлены 7 степени. Коэффициентами являются биты. Сложение подставляет собой операцию XOR. Умножение байт выполняется с помощью представления их многочленами и перемножения по обычным алгебраическим правилам, полученное произведение необходимо привести по модулю многочлена $m(x)$. Для любого ненулевого битового многочлен в поле существует многочлен обратный к нему по умножению. Для нахождения обратного элемента используют расширенный алгоритм Эвклида.

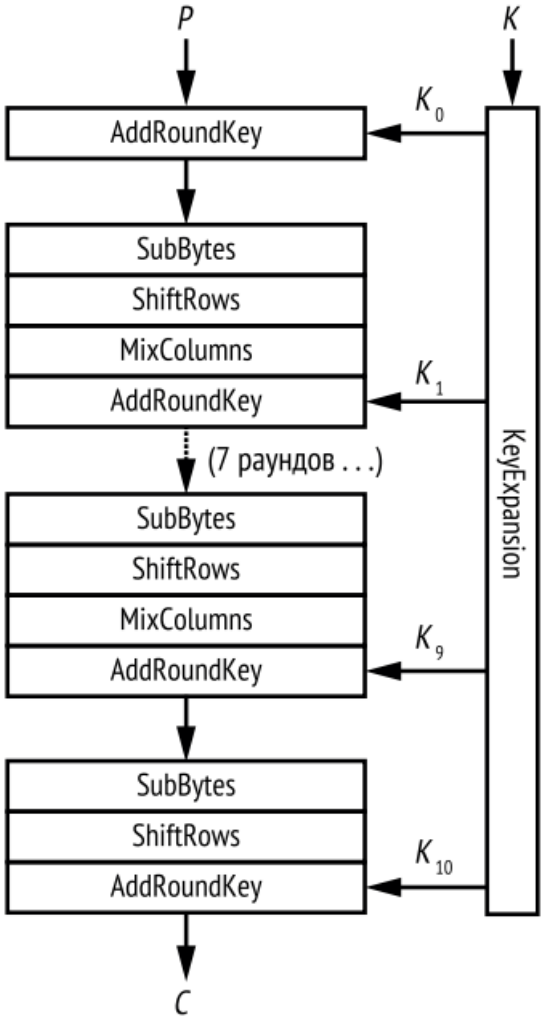

Рисунок 1.1 – Схема AES

SubBytes заменяет каждый байт блока другим байтом согласно S–блоку, который представляет собой таблицу из 256 элементов. SubBytes привносит нелинейные операции, повышающие криптографическую стойкость. Операция выполняет нелинейную замену байтов, выполняемую независимо с каждым байтом матрицы. Замена обратима и построена путем комбинации двух преобразований над входным байтом: нахождение обратного, умножение его на многочлен $x^4+x^3+x^2+x+1$ и суммирование с многочленом x^6+x^5+x+1. Эту операцию можно представить в матричном виде. Нелинейность преобразования обусловлена нелинейностью инверсии, а обратимость – обратимостью матрицы. В программной реализации используется массив с уже подсчитанными подстановками для каждого элемента поля. Схема на рисунке 1.2.

.

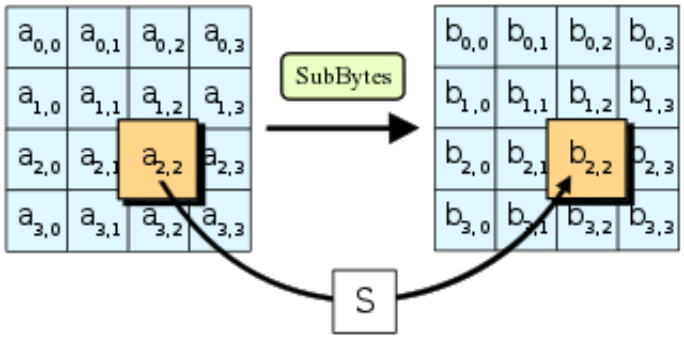

Рисунок 1.2 – Схема SubBytes

ShiftRows циклически сдвигает i-ю строку на i позиций, где i изменяется от 0 до 3. Без ShiftRows изменения в любом столбце не оказывали бы влияния на другие столбцы. Схема представлена на рисунке 1.3.

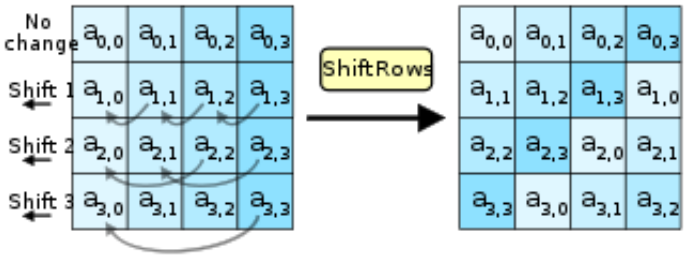

Рисунок 1.3 – Схема ShiftRows

.

В MixColumns используются многочлены третьей степени с коэффициентами из конечного поля $GF(2^8)$ и имеют вид $a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$. Таким образом, в этих многочленах в роли коэффициентов при неизвестных задействованы байты вместо бит. Введём дополнительно многочлен $b(x) = b_3x^3 + b_2x^2 + b_1x + b_0$, тогда сложение определено как $a(x) + b(x) = (a_3 \oplus b_3)x^3 + (a_2 \oplus b_2)x^2 + (a_1 \oplus b_1)x + (a_0 \oplus b_0)$. Умножение аналогично умножению многочленов, взятое по модулю $x^4 + 1$.

MixColumns применяет одно и то же линейное преобразование к каждому из четырех столбцов состояния. Без MixColumns изменение одного байта не влияло бы на остальные байты состояния. Каждый столбец этой матрицы принимается за многочлен над полем $GF(2^8)$ и умножается на фиксированный многочлен $c(x) = c_3x^3 + c_2x^2 + c_1x^1 + c_0 = 3x^3 + 1x^2 + 1x + 2$. Такую операцию можно записать в матричном виде как

$$\begin{pmatrix} c_0 & c_3 & c_2 & c_1 \\ c_1 & c_0 & c_3 & c_2 \\ c_2 & c_1 & c_0 & c_3 \\ c_3 & c_2 & c_1 & c_0 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}.$$

Схема представлена на рисунке 1.4.

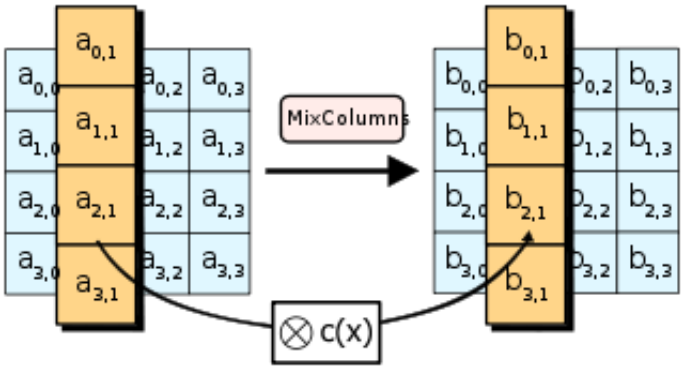

Рисунок 1.4 – Схема MixColumns

.

AddRoundKey применяет XOR к ключу раунда и внутреннему состоянию. Без KeyExpansion во всех раундах использовался бы один и тот ключ K.

Схема представлена на рисунке 1.5.

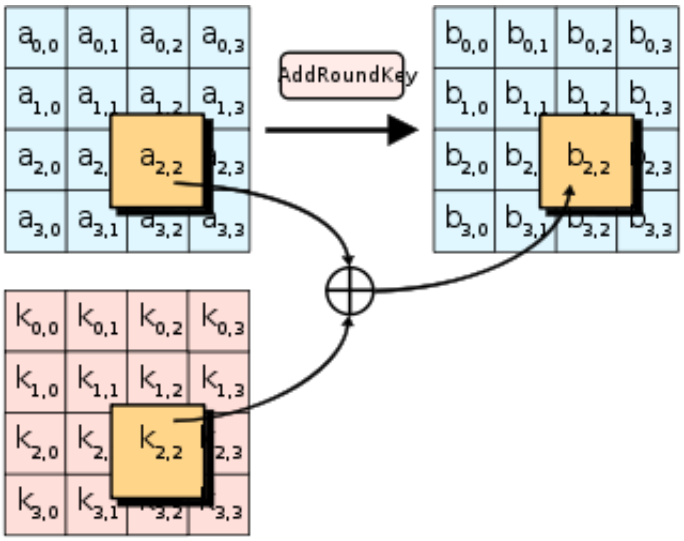

Рисунок 1.5 – Схема AddRoundKey

.

Для расшифрования шифротекста все используемые шифрующие преобразования могут быть инвертированы и применены в обратном порядке. Перед первым раундом дешифрования выполняется операция AddRoundKey , накладывающая на шифротекст четыре последних слова расширенного ключа. Порядок: InvShiftRows, InvSubBytes, AddRoundKey, InvMixColumns. В последнем раунде не выполняется InvMixColumns.

Раундовые ключи вырабатываются из ключа шифра K с помощью процедуры расширения ключа, в результате чего формируется массив раундовых ключей, из которого затем непосредственно выбирается необходимый раундовый ключ. Каждый раундовый ключ имеет длину 128 бит (или 4 32 битных слова слова w_i, w_{i+1}, w_{i+2}, w_{i+3}). Первые четыре слова являются ключом с номером 0. Новые слова w_{i+4}, w_{i+5}, w_{i+6}, w_{i+7} следующего раундового ключа определяются из словw_i, w_{i+1}, w_{i+2}, w_{i+3} предыдущего ключа на основе уравнения $w_{i+j} = w_{i+j-1} \oplus w_{i+j-4}, j = 5, 7$. Первое слово w_{i+4} в каждом ключе получается как $w_{i+4} = w_i \oplus g(w_{i+3})$. Действие функции g сводится к последовательному выполнению трёх шагов, отображающих слово в слово:

— циклический сдвиг четырехбайтового слова влево на один байт;

— замена каждого байта слова соответствии с таблицей из SubBytes;

— суммирование по модулю 2 байтов, раундовой постоянной заданной таблицей.

Цель суммирования с раундовыми константами — разрушить любую симметрию, что может возникнуть на разных этапах разворачивания ключа. Схема представлена на рисунке 1.6.

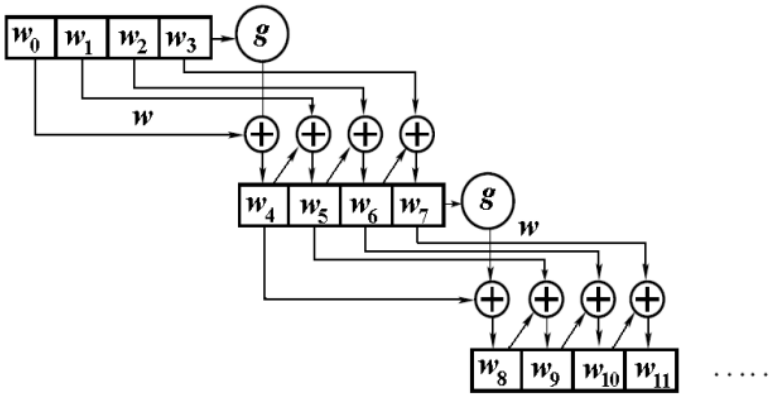

Рисунок 1.6 – Схема алгоритма расширения ключа

.

1.2 Режим шифрования CFB

Режим обратной связи по шифротексту или режим гаммирования с обратной связью (CFB) — один из вариантов использования симметричного блочного шифра, при котором для шифрования следующего блока открытого текста он складывается по модулю 2 с перешифрованным результатом шифрования предыдущего блока. Ошибка, которая возникает в шифротексте при передаче, сделает невозможным расшифровку как блока, в котором ошибка произошла, так и следующего за ним, однако не распространяется на последующие блоки. Криптостойкость определяется криптостойкостью используемого шифра. Возможности распараллеливания процедуры шифрования ограничены, можно распараллелить расшифровку. Схемы шифрования и расшифровки представлены на рисунках 1.7 и 1.8.

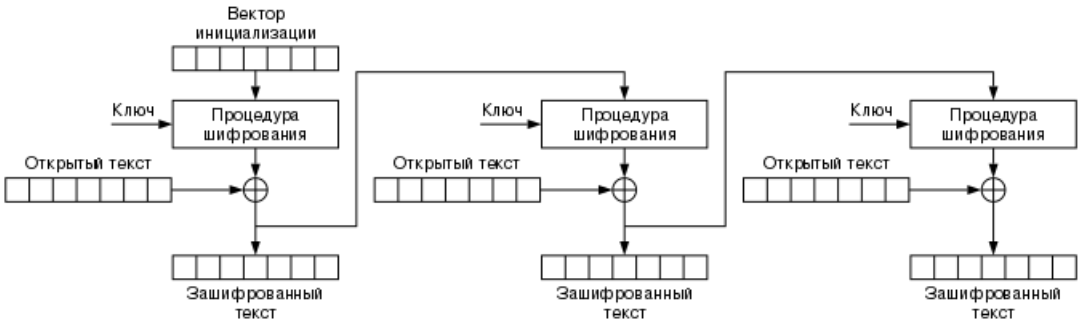

Рисунок 1.7 – Шифрование с CFB

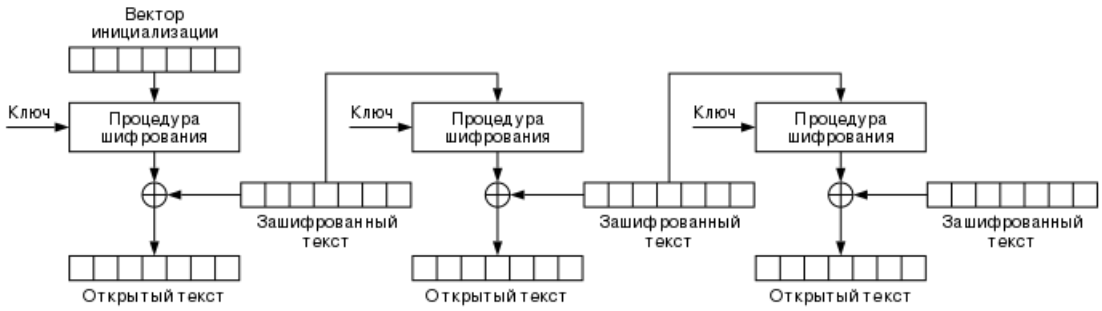

Рисунок 1.8 – Дешифрование с CFB

Вывод

В данном разделе был рассмотрен алгоритм шифрования AES с использованием режима шифрования CFB.

2 Конструкторская часть

В этом разделе будут представлены сведенья о модулях программы.

2.1 Сведения о модулях программы

Программа состоит из двух модулей:

1) *main.c* — файл, содержащий точку входа;

2) *AES.h* — файл, содержащий описание функций шифрования.

3) *AES.c* — файл, содержащий определение функций шифрования.

3 Технологическая часть

В данном разделе будут рассмотрены средства реализации, а также представлены листинги реализаций алгоритма шифрования AES.

3.1 Средства реализации

В данной работе для реализации был выбран язык программирования C. Данный язык удоволетворяет поставленным критериям по средствам реализации.

3.2 Реализация алгоритма

В листингах 3.1 и 3.2 представлена реализация алгоритма шифрования AES с использованием режима CFB.

Листинг 3.1 – Шифрование AES

```
1  void encode_block(uint8_t *dst, uint8_t *src, uint32_t *keys) {
2      uint8_t block[16];
3
4      copy_t_block(block, src);
5
6      add_round_key(block, keys);
7      for (uint8_t round = 1; round < ROUND_NUM; ++round) {
8          sub_bytes(block);
9          shift_rows(block);
10         mix_columns(block);
11         add_round_key(block, keys + (NB * round));
12     }
13     sub_bytes(block);
14     shift_rows(block);
15     add_round_key(block, keys + (NB * ROUND_NUM));
16
17     copy_t_block(dst, block);
18 }
```

Листинг 3.2 – Дешифрование AES

```
void decode_block(uint8_t *dst, uint8_t *src, uint32_t *keys) {
    uint8_t block[16];

    copy_t_block(block, src);

    add_round_key(block, keys + (NB * ROUND_NUM));
    for (int8_t round = ROUND_NUM - 1; round > 0; --round) {
        inv_shift_rows(block);
        inv_sub_bytes(block);
        add_round_key(block, keys + (NB * round));
        inv_mix_columns(block);
    }
    inv_shift_rows(block);
    inv_sub_bytes(block);
    add_round_key(block, keys);

    copy_t_block(dst, block);
}
```

3.3 Тестирование

Для тестирование написанной программы файл шифровался и дешифровался и сравнивалось их содержимое.

Таблица 3.1 – Функциональные тесты

Входной файл	Ожидаемый результат	Результат
1_in.txt (9 bytes) «encode_block me»	1_out.txt (9 bytes) «encode_block me»	1_out.txt (9 bytes) «encode_block me»
2_in.txt (1 bytes) «a»	2_out.txt (1 bytes) «a»	2_out.txt (1 bytes) «a»
3_in.txt (0 bytes) «»	3_out.txt (0 bytes) «»	3_out.txt (0 bytes) «»
4_in.tar.gz (21 Kb) test.png	4_out.tar.gz (21 Kb) test.png	4_out.tar.gz (21 Kb) test.png

Вывод

Были представлен листинг реализации алгоритма работы энигмы. Также в данном разделе была приведена информация о выбранных средствах для разработки алгоритмов.

Заключение

В результате лабораторной работы были изучены принципы работы алгоритма AES и режима CFB, была написана его программная реализация.

Были решены следующие задачи:

— изучен алгоритм работы алгоритма AES;

— изучен режим шифрования CFB;

— реализован в виде программы алогритм шифрования AES с применением режима шифрования CFB;

— обеспечено шифрование и расшифровка произвольного файла с использованием разработанной программы;

— предусмотрена работа программы с пустым, однобайтовым файлом и с файлами архива (rar, zip или др.).