Цель работы: реализация алгоритма Twofish для шифрования изображения в формате bmp. Программа должна работать в двух режимах: шифрования и дешифрования и позволять вводить ключ вручную и генерировать его автоматически.

1. Описание алгоритма

Авторы: Брюс Шнаер (Bruce Schneier), Джон Келси (John Kelsey), Дуг Уайтинг (Doug Whiting), Дэвид Вагнер (David Wagner), Крис Холл (Chris Hall).

Twofish — итеративный блоковый алгоритм с длиной информационного блока 128 бит и длиной ключа 128, 192 или 256 бит.

В алгоритме несколько шагов, рассмотрим подробнее каждый из них.

1.1. Отбеливание (Whitening)

Перед и после 16 раундов выполняется процедура отбеливания с целью усложнения поиска ключа. Отбеливание происходит следующим образом: исходное сообщение разбивается на четыре 32-битных слова (P_0, P_1, P_2, P_3), которые на начальном этапе складываются по модулю 2 (XOR) с четырьмя 32-битыми слова ключа K_0, K_1, K_2, K_3. В результате получаем четыре 32-битных слова R_0, R_1, R_2 и R_3.

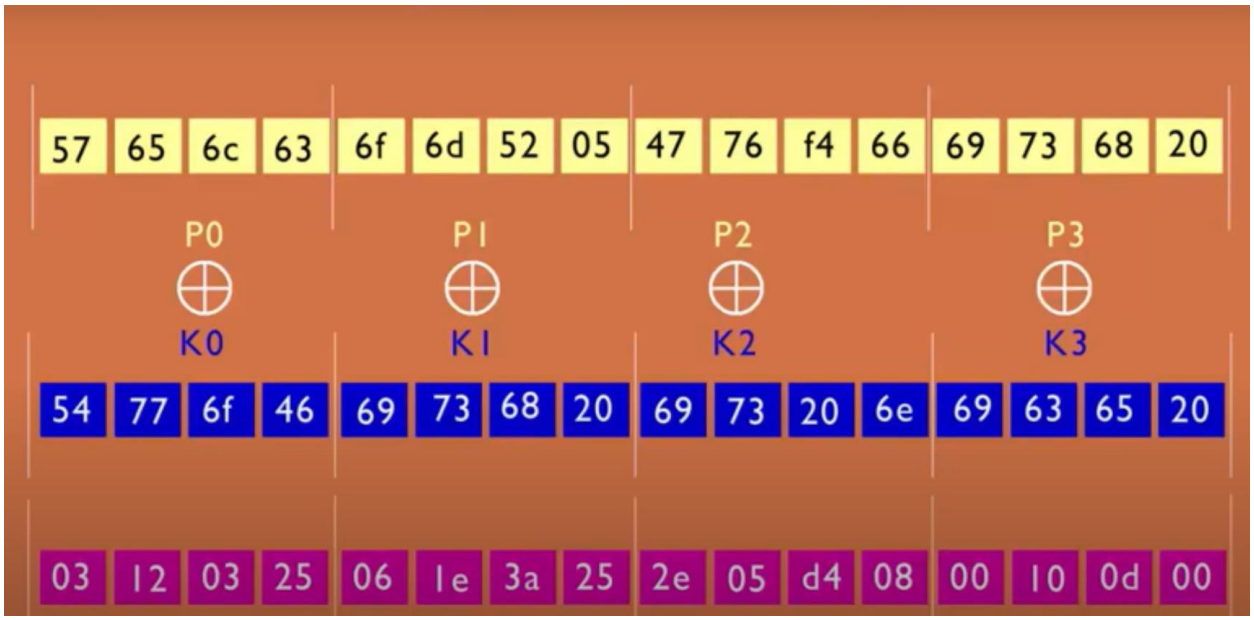

Рисунок 1 - Отбеливание

Затем полученные слова проходят 16 раундов (циклов преобразования).

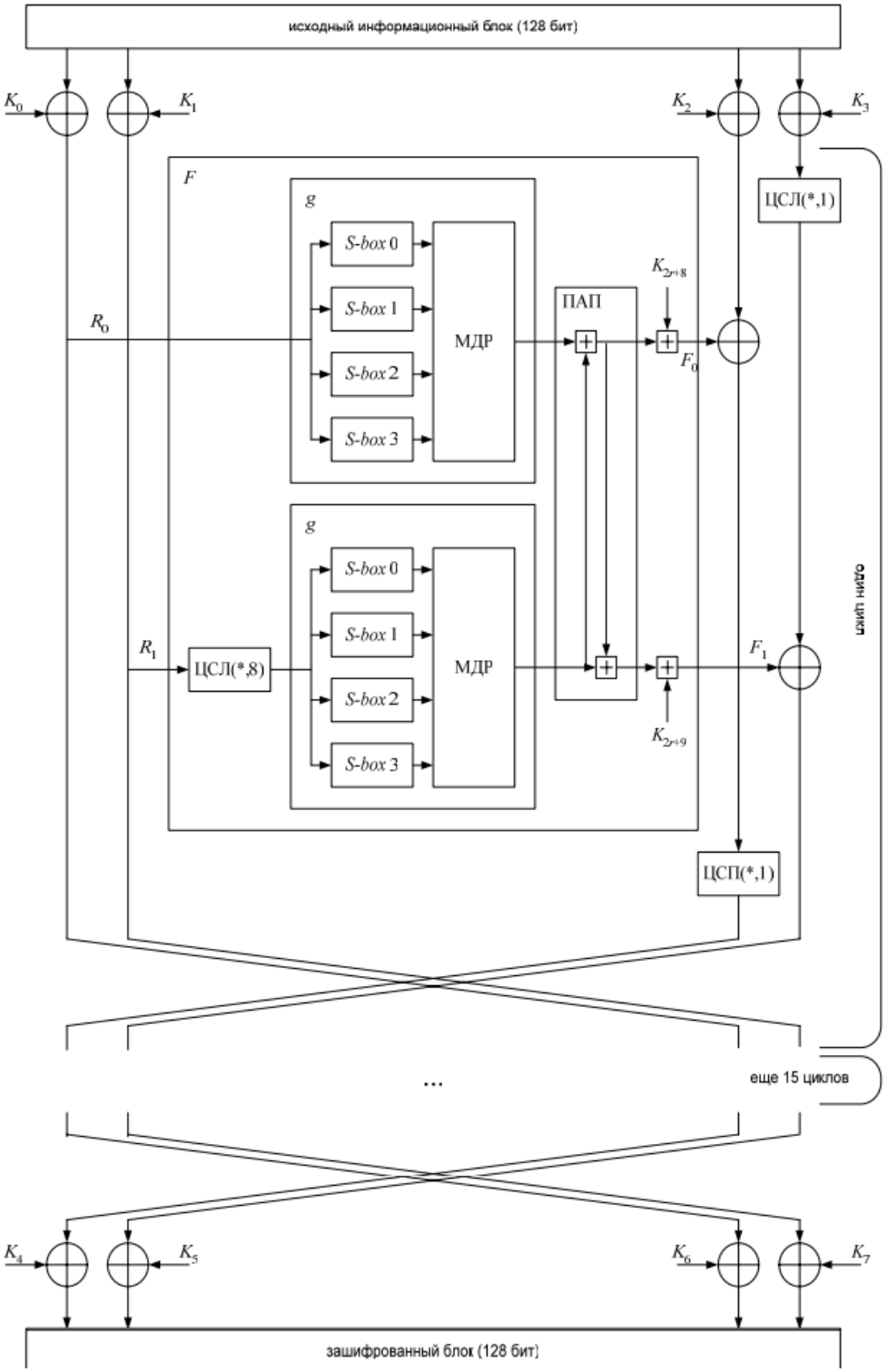

Рисунок 2 - Схема алгоритма

1.2. Функция F

Функция F — 64-битная перестановка, зависящая от ключа. Ее аргументами являются два 32-битных слова R_0 и R_1, а также индекс цикла r, причем предварительно R_1 циклически сдвигается на 8 бит влево (ЦСЛ). Затем оба слова подвергаются преобразованию g. Результатом будет T_0 и T_1. То есть

$$T_0 = g(R_0),$$

$$T_1 = g(\text{ЦСЛ}(R_1, 8)),$$

Далее блоки T_0 и T_1 трансформируются с помощью псевдоадамарового преобразования (ПАП, англ. PHT) и складываются по модулю с соответствующими блоками ключа. То есть:

$$F_0 = (T_0 + T_1 + K_{2r+8}) \; mod \; 2^{32}$$

$$F_1 = (T_0 + 2T_1 + K_{2r+9}) \; mod \; 2^{32}$$

где F_0 и F_1 – результат функции F.

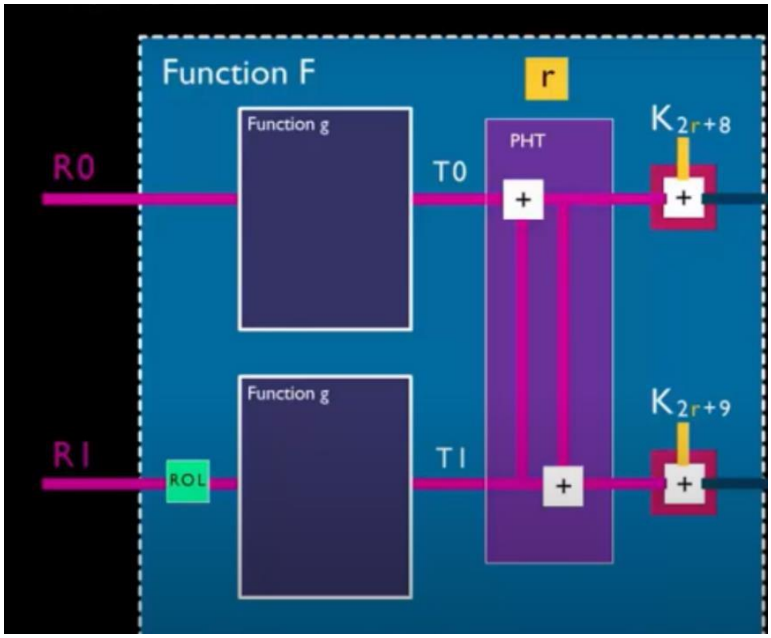

Рисунок 3 - Функция F

1.3. Преобразование g.

По признанию авторов, это преобразование является "сердцем" алгоритма Twofish. Исходный блок X (равный R_0 или ЦСЛ(R_1, 8)) разбивается на 4 байта. Каждый байт подвергается преобразованию в соответствующей $S - box$. На выходе получаем четыре 8-битных блока y_0, y_1, y_2 и y_3, которые представляются как вектор длины 4 над $GF(2^8)$, которые умножается на матрицу MDS размера 4*4. Все операции выполняются над полем $GF(2^8)$. На выходе получаем 32-битное слово Z.

$$x_i = [X/2^{8i}] \; mod \; 2^8, i = 0, \dots, 3$$

$$y_i = S - box[x_i], i = 0, \dots, 3$$

$$\begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} \cdot & \cdots & \cdot \\ \vdots & \text{МДР} & \vdots \\ \cdot & \cdots & \cdot \end{pmatrix} \cdot \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix},$$

$$Z = \sum_{i=0}^{3} z_i * 2^{8i}$$

Используемая в алгоритме MDS -матрица, где элементы представлены в 16 системе счисления имеет следующий вид:

$$\begin{pmatrix} \cdot & \cdots & \cdot \\ \vdots & \text{МДР} & \vdots \\ \cdot & \cdots & \cdot \end{pmatrix} = \begin{pmatrix} 01 & EF & 5B & 5B \\ 5B & EF & EF & 01 \\ EF & 5B & 01 & EF \\ EF & 01 & EF & 5B \end{pmatrix},$$

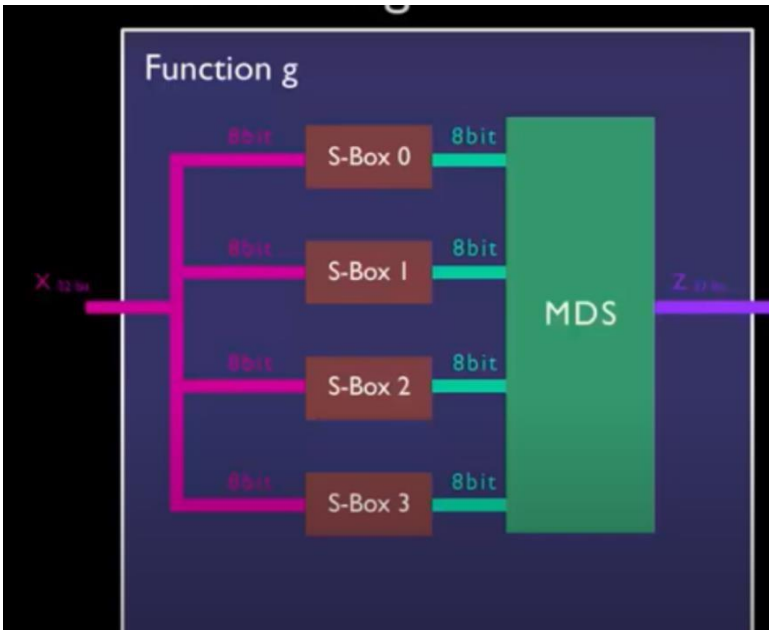

Рисунок 4 - Функция g

Преобразование g определяется через функцию h следующим образом:

$$g(X) = h(X, S)$$

1.4. Функция h

У функции два аргумента – 32-битное слово X и список $L = (L_0, L_1, .. L_{k-1})$ из k 32-битных слов, результатом выполнения является 32-битное слово Z. Схема приведена ниже:

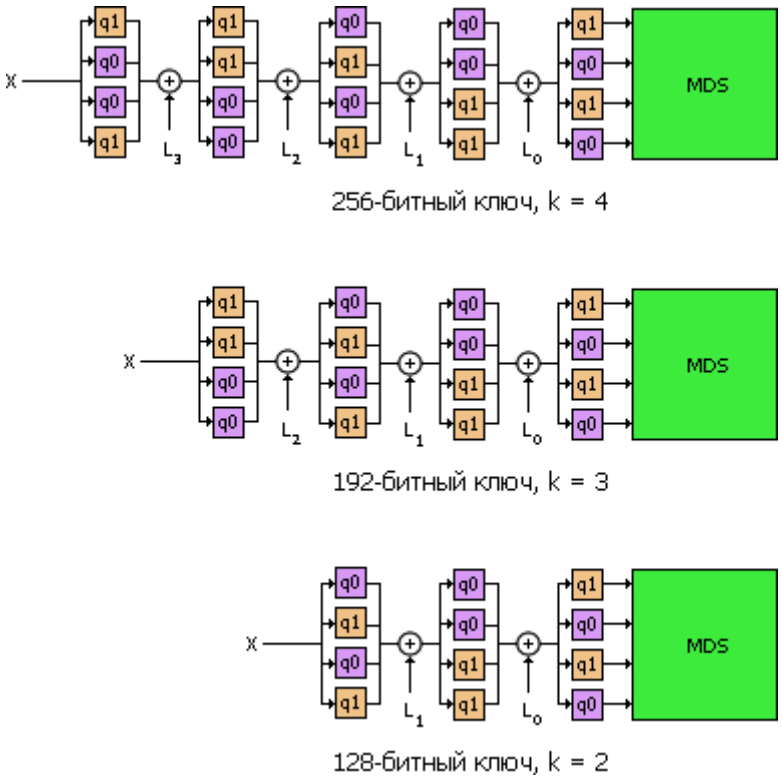

Рисунок 5 - Функция h

где q_0 и q_1 — фиксированные преобразования для 8-битных векторов, которые будут рассмотрены немного позже.

Таким образом, преобразование g определяется через функцию h следующим образом:

$$g(X) = h(X, S)$$

то есть $S - box$, используемые в функции g, обеспечивают преобразование из x_i в y_i с помощью функции h, для которой $L = S$, полученному из исходного ключа в соответствии со схемой генерации ключа, описанной ранее.

1.5. Преобразования q_0 и q_1

Из исходного 8-битного вектора x получается вектор y получается следующим образом:

$$a_0, b_0 = \lfloor x/16 \rfloor, \qquad x \bmod 16$$

$$a_1 = a_0 \oplus b_0$$

$$b_1 = a_0 \oplus \text{ЦСП}(b_0, 1) \oplus 8a_0 \bmod 16$$

$$a_2, b_2 = t_0[a_1], t_1[b_1]$$

$$a_3 = a_2 \oplus b_2$$

$$b_3 = a_2 \oplus \text{ЦСП}(b_2, 1) \oplus 8a_2 \bmod 16$$

$$a_4, b_4 = t_2[a_3], t_3[b_3]$$

$$y = 16b_4 + a_4$$

Для преобразования q_0 4-битные подстановки (S-box) задаются следующими векторами:

$$t_0 = [817D6F320B59ECA4],$$
$$t_1 = [ECB81235F4A6709D],$$
$$t_2 = [BA5E6D90C8F32471],$$
$$t_3 = [D7F4126E9B3085CA],$$

Для преобразования q_1 4-битные подстановки (S-box) задаются следующими векторами:

$$t_0 = [28BDF76E31940AC5],$$
$$t_1 = [1E2B4C376DA5F908],$$
$$t_2 = [4C75169A0ED82B3F],$$
$$t_3 = [B951C3DE647F208A].$$

1.6. Генерация ключей L

Процедура генерации ключей обеспечивает создание 40 слов расширенного ключа и четырех зависимых от ключа S-box, используемых в функции g. Как отмечалось ранее, алгоритм Twofish рассчитан на ключ длиной N = 128, 192 или 256 бит. Если длина ключа не совпадает с этими величинами, то недостающие до ближайшего установленного N биты ключа заполняются 0.

Определим величину $k = N/64$. Ключ M состоит из $8k$ байт $m_0, \dots, m_{8k-1}$. Эти байты первоначально переписываются в $2k$ слов по 32 бита в каждом,

$$M_i = \sum_{j=0}^{3} m_{(4i+j)} * 2^{8j}, i = 0, \dots, 2k-1$$

А затем в два вектора длины k:

$$M_{\text{ч}} = (M_0, M_2, \dots, M_{2k-2})$$

$$M_{\text{н}} = (M_1, M_3, \dots, M_{2k-1})$$

Третий вектор длины k также получается из исходного ключа. Для этого байты ключа группируются по 8 и интерпретируются как вектора, составленные из элементов $GF(2^8)$. Затем эти вектора умножаются справа на матрицу 4×8, полученную из порождающей матрицы помехоустойчивого кода Рида-Соломона над $GF(2^8)$ с длиной кодового слова 12 и порождающим многочленом четвертой степени, так что $(s_{i,0}\ s_{i,1}\ s_{i,2}s_{i,3}m_{8i}m_{8i+1}m_{8i+2}m_{8i+3}m_{8i+4}m_{8i+5}m_{8i+6}m_{8i+7})$ – кодовое слово этого кода. То есть $(m_{8i}m_{8i+1}m_{8i+2}m_{8i+3}m_{8i+4}m_{8i+5}m_{8i+6}m_{8i+7})$ – информационные, а $(s_{i,0}s_{i,1}\ s_{i,2}s_{i,3})$ – проверочные символы:

$$\begin{pmatrix} s_{i,0} \\ s_{i,1} \\ s_{i,2} \\ s_{i,3} \end{pmatrix} = \begin{pmatrix} \cdot & \cdots & \cdot \\ \cdot & \text{PC} & \cdot \\ \cdot & \cdots & \cdot \end{pmatrix} \cdot \begin{pmatrix} m_{8i} \\ m_{8i+1} \\ m_{8i+2} \\ m_{8i+3} \\ m_{8i+4} \\ m_{8i+5} \\ m_{8i+6} \\ m_{8i+7} \end{pmatrix},$$

$$S_i = \sum_{j=0}^{3} s_{i,j} * 2^{8j}$$

для $i = 0, \dots, k-1$.

Таким образом, из каждых восьми байт исходного ключа получается 32-битное слово. Эти слова и формируют третий вектор S:

$$S = (S_{k-1}S_{k-2}, \dots, S_0)$$

Для перемножения вектора на проверочную матрицу кода РС используется таблица умножения для элементов $GF(2^8)$ с примитивным многочленом $w(x) = x^8 + x^6 + x^3 + x^2 + 1$

$$\begin{pmatrix} \cdot & \cdots & \cdot \\ \cdot & \text{PC} & \cdot \\ \cdot & \cdots & \cdot \end{pmatrix} = \begin{pmatrix} 01 & A4 & 55 & 87 & 5A & 58 & DB & 9E \\ A4 & 56 & 82 & F3 & 1E & C6 & 68 & E5 \\ 02 & A1 & FC & C1 & 47 & AE & 3D & 19 \\ A4 & 55 & 87 & 5A & 58 & DB & 9E & 03 \end{pmatrix}.$$

Векторы $M_{\text{ч}}, M_{\text{н}}, S$ образуют базис для алгоритма генерации ключей с использованием функции h.

1.7. Генерация ключей K_j

Ключи K_j генерируются, используя функцию h:

$$K_{2i} = (A_i + B_i)\ mod\ 2^{32}$$

$$K_{2i+1} = \text{ЦСЛ}((A_i + 2B_i)\ mod\ 2^{32}, 9)$$

где

$$A_i = h(2ip, M_{\text{ч}}),$$

$$B_i = \text{ЦСЛ}(h((2i+1)p, M_{\text{н}}), 8),$$

$$p = 2^{24} + 2^{16} + 2^8 + 2^0$$

Константа p используется для создания вектора из четырёх повторяющихся байт для всех $i = 0, \dots, 255$.

2. Режим CFB

Режим обратной связи по шифротексту. Во время шифрования каждый блок открытого текста складывается по модулю 2 с блоком, зашифрованном на предыдущем шаге.

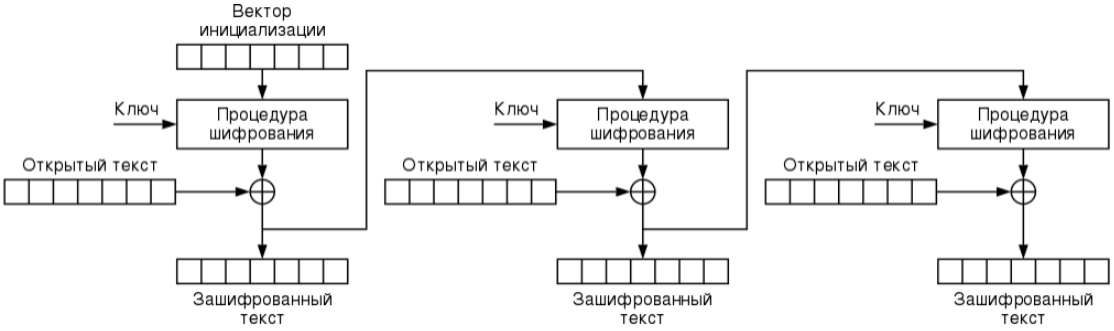

Рисунок 6 - Шифрование в режиме CFB

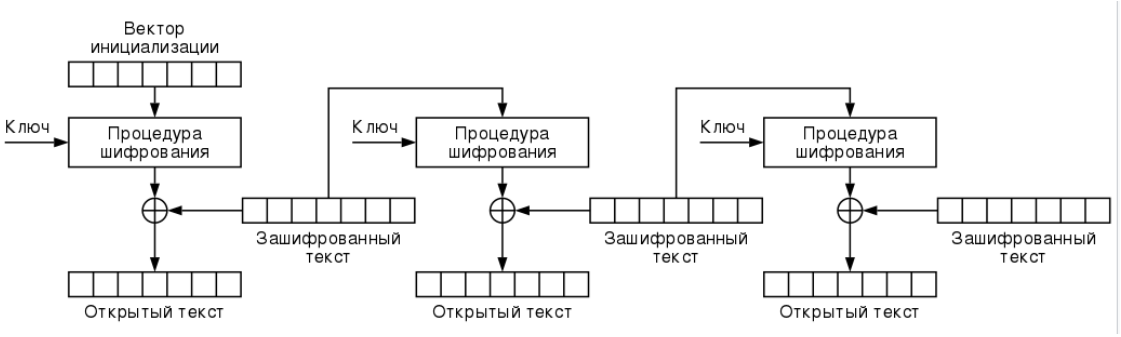

Рисунок 7 - Расшифрование в режиме CFB

3. Описание реализации

Программа реализована на языке программирования Java.

Шифруется изображение в формате bmp. Ключ вводится пользователем, но предусмотрена и автоматическая генерация. Отдельно считывается заголовок и вся служебная информация о файле, которая преобразованию не подлежит. Затем считываются цвета (RGB), записываются в байтовый массив и передаются вместе с ключом в функцию шифрования.

Функция шифрования возвращает зашифрованный массив байт, который затем преобразуется в массив цветов (RGB) и записывается в новое изображение, с сохранением старого заголовка и служебной информации об изображении.

Аналогичным образом происходит дешифрование: цвета зашифрованного изображения преобразуются в массив байт и передаются в функцию дешифрования. Результат аналогичным образом записывается в новый bmp файл.

4. Пример работы программы

Исходное изображение “Image.bmp:

Рисунок 8 - Исходное изображение

Зашифрованное изображение “Encrypted.bmp”:

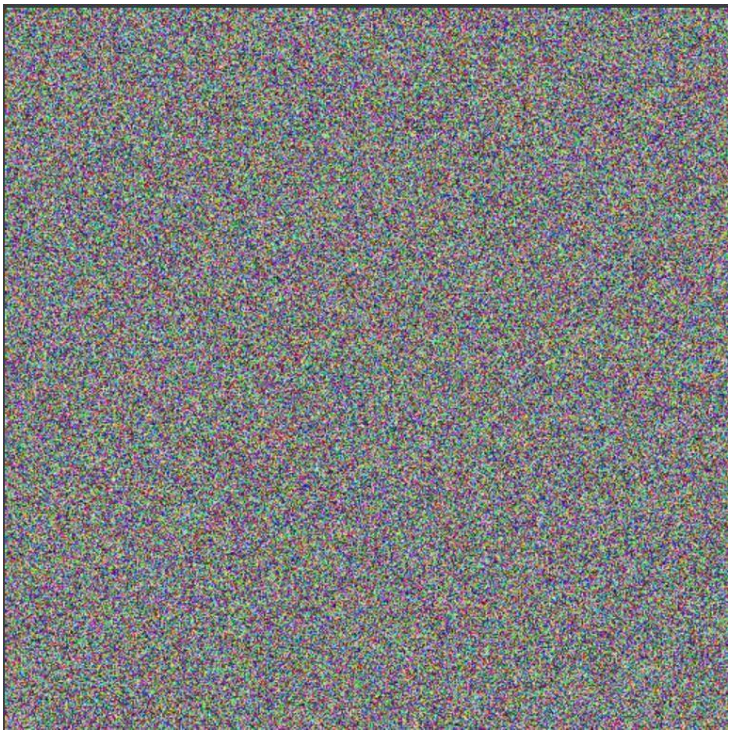

Рисунок 9 - Зашифрованное изображение

Дешифрованное изображение “Decrypted.bmp”:

Рисунок 10 - Дешифрованное изображение

Дешифрованное изображение полностью совпадает с исходным, что говорит о корректной работе программы.

5. Известные атаки

Изучение Twofish с сокращенным числом раундов показало, что алгоритм обладает большим запасом прочности, и, по сравнению с остальными финалистами конкурса AES, он оказался самым стойким. Однако его необычное строение и относительная сложность породили некоторые сомнения в качестве этой прочности.

Нарекания вызвало разделение исходного ключа на две половины при формировании раундовых подключей. Криптографы предположили, что такое разделение дает возможность организовать атаку по принципу «разделяй и властвуй», то есть разбить задачу на две аналогичные, но более простые. Однако реально подобную атаку провести не удалось.

На 2008 год лучшим вариантом криптоанализа Twofish является вариант усечённого дифференциального криптоанализа, который был опубликован в Японии в 2000 году. Он показал, что для нахождения необходимых дифференциалов требуется 2^{51} подобранных открытых текстов. Тем не менее исследования носили теоретический характер, никакой реальной атаки проведено не было.

Вывод: в ходе выполнения лабораторной работы был реализован алгоритм шифрования Twofish в режиме шифрования и дешифрования**.**

Алгоритм Twofish обладает рядом достоинств, однако именно сложность структуры алгоритма и, соответственно, сложность его анализа на предмет слабых ключей или скрытых связей, а также достаточно медленное время выполнения на большинстве платформ, сыграло не в его пользу.

На данный момент рассмотренный алгоритм считается криптостойким, однако ввиду недостатков используется весьма редко.

Используемые источники:

1. Овчинников, А. А. Криптографические методы защиты информации: учеб. пособие / А. А. Овчинников. – СПб.: ГУАП, 2021. – 133 с.

2. Беззатеев С.В., Крук Е.А., Овчинников А.А. Блоковые шифры: Учеб.пособие/СПб.:Изд-во Нестор, 2003, 64 с.