МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
(МОСКОВСКИЙ ПОЛИТЕХ)

Факультет информационных технологий
Кафедра «Инфокогнитивные технологии»

ЛАБОРАТОРНАЯ РАБОТА №3

на тему: *«Применение блочных шифров»*

Направление подготовки 09.03.03 «Прикладная информатика»
Профиль «Корпоративные информационные системы»
Дисциплина «Защита информации»

Выполнил:

студентка группы 201-361

Саблина Анна Викторовна

Проверил:

Харченко Елена Алексеевна

Москва 2023

Теоретическая часть

AES (Advanced Encryption Standard) – это симметричный алгоритм блочного шифрования, разработанный для защиты информации путем шифрования и расшифрования данных.

AES оперирует на блоках данных размером 128 бит и использует ключи различной длины: 128 бит, 192 бит и 256 бит. Он состоит из нескольких раундов шифрования, в каждом из которых выполняются определенные операции над данными.

Раунды AES:

1. SubBytes: каждый байт входного блока заменяется на другой байт из специальной таблицы замен (S-блок). Это помогает внести нелинейность в шифрование.
2. ShiftRows: каждая строка входного блока сдвигается влево. При этом первая строка остается на месте, вторая сдвигается на одну позицию влево, третья на две позиции влево, а четвертая на три позиции влево. Это делается для перестановки байтов внутри каждого блока.
3. MixColumns: каждый столбец входного блока перемешивается. Каждый байт в столбце умножается на определенные значения и складывается по модулю. Это помогает в перемешивании байтов между столбцами блока.
4. AddRoundKey: каждый байт входного блока комбинируется с байтом из раундового ключа с использованием побитовой операции XOR. Раундовой ключ генерируется из основного ключа с использованием специального расписания ключей.

После выполнения всех раундов, за исключением последнего, выполняется финальный раунд без операции MixColumns.

Режимы шифрования AES позволяют использовать AES для защиты данных большего размера или для выполнения различных задач шифрования:

1. *ECB (Electronic Codebook)* – электронная книга кодов. Каждый блок данных шифруется независимо от других блоков. Однако этот режим не обеспечивает конфиденциальность для одинаковых блоков данных.
2. *CBC (Cipher Block Chaining)* – режим связывания блоков шифрования. Каждый блок данных перед шифрованием комбинируется с предыдущим зашифрованным блоком данных путем побитовой операции XOR. Это обеспечивает конфиденциальность и стойкость к атакам с повторением блоков.
3. *CFB (Cipher Feedback)* – режим обратной связи по шифротексту. В этом режиме шифрования предыдущий зашифрованный блок данных обратно связывается с функцией обратной связи и используется для шифрования следующего блока данных. Он позволяет шифровать данные меньшего размера и обеспечивает конфиденциальность и целостность данных.
4. *OFB (Output Feedback)* – режим обратной связи по выходу. В этом режиме шифрования предыдущий блок шифротекста используется для генерации потока псевдослучайных битов, который затем комбинируется с открытым текстом путем побитовой операции XOR для получения шифротекста. Этот режим обеспечивает конфиденциальность и независимость от блоков данных.

- *Вектор обратной связи (Feedback Vector, IV)* – это случайно выбираемое начальное значение, которое используется в режимах блочного шифрования с обратной связью, таких как CBC, CFB и OFB.

Все эти режимы шифрования и раунды AES вместе образуют надежную систему шифрования, обеспечивая конфиденциальность, целостность и защиту данных от несанкционированного доступа и атак.

В задании работы предлагается использовать файл формата PNG.

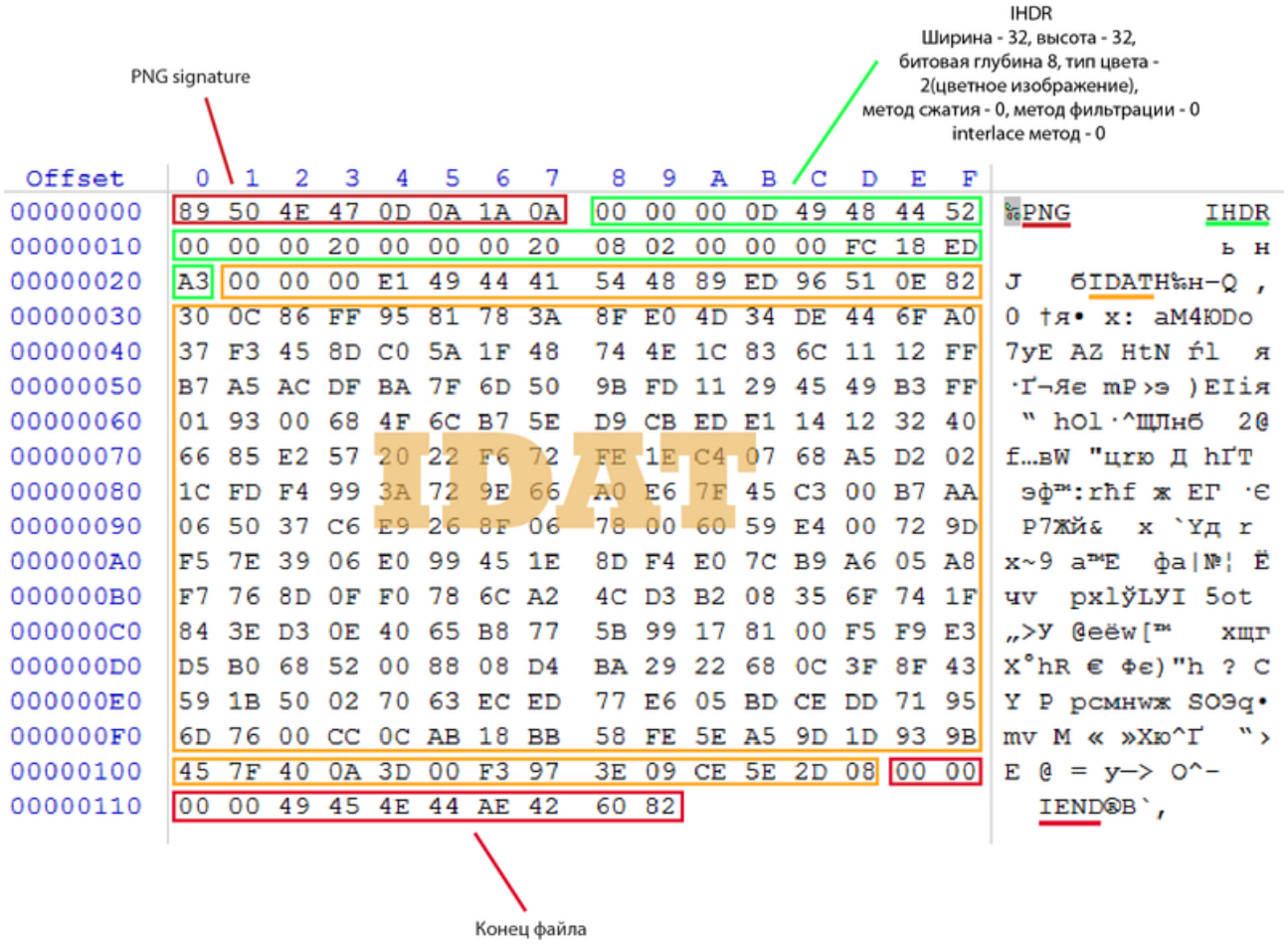

Рисунок 1 – Структура PNG-файла

Чанки – это основные блоки данных, из которых состоит файл PNG. Каждый чанк представляет собой независимую единицу информации, которая содержит определенные данные, такие как изображение, метаданные или палитра цветов.

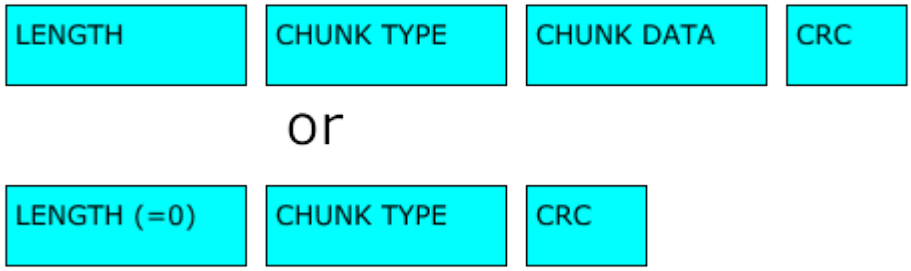

Рисунок 2 – Структура чанков

Представленные выше рисунки демонстрируют структуру файла. Это означает, что для успешного обработки файла необходимо не только отсечь

заголовок, но также обрезать окончание файла, а также учитывать структуру чанков, из которых состоит формат PNG. В каждом чанке, помимо прочего, присутствует контрольная сумма, которая обеспечивает целостность файла. При шифровании файлов контрольная сумма изменяется, что может привести к неправильному отображению изображения или невозможности его открытия. Поэтому в данном случае был выбран аналогичный файл формата BMP, который не имеет чанков и контрольных сумм.

Хедер файла BMP состоит из нескольких параметров, содержащих переменные значения заголовка. В данном конкретном случае, заголовок составляет 110 байт.

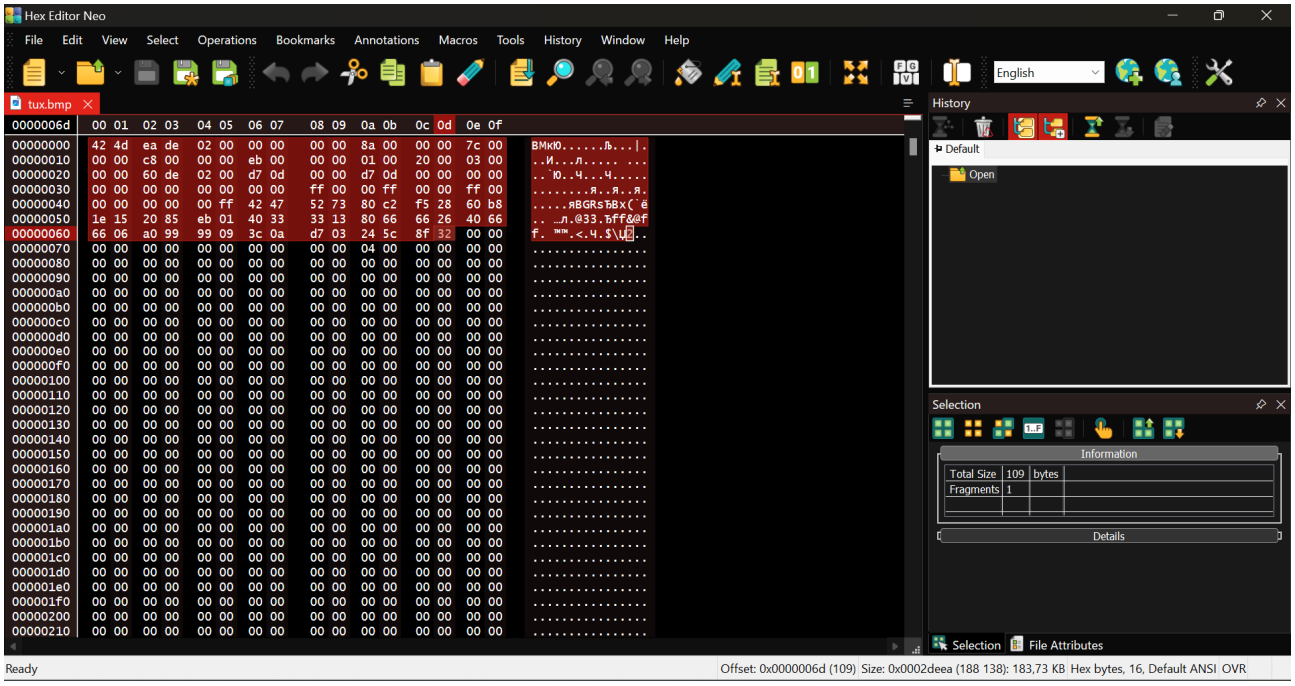

Рисунок 3 –Структура tux.bmp с выделенным заголовком

Таким образом, было принято решение использовать файлы данного формата для обработки, чтобы избежать проблем с контрольными суммами и структурой чанков, присущих формату PNG.

Практическая часть

Для реализации программы, способной зашифровать файл tux.bmp с помощью шифра AES в режимах шифрования ECB, CBC, CFB, OFB, была использована утилита `openssl`.

Был создан проект на языке Java.

Общий принцип работы программы:

1. Задается имя входного файла изображения (`inputFileName`).
2. Читаются все байты изображения в массив `imageData`.
3. Создается массив `header` заданной длины (`headerLength`) и копируются первые байты из `imageData` в `header`.
4. Создается массив `imageDataWithoutHeader`, который содержит данные изображения без заголовка путем копирования оставшихся байтов из `imageData`.
5. Заголовок сохраняется в отдельном файле `tux_header.bin` путем записи байтов из `header` в файл.
6. Данные без заголовка сохраняются в отдельном файле `tux_without_header.bmp` путем записи байтов из `imageDataWithoutHeader` в файл.
7. Вызывается метод `encryptAndSaveData` с передачей заголовка и различных режимов шифрования ("ecb", "cbc", "cfb", "ofb").
    - Метод `encryptAndSaveData` генерирует ключ шифрования, используя внешнюю программу `openssl`.
    - Затем данные изображения без заголовка шифруются с использованием указанного режима шифрования и сохраняются в зашифрованном виде в соответствующем файле.

8. Программа завершается или продолжает выполнение дальше после шифрования и сохранения данных в различных режимах.

Программа выполняет чтение изображения, извлечение заголовка, сохранение заголовка и беззаголовочных данных, а также шифрование и сохранение данных с использованием различных режимов шифрования с помощью внешней программы `openssl`.

```
public static void main(String[] args) {
    // Имя входного файла изображения
    String inputFileName = "tux.bmp";
    byte[] imageData;
    byte[] header;
    int headerLength = 110;
    try {
        // Чтение всех байтов изображения
        imageData = Files.readAllBytes(Path.of(inputFileName));
        // Извлечение заголовка
        header = new byte[headerLength];
        System.arraycopy(imageData, 0, header, 0, headerLength);

        // Извлечение данных без заголовка
        byte[] imageDataWithoutHeader = new byte[imageData.length - headerLength];
        System.arraycopy(imageData, headerLength, imageDataWithoutHeader,
                0, imageData.length - headerLength);

        // Сохранение заголовка в отдельный файл
        FileOutputStream outputStream = new FileOutputStream("tux_header.bin");
        outputStream.write(header);
        outputStream.close();

        // Сохранение данных без заголовка в отдельный файл
        FileOutputStream outputStreamWithoutHeader = new
                 FileOutputStream("tux_without_header.bmp");
        outputStreamWithoutHeader.write(imageDataWithoutHeader);
        outputStreamWithoutHeader.close();

        // Шифрование и сохранение данных с использованием различных режимов шифрования
        encryptAndSaveData(header, "ecb");
        encryptAndSaveData(header, "cbc");
        encryptAndSaveData(header, "cfb");
        encryptAndSaveData(header, "ofb");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Листинг 1 – Метод main

Метод `encryptAndSaveData(byte[] header, String encryptionMode)` шифрует и сохраняет данные изображения с использованием указанного режима шифрования.

```
public static void encryptAndSaveData(byte[] imageData, byte[] header,
                                       String encryptionMode) {
    try {
        // Генерация AES ключа
```

```
        KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");
        keyGenerator.init(128);
        SecretKey secretKey = keyGenerator.generateKey();

        // Инициализация шифра с ключом AES и режимом шифрования
        Cipher cipher = Cipher.getInstance("AES/" + encryptionMode + "/PKCS5Padding");
        cipher.init(Cipher.ENCRYPT_MODE, secretKey);

        // Шифрование данных изображения
        byte[] encryptedData = cipher.doFinal(imageData);

        // Объединение заголовка и зашифрованных данных изображения
        byte[] encryptedImage = new byte[header.length + encryptedData.length];
        System.arraycopy(header, 0, encryptedImage, 0, header.length);
        System.arraycopy(encryptedData, 0, encryptedImage, header.length,
                         encryptedData.length);

        // Сохранение зашифрованного изображения в новый файл
        String outputFileName = "tux" + encryptionMode + "_encrypted.bmp";
        FileOutputStream outputStream = new FileOutputStream(outputFileName);
        outputStream.write(encryptedImage);
        outputStream.close();

        System.out.println("Image encrypted and saved: " + outputFileName);

    } catch (NoSuchAlgorithmException | NoSuchPaddingException | InvalidKeyException |
                 IllegalBlockSizeException | BadPaddingException | IOException e) {
        e.printStackTrace();
    }
}
public static void encryptAndSaveData(byte[] header, String encryptionMode) throws
IOException {
    // Путь к файлу ключа шифрования
    String keyPath = "key.bin";
    // Путь к исходному файлу изображения без заголовка
    String tuxWithoutHeaderPath = "tux_without_header.bmp";
    // Путь к зашифрованному файлу изображения
    String tuxEncryptedPath = "tux_" + encryptionMode + ".bmp";

    // Генерация ключа шифрования
    ProcessBuilder keyGenProcessBuilder = new ProcessBuilder("openssl", "rand",
            "-hex", "16");
    keyGenProcessBuilder.redirectOutput(ProcessBuilder.Redirect.to(new File(keyPath)));
    Process keyGenProcess = keyGenProcessBuilder.start();
    try {
        keyGenProcess.waitFor();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    // Шифрование данных изображения без заголовка
    ProcessBuilder encryptionProcessBuilder = new ProcessBuilder("openssl", "enc",
            "-aes-256-" + encryptionMode,
            "-in", tuxWithoutHeaderPath,
            "-out", tuxEncryptedPath,
            "-pass", "file:key.bin");

    encryptionProcessBuilder.redirectErrorStream(true);
    Process encryptionProcess = encryptionProcessBuilder.start();
    try {
        BufferedReader reader = new BufferedReader(
                new InputStreamReader(encryptionProcess.getInputStream()));
        String line;
        while ((line = reader.readLine()) != null) {
            System.out.println(line);
        }
        encryptionProcess.waitFor();
    } catch (InterruptedException e) {
```

```
            e.printStackTrace();
        }

        // Чтение зашифрованных данных
        byte[] encryptedImageData = Files.readAllBytes(Path.of(tuxEncryptedPath));

        // Создание выходного файла и запись заголовка + зашифрованных данных
        FileOutputStream outputStream = new FileOutputStream(tuxEncryptedPath);
        outputStream.write(header);
        outputStream.write(encryptedImageData);
        outputStream.close();
    }
```

Листинг 2 – Метод encryptAndSaveData(byte[] header, String encryptionMode)

Получившиеся в итоге зашифрованные изображения в формате .bmp:

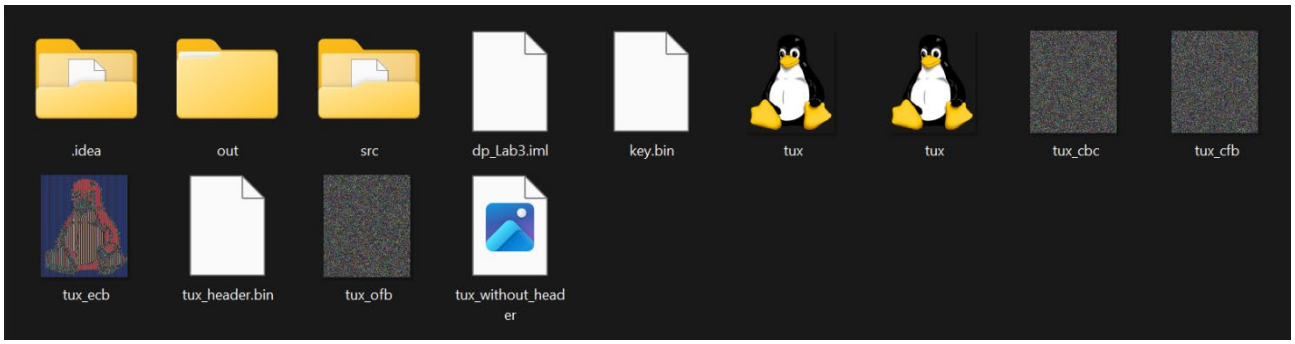

Рисунок 4 – Созданные файлы

Ссылка на проект в репозитории GitHub:

– https://github.com/LazyShAman/dp/tree/main/3.