Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

[Кафедра информационных](https://www.belstu.by/fakultety/fit/vm) систем и технологий

Специальность 1-40 01 01 Программное обеспечение информационных технологий

**Отчёт по лабораторной работе №14**

по дисциплине Информационная безопасность

Тема: Исследование стеганографического метода на основе преобразования наименее значащих битов

Исполнитель:

Студент 3 курса группы 4

Станчик Максим Андреевич

Руководитель:

Ассистент Савельева М. Г.

Минск, 2025

**Лабораторная работа №14**

**Цель:** изучение стеганографического метода встраивания/извлечения тайной информации с использованием электронного файла-контейнера на основе преобразования наименее значащих битов (НЗБ), приобретение практических навыков программной реализации данного метода.

**Задачи:**

1. Закрепить теоретические знания из области стеганографического преобразования информации, моделирования стеганосистем, классификации и сущности методов цифровой стеганографии.

2. Изучить алгоритм встраивания/извлечения тайной информации на основе метода НЗБ (LSB – Least Significant Bit), получить опыт практической реализации метода.

3. Разработать приложение для реализации алгоритма встраивания/извлечения тайной информации с использованием электронного файла-контейнера на основе метода НЗБ.

4. Познакомиться с методиками оценки стеганографической стойкости метода НЗБ.

5. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента

# Теоретические сведения

Стеганографическая система (stegosystem, стегосистема или стеганосистема – в русскоязычной тематической литературе используются оба сокращения) – совокупность средств и методов, которые используются для формирования скрытого канала передачи (или хранения) информации.

Абстрактно стеганографическая система обычно определяется как некоторое множество отображений одного пространства (множества возможных сообщений *М*) в другое пространство (множество возможных стеганосообщений *S*), и наоборот.

Основные компоненты стеганосистемы:

• контейнер *С* (файл-контейнер или электронный документ произвольного формата), в котором размещается (осаждается, скрывается) тайное сообщение *М*; именно контейнер является упомянутым скрытым каналом;

• тайное сообщение *М*, осаждаемое в контейнер для передачи или хранения (например, с целью доказательства или защиты авторских прав на документ-контейнер; здесь речь может идти о невидимых цифровых водяных знаках (ЦВЗ));

• ключи, или ключевая информация, *K* системы, выполняющие ту же функцию, что и криптографические ключи; ключей может быть несколько, в соответствии с этим современные стеганосистемы характеризуют как многоключевые: один ключ отождествляется с методом встраивания/извлечения тайной информации, другой – с выбором элементов (например, битов) контейнера для его модификации при осаждении тайной информации, третий – для предварительного (перед встраиванием) преобразования тайной информации (например, на основе помехоустойчивого кодирования, сжатия или зашифрования) и т. д.;

• контейнер со встроенным сообщением, или стеганоконтейнер, *S*, который передается по открытому каналу, также являющемуся важным компонентом анализируемой системы; стеганоконтейнер будем именовать также стеганосообщением;

• для полноты упомянем также субъектов системы: отправителя и получателя.

В зависимости от формата документа-контейнера цифровую (или компьютерную) стеганографию подразделяют на классы:

• аудиостеганография;

• видеостеганография;

• графическая стеганография;

• текстовая стеганография;

• и др.

Стеганографической системой ∑ будем называть совокупность сообщений *M*, контейнеров *C*, ключей *K*, стеганосообщений (заполненных контейнеров) *S* и преобразований (прямого *F* и обратного *F*–1), которые их связывают:

∑ = (*M*, *C*, *K*, *S*, *F*, *F–1*).

Метод НЗБ основывается на ограниченных способностях зрения или слуха человека, вследствие чего людям тяжело различать незначительные вариации цвета или звука.

# Ход работы

Задача: необходимо разработать приложение, реализующее метод НЗБ.

Метод getColorMatrix предназначен для создания цветовой матрицы из заданного изображения, где каждый пиксель нового изображения будет содержать только младшие биты цветовых каналов (красного, зеленого и синего). Это полезно для анализа или визуализации информации, скрытой в изображении, например, в контексте стеганографии. В качестве входных параметров принимает путь к исходному изображению (файл), из которого будет извлекаться информация о цветах, путь, по которому будет сохранено новое изображение (цветовая матрица), созданное на основе исходного изображения. Код метода представлен в листинге 2.1.

public static void getColorMatrix(String imagePath, String outputPath)

throws IOException, ImageReadException, ImageWriteException {

BufferedImage image = Imaging.getBufferedImage(new File(imagePath));

BufferedImage newImage = new BufferedImage(image.getWidth(), image.getHeight(), BufferedImage.TYPE\_INT\_ARGB);

for (int y = 0; y < image.getHeight(); y++) {

for (int x = 0; x < image.getWidth(); x++) {

int rgb = image.getRGB(x, y);

int red = (rgb >> 16) & 0xFF;

int green = (rgb >> 8) & 0xFF;

int blue = rgb & 0xFF;

int newRed = (red & 0x01) \* 255;

int newGreen = (green & 0x01) \* 255;

int newBlue = (blue & 0x01) \* 255;

int newColor = (newRed << 16) | (newGreen << 8) | newBlue | 0xFF000000;

newImage.setRGB(x, y, newColor);

}

}

Imaging.writeImage(newImage, new File(outputPath), ImageFormats.BMP);

}

Листинг 2.1 – Код метода для создания цветовой матрицы

Перечисление Method определяет два возможных метода для работы с изображениями в контексте стеганографии. ROWS указывает, что обработка пикселей будет происходить построчно. То есть сначала обрабатывается вся строка, а затем переходит к следующей строке. COLS указывает, что обработка пикселей будет происходить по столбцам. То есть сначала обрабатывается весь столбец, а затем переходит к следующему столбцу. Код перечисления приведен в листинге 2.2.

public enum Method {

ROWS, COLS

}

Листинг 2.2 – Код перечисления

Метод embedMessage предназначен для встраивания сообщения в изображение с использованием стеганографии. Он изменяет младшие биты цветовых каналов пикселей изображения, чтобы зашифровать информацию без заметных изменений в изображении. В качестве входных параметров принимает путь к исходному изображению, в которое будет встроено сообщение, сообщение, которое нужно встроить в изображение, путь для сохранения нового изображения с встроенным сообщением, и метод обработки пикселей (ROWS или COLS), который определяет порядок, в котором будут обрабатываться пиксели. Метод embedMessage позволяет встроить текстовое сообщение в изображение, используя младшие биты цветовых каналов, что делает изменения незаметными для глаза. Этот метод является основным методом стеганографии в вашем коде, позволяющим скрывать информацию в изображениях. Код метода приведен в листинге 2.3.

public static void embedMessage(String containerPath, String message, String outputImagePath, Method method) throws IOException, ImageReadException, ImageWriteException {

BufferedImage containerImage = Imaging.getBufferedImage(new File(containerPath));

byte[] messageBytes = message.getBytes();

StringBuilder messageBits = new StringBuilder();

for (byte b : messageBytes) {

messageBits.append(String.format("%8s", Integer.toBinaryString(b & 0xFF)).replace(' ', '0'));

}

int maxMessageBits = (containerImage.getWidth() \* containerImage.getHeight()) \* 3;

if (messageBits.length() > maxMessageBits) {

throw new IllegalArgumentException("Message is too large for the container");

}

int messageBitIndex = 0;

int xMax, yMax;

if (method == Method.ROWS) {

xMax = containerImage.getWidth();

yMax = containerImage.getHeight();

} else {

xMax = containerImage.getHeight();

yMax = containerImage.getWidth();

}

for (int y = 0; y < yMax; y++) {

for (int x = 0; x < xMax; x++) {

int rgb = method == Method.ROWS ? containerImage.getRGB(x, y) : containerImage.getRGB(y, x);

int red = (rgb >> 16) & 0xFF;

int green = (rgb >> 8) & 0xFF;

int blue = rgb & 0xFF;

int newRed = red, newGreen = green, newBlue = blue;

if (messageBitIndex < messageBits.length()) {

newRed = (messageBits.charAt(messageBitIndex++) == '1') ? (red | 1) : (red & ~1);

}

if (messageBitIndex < messageBits.length()) {

newGreen = (messageBits.charAt(messageBitIndex++) == '1') ? (green | 1) : (green & ~1);

}

if (messageBitIndex < messageBits.length()) {

newBlue = (messageBits.charAt(messageBitIndex++) == '1') ? (blue | 1) : (blue & ~1);

}

int newColor = (newRed << 16) | (newGreen << 8) | newBlue | 0xFF000000;

if (method == Method.ROWS) {

containerImage.setRGB(x, y, newColor);

} else {

containerImage.setRGB(y, x, newColor);

}

if (messageBitIndex >= messageBits.length()) {

break;

}

}

if (messageBitIndex >= messageBits.length()) {

break;

}

}

Imaging.writeImage(containerImage, new File(outputImagePath), ImageFormats.BMP);

}

Листинг 2.3 – Код метода для встраивания текстового сообщения в изображение

Метод extractMessage извлекает сообщение, которое было встроено в изображение, анализируя младшие биты цветовых каналов пикселей. В качестве входных параметров принимает путь к изображению, из которого нужно извлечь сообщение, метод обработки пикселей (ROWS или COLS), который определяет порядок обхода пикселей. Метод extractMessage позволяет извлекать скрытые сообщения из изображений, анализируя младшие биты цветовых каналов. Он использует определенный метод обхода пикселей и учитывает конец сообщения, который определяется тремя нулями подряд. Этот метод является ключевым для стеганографических приложений, позволяя восстанавливать скрытую информацию. Код метода представлен в листинге 2.4.

public static String extractMessage(String imagePath, Method method) throws IOException, ImageReadException {

BufferedImage containerImage = Imaging.getBufferedImage(new File(imagePath));

StringBuilder messageBits = new StringBuilder();

int messageBitIndex = 0;

int endZeroGroup = 0;

int xMax, yMax;

if (method == Method.ROWS) {

xMax = containerImage.getWidth();

yMax = containerImage.getHeight();

} else {

xMax = containerImage.getHeight();

yMax = containerImage.getWidth();}

for (int y = 0; y < yMax; y++) {

for (int x = 0; x < xMax; x++) {

int rgb = method == Method.ROWS ? containerImage.getRGB(x, y) : containerImage.getRGB(y, x);

int red = (rgb >> 16) & 0xFF;

int green = (rgb >> 8) & 0xFF;

int blue = rgb & 0xFF;

messageBits.append(red & 1);

messageBitIndex++;

if (messageBitIndex < (xMax \* yMax \* 3)) {

messageBits.append(green & 1);

messageBitIndex++;}

if (messageBitIndex < (xMax \* yMax \* 3)) {

messageBits.append(blue & 1);

messageBitIndex++;}

if (messageBits.length() >= 3 && messageBits.substring(messageBits.length() - 3).equals("000")) {

endZeroGroup++;

} else {

endZeroGroup = 0;}

if (endZeroGroup >= 3) {

break;

}}

if (endZeroGroup >= 3) {

break;

}}

return convertBinaryToString(messageBits.toString());}

Листинг 2.4 – Код метода для извлечения сообщения из изображения

Метод convertBinaryToString является полезным для работы с двоичными данными, позволяя восстанавливать текстовую информацию из двоичного представления. Метод convertBinaryToString принимает строку, содержащую двоичные данные, и преобразует ее в строку символов (текст). Код метода представлен в листинге 2.5.

private static String convertBinaryToString(String binaryString) {

StringBuilder text = new StringBuilder();

for (int i = 0; i < binaryString.length(); i += 8) {

String byteString = binaryString.substring(i, Math.min(i + 8, binaryString.length()));

int charCode = Integer.parseInt(byteString, 2);

text.append((char) charCode);}

return text.toString();}

Листинг 2.5 – Метод для извлечения сообщения

Результат работы приложения с исходным текстом «Maxim Stanchik Andreevich» и внедрением по строкам представлен на рисунке 2.1.

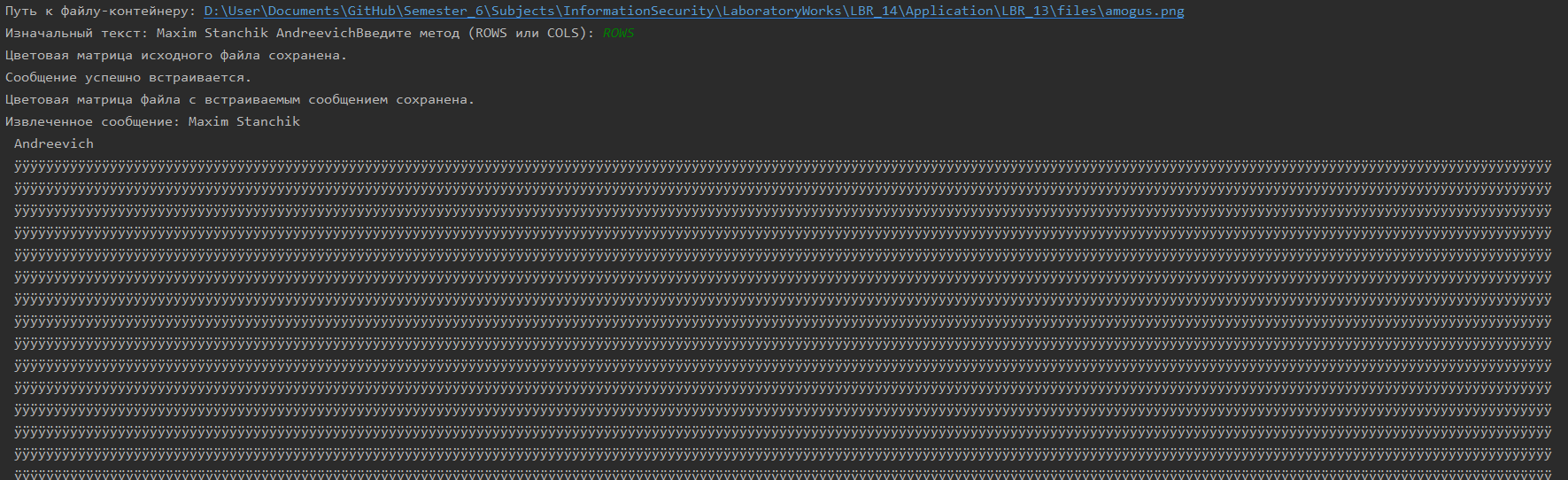


Рисунок 2.1 – Результат работы приложения

Исходный контейнер amogus.png представлен на рисунке рисунке 2.4.



Рисунок 2.2 – Исходный контейнер

Контейнер со встроенным сообщением представлен на рисунке 2.5.



Рисунок 2.3 – Контейнер со встроенным сообщением

Визуально эти два контейнера ничем не отличаются. Для обнаружения следов осаждения сообщения необходимо сформировать цветовые матрицы.

Цветовая матрица исходного контейнера представлена на рисунке 2.4.

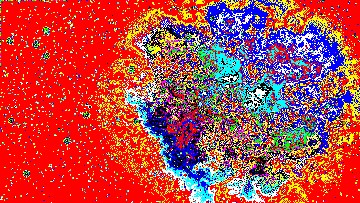


Рисунок 2.4 – Цветовая матрица исходного контейнера

Цветовая матрица контейнера с внедренным сообщением (по строкам) представлена на рисунке 2.5.

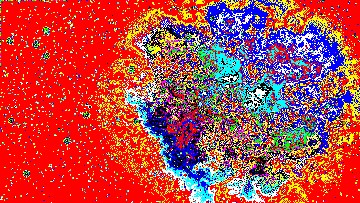


Рисунок 2.5 – Цветовая матрица контейнера с внедренным сообщением (внедряли по строкам)

Цветовая матрица контейнера с внедренным сообщением (по столбцам) представлена на рисунке 2.6.

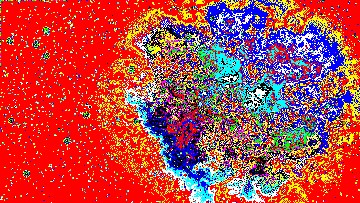


Рисунок 2.6 – Цветовая матрица контейнера с внедренным сообщением (внедряли по столбцам)

**Вывод:** в ходе выполнения лабораторной работы были изучены стеганографические методы встраивания/извлечения тайной информации с использованием электронного файла-контейнера на основе преобразования наименее значащих битов (НЗБ). Также было создано приложение, реализующее метод НЗБ.