МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Лабораторная работа № 14**

**ИССЛЕДОВАНИЕ СТЕГАНОГРАФИЧЕСКОГО**

**МЕТОДА НА ОСНОВЕ ПРЕОБРАЗОВАНИЯ**

**НАИМЕНЕЕ ЗНАЧАЩИХ БИТОВ**

Разработал: Бай И.О.

ФИТ 3 курс 5 группа

Преподаватель: Савельева М.Г.

Минск 2023

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

1. **Теоретические сведения**

Определение 1. 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.

При этом скрытый канал организуется на базе и внутри открытого канала с использованием особенностей восприятия информации. «Скрытость» канала передачи тайной информации отличает стеганографию от криптографии: в первом случае тайной является сам факт наличия канала (передачи информации).

 Определение 2. Абстрактно стеганографическая система обычно определяется как некоторое множество отображений одного пространства (множества возможных сообщений М) в другое пространство (множество возможных стеганосообщений S), и наоборот.

Определение 3. 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).

Как видим, сущностью рассматриваемой системы является тайное хранение или передача одной информации в другой информации, которая является открытой.

Большинство исследований в предметной области посвящено использованию в качестве стеганоконтейнеров изображений (текст также можно рассматривать как изображение). Это обусловлено следующими причинами:

• относительно большим объемом цифрового представления изображений, что позволяет внедрять большой объем данных;

• заранее известным размером контейнера, отсутствием ограничений, накладываемых требованиями реального времени;

• наличием в большинстве реальных изображений текстурных областей, имеющих шумовую структуру и хорошо подходящих для встраивания информации;

• слабой чувствительностью человеческого глаза к незначительным изменениям цветов изображения, его яркости, контрастности, содержанию в нем шума, искажениям вблизи контуров;

• хорошо разработанными в последнее время методами цифровой обработки изображений.

1. **Практическая часть**

Для выполнения задания было разработано приложение, реализующее метод встраивания сообщения в младшие биты, кодирующие цвет его пикселей. Для этого используется функция, фрагмент кода которой приведён в листинге 14.1.

|  |
| --- |
| for (int i = 0; i < bmpCoded.Width && k < Message\_Binary.Length; i++)  {  for (int j = 1; j < bmpCoded.Height && k < Message\_Binary.Length; j++)  {  color = bmpCoded.GetPixel(i, j);  Red\_Binary = Convert.ToString(color.R, 2).PadLeft(8, '0');  Red\_Binary = ReplaceLastBitOfByte(Red\_Binary, Message\_Binary[k]);  red = Convert.ToInt32(Red\_Binary, 2);  k++;  if (k >= Message\_Binary.Length)  {  bmpCoded.SetPixel(i, j, Color.FromArgb(red, color.G, color.B));  break;  }  Green\_Binary = Convert.ToString(color.G, 2).PadLeft(8, '0');  Green\_Binary = ReplaceLastBitOfByte(Green\_Binary, Message\_Binary[k]);  green = Convert.ToInt32(Green\_Binary, 2);  k++;  if (k >= Message\_Binary.Length)  {  bmpCoded.SetPixel(i, j, Color.FromArgb(red, green, color.B));  break;  }  Blue\_Binary = Convert.ToString(color.B, 2).PadLeft(8, '0');  Blue\_Binary = ReplaceLastBitOfByte(Blue\_Binary, Message\_Binary[k]);  blue = Convert.ToInt32(Blue\_Binary, 2);  k++;    bmpCoded.SetPixel(i, j, Color.FromArgb(red, green, blue));  }  }  int jj = bmpCoded.Height - 1;  for (int i = 0; i < bmpCoded.Width ; i++)  {  color = bmpCoded.GetPixel(i, jj);  Green\_Binary = Convert.ToString(color.G, 2).PadLeft(8, '0');  if(i%2 == 0)  {  Green\_Binary = ReplaceLastBitOfByte(Green\_Binary, '0');  }  else  {  Green\_Binary = ReplaceLastBitOfByte(Green\_Binary, '1');  }    green = Convert.ToInt32(Green\_Binary, 2);    bmpCoded.SetPixel(i, jj, Color.FromArgb(color.R, green, color.B));  } |

Листинг 14.1 – Процесс внедрения скрытой последовательности в документ

Код является частью приложения для стеганографии, которое кодирует сообщение в изображение. Циклы for повторяются по всем пикселям в растровом изображении (bmpCoded). Внешний цикл повторяется по ширине изображения, а внутренний цикл повторяется по высоте изображения. Переменная "k" - это счетчик, который отслеживает биты двоичного представления сообщения.

Внутри вложенного цикла for код извлекает значения RGB текущего пикселя (цвета) и преобразует каждое значение RGB в 8-битное двоичное представление (Red\_Binary, Green\_Binary, Blue\_Binary). Затем младший значащий бит (LSB) каждого цветового байта заменяется соответствующим битом сообщения с помощью функции ReplaceLastBitOfByte(). Полученное двоичное число затем преобразуется обратно в целое число (красное, зеленое, синее).

Если достигнут конец сообщения (Message\_Binary), код записывает оставшиеся пиксели с исходными значениями RGB.

Второй цикл for используется для изменения младшего значащего бита зеленого компонента последней строки растрового изображения. LSB каждого зеленого байта устанавливается на определенное значение ('0' или '1') в зависимости от значения координаты x (i). Это делается для указания конца сообщения при декодировании изображения.

В первую строку пикселей заносится информация о длине содержащегося в изображении сообщения.

Для извлечения сообщения используется функция, фрагмент кода которой приведён в листинге 14.2.

|  |
| --- |
| for (int i = 0; i < bmpOpenedCoded.Width ; i++)  {  color = bmpOpenedCoded.GetPixel(i, jj);  Green\_Binary = Convert.ToString(color.G, 2).PadLeft(8, '0');  if (i % 2 == 0)  {  if(Green\_Binary[7] != '0')  {  flag = false;  break;  }  }  else  {  if (Green\_Binary[7] != '1')  {  flag = false;  break;  }  }  }  if(flag == true)  {  for (int i = 0; i < bmpOpenedCoded.Width ; i++)  {  color = bmpOpenedCoded.GetPixel(i, 0);  Red\_Binary = Convert.ToString(color.R, 2).PadLeft(8, '0');  Text\_Len\_Binary += Red\_Binary[7];  }  Text\_Len = Convert.ToInt32(Text\_Len\_Binary, 2);  for (int i = 0; i < bmpOpenedCoded.Width && k < Text\_Len \* 13; i++)  {  for (int j = 1; j < bmpOpenedCoded.Height && k < Text\_Len \* 13; j++)  {  color = bmpOpenedCoded.GetPixel(i, j);  Red\_Binary = Convert.ToString(color.R, 2).PadLeft(8, '0');  OutPutText\_Binary += Red\_Binary[7];  k++;  if (k >= Text\_Len \* 13)  {  break;  }  Green\_Binary = Convert.ToString(color.G, 2).PadLeft(8, '0');  OutPutText\_Binary += Green\_Binary[7];  k++;  if (k >= Text\_Len \* 13)  {  break;  }  Blue\_Binary = Convert.ToString(color.B, 2).PadLeft(8, '0');  OutPutText\_Binary += Blue\_Binary[7];  k++;  }  } |

Листинг 14.2 – Процесс извлечения скрытой последовательности из документа

Первый цикл for выполняет итерацию по первой строке закодированного растрового изображения (bmpOpenedCoded) и извлекает 8-битное двоичное представление зеленого компонента каждого пикселя. Условные операторы проверяют младший значащий бит (LSB) двоичного числа, чтобы определить, представляет ли оно '0' или '1'. Он делает это, проверяя, равен ли LSB каждого четного пикселя "0", а каждого нечетного пикселя - "1". Если шаблон не соблюдается, это означает, что скрытого сообщения нет, и переменной flag присваивается значение false, чтобы прервать цикл.

Оператор if (flag == true) выполняется, если соблюдается шаблон, что означает, что в закодированном изображении присутствует действительное сообщение. Этот блок кода извлекает длину сообщения, закодированного в растровом изображении, путем считывания 8-го бита (MSB) красного компонента каждого пикселя в первой строке растрового изображения. 8-й бит из каждого пикселя объединяется для формирования двоичной строки Text\_Len\_Binary, из которой вычисляется длина сообщения Text\_Len (в символах).

Затем второй цикл for выполняет итерацию по всем пикселям в растровом изображении, начиная со второй строки, и извлекает сообщение, считывая 8-й бит (MSB) красного, зеленого и синего компонентов пикселя. Биты объединяются в двоичную строку OutPutText\_Binary. Цикл продолжается до тех пор, пока длина извлеченного сообщения не станет равной Text\_Len \* 13 битам, что является ожидаемой длиной закодированного сообщения. Обратите внимание, что длина извлеченного сообщения указана в битах, поскольку каждый пиксель кодирует 3 бита сообщения (по 1 биту на цветовой компонент).

После этого цикла двоичная строка OutPutText\_Binary содержит извлеченное сообщение в двоичном формате, которое затем при необходимости может быть преобразовано обратно в обычный текст.

Был произведён визуальный анализ в ходе которого не было найдено различий в изображении с зашифрованным сообщением и без, это, а так же цветовые матрицы представлены на рис. 14.1



Рис 14.1 – визуальный анализ и цветовые матрицы

На рис. 14.2 и 14.3 представлены два метода размещения битового потока построчно и по столбцам.

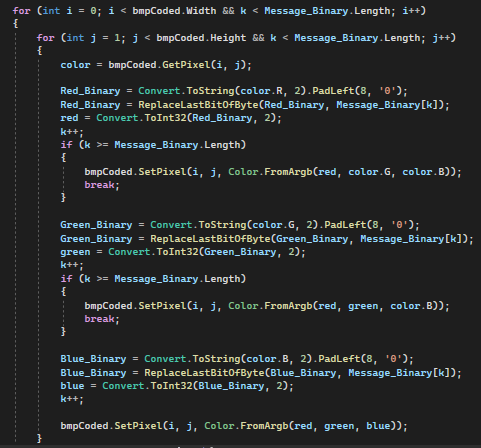


Рис. 14.2 – построчное размещение битового потока



Рис. 14.3 – размещение битового потока по столбцам

**Вывод:** исследование было посвящено стеганографическому методу на основе изменения наименее значащих битов. Этот метод позволяет скрыть информацию в носителе (например, изображении или аудиофайле) путем замены наименее значимых битов секретными данными. Предполагается, что такие изменения не будут заметны для человеческого восприятия и не вызовут подозрений у посторонних. Этот метод может быть полезным для обмена конфиденциальной информацией или защиты данных.