Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Исследование стеганографического метода на основе преобразования наименее значащих битов

Студент: Авсюкевич П.В.

ФИТ 3 курс 5 группа

Минск 2023

Метод наименее значимых битов опирается на ограниченные возможности зрения или слуха человека, из-за чего им трудно различить малозаметные изменения цвета или звука.

Младшие биты каждой компоненты цвета пикселя вносят незначительный вклад в изображение по сравнению со старшими битами. Замена одного или даже нескольких младших битов для человеческого глаза будет почти незаметна, так как человек способен различить около ста пятидесяти цветовых оттенков. Картинка, которая является основанием для данного преобразования прдествалена на рисунке 1.



Рисунок 1 – Исходное изображение

Цветовая матрица наименее значащих битов для исходного изображения представлена на рисунке 3.

На листинге 1 можно ознакомиться с функцией встраивания сообщения.

|  |
| --- |
| public static void EmbedMessageByRows(string containerPath, string message, string outputImagePath){  Bitmap containerImage = new Bitmap(containerPath);  byte[] messageBytes = Encoding.UTF8.GetBytes(message);  BitArray messageBits = new BitArray(messageBytes);  int maxMessageBits = (containerImage.Width \* containerImage.Height) \* 3;  if (messageBits.Length > maxMessageBits)  throw new Exception("Сообщение слишком большое для данного контейнера");int messageBitIndex = 0;  for (int y = 0; y < containerImage.Height; y++){  for (int x = 0; x < containerImage.Width; x++){  Color pixel = containerImage.GetPixel(x, y);  if (messageBitIndex < messageBits.Length){  BitArray redBits = new BitArray(new byte[] { pixel.R });  BitArray greenBits = new BitArray(new byte[] { pixel.G });  BitArray blueBits = new BitArray(new byte[] { pixel.B });  if (messageBits[messageBitIndex])redBits[0] = true;  else redBits[0] = false;  messageBitIndex++;  if (messageBitIndex < messageBits.Length)  {  if (messageBits[messageBitIndex])  greenBits[0] = true;  else  greenBits[0] = false;  messageBitIndex++;  }  if (messageBitIndex < messageBits.Length)  {  if (messageBits[messageBitIndex])  blueBits[0] = true;  else  blueBits[0] = false;  messageBitIndex++;}  byte[] newRedBytes = new byte[1];  byte[] newGreenBytes = new byte[1];  byte[] newBlueBytes = new byte[1];  redBits.CopyTo(newRedBytes, 0);  greenBits.CopyTo(newGreenBytes, 0);  blueBits.CopyTo(newBlueBytes, 0);  Color newPixel = Color.FromArgb(newRedBytes[0], newGreenBytes[0], newBlueBytes[0]);  containerImage.SetPixel(x, y, newPixel); }  containerImage.Save(outputImagePath, ImageFormat.Png); |

Листинг 1 – Реализация встраивания текста

Результат изображения представлен на рисунке 3.



Рисунок 2– Изображение с встроенным текстом по строкам

Цветовая карта НЗБ данного изображения представлена на рисунке 4.



Рисунок 3 – Цветовая матрица НЗБ изображения

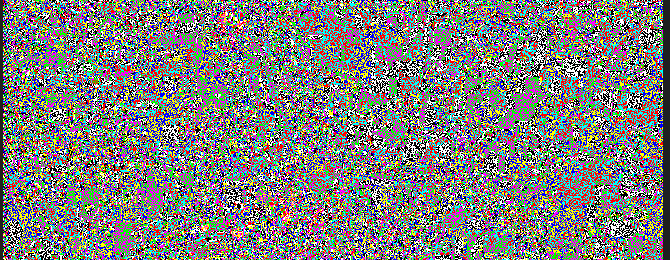


Рисунок 4– Цветовая матрица НЗБ с встроенным текстом

По изображению нельзя обнаружить никаких изменений. Тем не менее, в крайнем левом углу цветовой матрицы можно заметить небольшое различие, где наше сообщение было встроено.

Для получения сообщения из изображения была реализована функция представленная на листинге 2.

|  |
| --- |
| public static string ExtractMessageByRows(string imagePath)  {  Bitmap containerImage = new Bitmap(imagePath);  int messageBitsLength = (containerImage.Width \* containerImage.Height) \* 3;  BitArray messageBits = new BitArray(messageBitsLength);  int messageBitIndex = 0;  for (int y = 0; y < containerImage.Height; y++)  {  for (int x = 0; x < containerImage.Width; x++)  {  Color pixel = containerImage.GetPixel(x, y);  BitArray redBits = new BitArray(new byte[] { pixel.R });  BitArray greenBits = new BitArray(new byte[] { pixel.G });  BitArray blueBits = new BitArray(new byte[] { pixel.B });  bool redBit = redBits[0];  messageBits[messageBitIndex] = redBit;  messageBitIndex++;  if (messageBitIndex < messageBitsLength)  {  bool greenBit = greenBits[0];  messageBits[messageBitIndex] = greenBit;  messageBitIndex++;  }  if (messageBitIndex < messageBitsLength)  {  bool blueBit = blueBits[0];  messageBits[messageBitIndex] = blueBit;  messageBitIndex++;  if (messageBitIndex >= messageBitsLength)  break;  if (messageBitIndex >= messageBitsLength)  break;  }  byte[] messageBytes = new byte[(messageBits.Length + 7) / 8];  messageBits.CopyTo(messageBytes, 0);  string message = Encoding.UTF8.GetString(messageBytes);  var sb = new StringBuilder("");  foreach (var ch in message)  {  if (ch > 255)  break;  sb.Append(ch);  }  return sb.ToString(); } |

Листинг 2 – Реализация функции получения сообщения

Результат выполнения представлен на рисунке 5.



Рисунок 5 – Вывод функции встраивания текста по строкам

Также была реализована функция чтения по столбцам, которая отличается от представленной в листенге методом чтения и записи сообщения.

Для проверки было выполнено зашифрование по строкам большого текста результат в цветовой карте представлен на рисунке 6, 7.



Рисунок 6 – Тепловая карта исходного текста

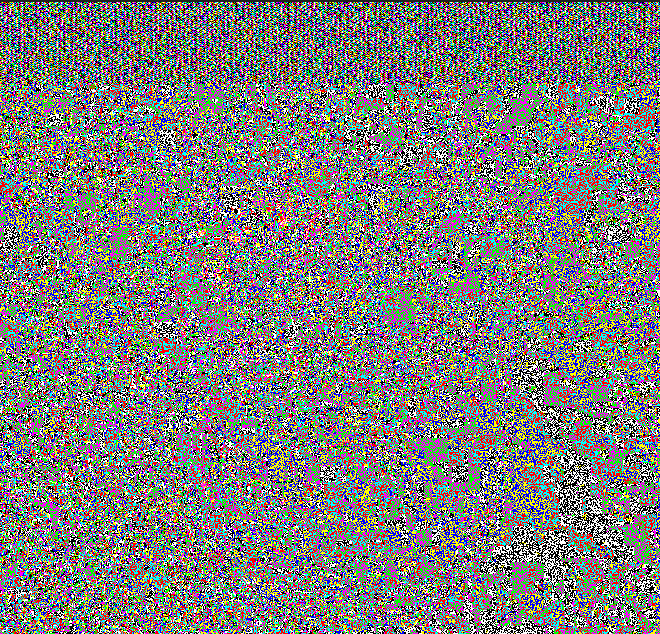


Рисунок 7 – Тепловая карта с встроенным текстом по горизовнтали

Этот метод позволяет встраивать и извлекать скрытую информацию без существенного воздействия на внешний вид или качество контейнерного файла. Полученные знания позволили приобрести практические навыки программной реализации данного метода. В процессе работы была разработана функциональная программа, способная встраивать секретные сообщения в изображения и извлекать их обратно, сохраняя при этом внешний вид контейнера.