Пшенко Артем ФИТ 3-4

Отчет по лабораторной работе № 13

Исследование стеганографического метода на основе преобразования наименее значащих битов

Информационная безопасность

Вариант 9

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.

Метод НЗБ основывается на ограниченных способностях зрения или слуха человека, вследствие чего людям тяжело различать незначительные вариации цвета или звука. Рассмотрим это на примере 24-битного растрового RGB-изображения. Как известно, каждая точка кодируется тремя байтами. Каждый байт определяет интенсивность красного (Red), зеленого (Green) и синего (Blue) цветов. Совокупность интенсивностей цвета в каждом из трех каналов определяет оттенок пикселя.

Замена одного или даже нескольких младших битов для человеческого глаза будет почти незаметна, поскольку реально человек может различать около полторы сотни цветовых оттенков.

Вторым способом осаждения сообщения является операция «размазывания» его по контейнеру. Для операций «размазывания» сообщения по контейнеру могут применяться ключевой файл и пароль в виде текстов, которые символы которых заменяются числами и в совокупности определяют местоположение пикселя для записи в него части секретного сообщения. Простейшая реализация этого подхода показана на рис. 1.

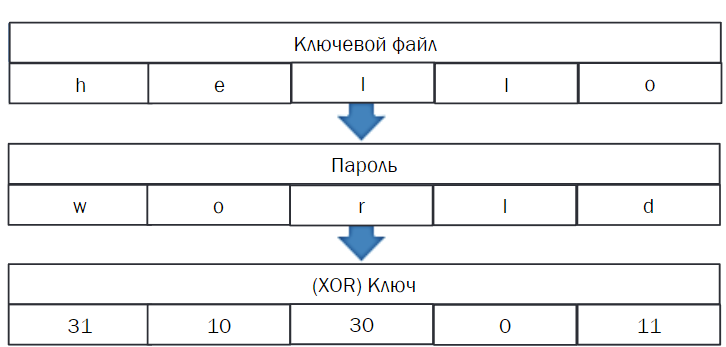


Рисунок 1. Пояснение к расчету местоположения пикселей для встраивания/извлечения сообщения

Исходный контейнер изображен на рисунке 2.



Рисунок 2. Исходный контейнер

Исходный код с реализацией данных алгоритмов, а также получением цветовых матриц R, G, B приведен в листинге 1.

Листинг 1. Реализация алгоритма НЗБ и с помощью пароля, а также генерация матриц.

|  |
| --- |
| from PIL import Image  from stegano import lsb  import numpy as np  import matplotlib.pyplot as plt  def generate\_message():  return "Pshenko Artyom Fyodorovich"  def embed\_message\_LSB(container\_file, message):  # Метод размещения битового потока с использованием младших значащих битов  secret = lsb.hide(container\_file, message)  secret.save("embedded\_LSB.jpg")  print("Сообщение успешно осаждено методом LSB.")  def embed\_message\_with\_password(container\_file, message, password):  # Метод размещения сообщения с использованием пароля  binary\_message = ''.join([format(ord(char), '08b') for char in message])  binary\_message += '00000000' # Null byte to indicate end of message  img = Image.open(container\_file)  img\_data = np.array(img)  stego\_image\_data = img\_data.copy()  # Преобразование пароля в бинарный формат  password\_binary = ''.join(format(ord(char), '08b') for char in password)  password\_length = len(password\_binary)  # Осаждение сообщения с использованием пароля  idx = 0  for i in range(len(img\_data)):  for j in range(len(img\_data[0])):  for k in range(3): # 3 цветовых канала  if idx < len(binary\_message):  # Применяем XOR шифрование с битами пароля к битам сообщения  stego\_image\_data[i][j][k] = (img\_data[i][j][k] & ~1) | (int(binary\_message[idx]) ^ int(password\_binary[idx % password\_length]))  idx += 1  else:  break  stego\_image = Image.fromarray(stego\_image\_data)  stego\_image.save("embedded\_with\_password.png")  print("Сообщение успешно осаждено с использованием пароля.")  # Извлечение сообщения с использованием пароля  img = Image.open(container\_file)  img\_data = np.array(img)  # Преобразование пароля в бинарный формат  password\_binary = ''.join(format(ord(char), '08b') for char in password)  password\_length = len(password\_binary)  extracted\_message = ""  idx = 0  for i in range(len(img\_data)):  for j in range(len(img\_data[0])):  for k in range(3): # 3 цветовых канала  if idx < 8: # Пропускаем первые 8 бит (маркер конца сообщения)  idx += 1  continue  # Применяем XOR шифрование с битами пароля к битам сообщения  extracted\_bit = (img\_data[i][j][k] & 1) ^ int(password\_binary[idx % password\_length])  extracted\_message += str(extracted\_bit)  idx += 1  # Находим маркер конца сообщения и извлекаем только биты сообщения  end\_idx = extracted\_message.find("00000000")  extracted\_message = extracted\_message[8:end\_idx]  # Преобразуем бинарное сообщение в строку символов ASCII  extracted\_text = ""  for i in range(0, len(extracted\_message), 8):  extracted\_text += chr(int(extracted\_message[i:i+8], 2))  return extracted\_text  def generate\_color\_matrix(container\_file):  # Формирование цветовой матрицы  img = Image.open(container\_file)  img\_data = np.array(img)    # Создаем пустую матрицу для каждого канала цвета (R, G, B)  color\_matrices = { "R": np.zeros\_like(img\_data[:,:,0]),  "G": np.zeros\_like(img\_data[:,:,1]),  "B": np.zeros\_like(img\_data[:,:,2]) }    # Заполняем матрицы значениями младших битов каждого канала  for channel, matrix in color\_matrices.items():  matrix.flat = img\_data[:,:,["R", "G", "B"].index(channel)].flatten() & 1    # Отображение цветовых матриц  fig, axs = plt.subplots(1, 3, figsize=(15, 5))  for i, (channel, matrix) in enumerate(color\_matrices.items()):  axs[i].imshow(matrix, cmap='gray')  axs[i].set\_title(f"Channel: {channel}")  axs[i].axis('off')  plt.show()  # Генерация сообщения и выбор пароля  message = generate\_message()  password = "mysecretpassword"  # Выбор и применение методов размещения сообщения  embed\_message\_LSB("container.jpg", message)  embed\_message\_with\_password("container.jpg", message, password)  # Формирование цветовых матриц  generate\_color\_matrix("container.jpg") |

Результат работы алгоритма приведен на рисунке 3.

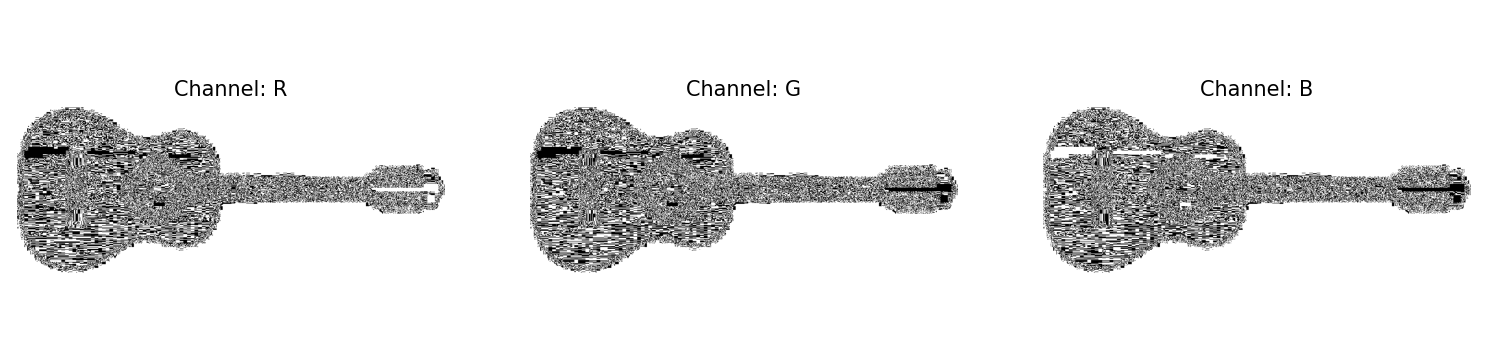


Рисунок 3. Цветовые матрицы R, G, B

Таким образом, в данной лабораторной работе был изучен метод НЗБ, с использованием пароля и XOR, выполнено осаждение/извлечение сообщения в контейнере, а также построены цветовые матрицы R, G, B.