Учреждение образования

«Белорусский государственный технологический университет»

**Кафедра информационных систем и технологий**

**«Отчёт по лабораторной работе №14**

“Исследование стеганографического метода на основе преобразования наименее значащих битов”

**Выполнил:** студент 4 курса

1 группы специальности ИСиТ

Халалеенко Андрей Николаевич

**Проверил:** преподаватель

Сазонова Дарья Владимировна

Минск 2024

**Разработать собственное приложение, в котором должен быть реализован метод НЗБ. При этом:**

**• выбор файла-контейнера – по согласованию с преподавателем;**

**• реализовать два варианта осаждаемого/извлекаемого сообщения:**

**− собственные фамилия, имя и отчество;**

**− текстовая часть отчета по одной из выполненных лабораторных работ;**

**• реализовать два метода (на собственный выбор) размещения битового потока осаждаемого сообщения по содержимому контейнера;**

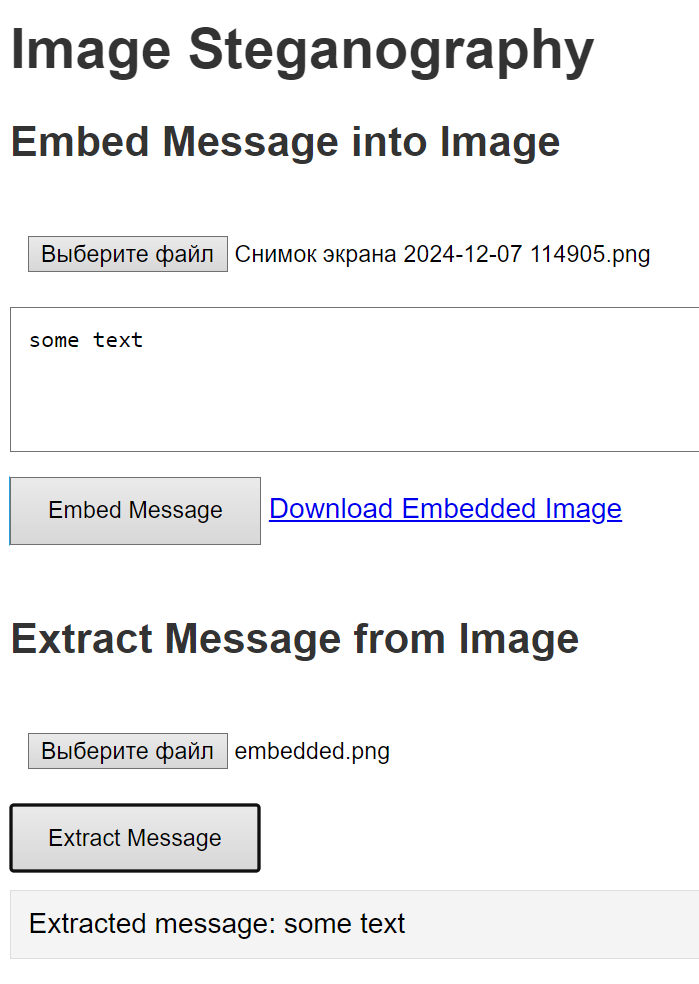
**• сформировать цветовые матрицы (по аналогии с рис. 12.7), отображающие каждый задействованный для осаждения уровень младших значащих битов контейнера;**

**• выполнить визуальный анализ (с привлечением коллег в качестве экспертов) стеганоконтейнеров с различным внутренним содержанием; сделать выводы на основе выполненного анализа**

Реализация приложения:

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <title>Image Steganography</title>  <style>  body {  font-family: Arial, sans-serif;  margin: 20px;  }  h1, h2 {  color: #333;  }  .container {  margin-bottom: 40px;  }  input[type="file"], textarea, input[type="text"] {  width: 100%;  padding: 10px;  margin-top: 10px;  box-sizing: border-box;  }  button {  padding: 10px 20px;  margin-top: 10px;  cursor: pointer;  }  canvas {  display: none;  margin-top: 20px;  }  .output {  margin-top: 10px;  padding: 10px;  background-color: #f4f4f4;  border: 1px solid #ddd;  word-break: break-all;  }  </style>  </head>  <body>  <h1>Image Steganography</h1>  <!-- Embed Message Section -->  <div class="container" id="embed-section">  <h2>Embed Message into Image</h2>  <input type="file" id="container-image" accept="image/png">  <textarea id="secret-message" rows="4" placeholder="Enter your secret message here..."></textarea>  <button id="embed-button">Embed Message</button>  <a id="download-link" style="display:none;">Download Embedded Image</a>  </div>  <!-- Extract Message Section -->  <div class="container" id="extract-section">  <h2>Extract Message from Image</h2>  <input type="file" id="embedded-image" accept="image/png">  <button id="extract-button">Extract Message</button>  <div class="output" id="extracted-message"></div>  </div>  <!-- Hidden Canvas for Image Processing -->  <canvas id="image-canvas"></canvas>  <script>  const messageBits = 8; // Number of bits for each character in the message  // Helper function to convert a string to a binary string  function stringToBinary(message) {  return message.split('').map(char => {  return char.charCodeAt(0).toString(2).padStart(8, '0');  }).join('');  }  // Helper function to convert binary string to text  function binaryToString(binary) {  const chars = binary.match(/.{1,8}/g);  if (!chars) return '';  return chars.map(byte => String.fromCharCode(parseInt(byte, 2))).join('');  }  // Embed Message Function  function embedMessage(containerImg, message) {  const canvas = document.getElementById('image-canvas');  const ctx = canvas.getContext('2d');  const binaryMessage = stringToBinary(message) + '00000000'; // Append null character to signify end  const totalBits = binaryMessage.length;  let bitIndex = 0;  canvas.width = containerImg.width;  canvas.height = containerImg.height;  ctx.drawImage(containerImg, 0, 0);  const imageData = ctx.getImageData(0, 0, canvas.width, canvas.height);  const data = imageData.data;  for (let i = 0; i < data.length; i += 4) { // Iterate over each pixel  if (bitIndex < totalBits) {  // Modify Red, Green, and Blue channels  for (let channel = 0; channel < 3; channel++) {  if (bitIndex < totalBits) {  const bit = parseInt(binaryMessage[bitIndex]);  data[i + channel] = (data[i + channel] & 0xFE) | bit;  bitIndex++;  }  }  } else {  break;  }  }  ctx.putImageData(imageData, 0, 0);  return canvas.toDataURL('image/png');  }  // Extract Message Function  function extractMessage(embeddedImg) {  const canvas = document.getElementById('image-canvas');  const ctx = canvas.getContext('2d');  canvas.width = embeddedImg.width;  canvas.height = embeddedImg.height;  ctx.drawImage(embeddedImg, 0, 0);  const imageData = ctx.getImageData(0, 0, canvas.width, canvas.height);  const data = imageData.data;  let binaryMessage = '';  for (let i = 0; i < data.length; i += 4) { // Iterate over each pixel  for (let channel = 0; channel < 3; channel++) { // Red, Green, Blue  binaryMessage += (data[i + channel] & 0x01).toString();  }  }  // Split binary message into 8-bit chunks  const bytes = binaryMessage.match(/.{1,8}/g);  if (!bytes) return '';  let message = '';  for (let byte of bytes) {  if (byte === '00000000') break; // Null character signifies end of message  message += String.fromCharCode(parseInt(byte, 2));  }  return message;  }  // Handle Embed Button Click  document.getElementById('embed-button').addEventListener('click', () => {  const fileInput = document.getElementById('container-image');  const messageInput = document.getElementById('secret-message');  const downloadLink = document.getElementById('download-link');  if (fileInput.files.length === 0) {  alert('Please select an image to embed the message.');  return;  }  const message = messageInput.value;  if (message.length === 0) {  alert('Please enter a secret message to embed.');  return;  }  const img = new Image();  img.onload = () => {  const maxMessageLength = Math.floor((img.width \* img.height \* 3) / 8) - 1; // Reserve for null terminator  if (message.length > maxMessageLength) {  alert(`Message too long! Maximum length is ${maxMessageLength} characters.`);  return;  }  const embeddedDataURL = embedMessage(img, message);  downloadLink.href = embeddedDataURL;  downloadLink.download = 'embedded.png';  downloadLink.textContent = 'Download Embedded Image';  downloadLink.style.display = 'inline';  };  img.onerror = () => {  alert('Failed to load the image. Please try another image.');  };  img.src = URL.createObjectURL(fileInput.files[0]);  });  // Handle Extract Button Click  document.getElementById('extract-button').addEventListener('click', () => {  const fileInput = document.getElementById('embedded-image');  const output = document.getElementById('extracted-message');  if (fileInput.files.length === 0) {  alert('Please select an embedded image to extract the message.');  return;  }  const img = new Image();  img.onload = () => {  const message = extractMessage(img);  if (message.length === 0) {  output.textContent = 'No hidden message found or message is empty.';  } else {  output.textContent = `Extracted message: ${message}`;  }  };  img.onerror = () => {  alert('Failed to load the image. Please try another image.');  };  img.src = URL.createObjectURL(fileInput.files[0]);  });  </script>  </body>  </html> |

Результат выполнения:



Контейнер:



Контейнер с сообщением:

