**《信息安全概论》实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **姓名** | 陈鹏宇 | | **年级** | | 2020 |
| **学号** | 20204227 | | **专业、班级** | | 计算机科学与技术01班 |
| **实验名称** | 信息隐藏实验 | | | | |
| **实验时间** | 2023.10.27 | **实验地点** | | Ds3402 | |
| **实验成绩** |  | **实验性质** | | □验证性 □设计性 □综合性 | |
| **教师评价：**  □算法/实验过程正确； □源程序/实验内容提交 □程序结构/实验步骤合理；  □实验结果正确； □语法、语义正确； □报告规范；  评语：  评价教师签名（电子签名）： | | | | | |
| 1. **实验目的** 2. 学习并掌握图像信息隐藏的基本原理和方法 3. 实现基于LSB的信息隐藏和提取算法 | | | | | |
| 1. **实验项目内容** 2. 使用LSB算法在图片中隐藏如下信息：CQUWATERMASKEXP 3. 从被隐藏数据的图片中解析出如上信息，建议使用Matlab 4. 在实际应用中，隐藏信息量通常是不可预知的，同时，攻击者也很容易从最低位像素提取到隐藏信息。另一方面，如何确保信息来源于正确的发送者？针对这些问题，请设计完整的保密通信方案。 | | | | | |
| 1. **实验设计**   任何多媒体信息在数字化时都会产生物理随机噪声，而人的感官系统对这些随机噪声并不敏感，通过使用秘密信息比特替换随机噪声，从而实现信息隐藏。在图像中，高位平面对图像感官质量起主要作用，去除图像最低几个位平面并不会造成画面质量的下降。利用这个原理可用秘密信息（或称水印信息）替代载体图像低位平面以实现信息嵌入。  LSB算法选用最低位平面来嵌入信息，最低位平面对图像的视觉效果影响最轻微，因此在视觉上很难察觉。作为大数据量的信息隐藏方法，LSB在保密通信中仍占据相当重要的地位。  **LSB主要步骤：**  **加密：**  1. 读入图片  2. 准备待隐藏的信息，将其转换为二进制  3. 遍历图像，对像素的最低1bit置0，同时在该比特位写入1位二进制表示隐藏的信息  **解密：**  1. 预知隐藏信息量（等同于key）  2. 提取出像素的最低1bit，组合成连续bit数据，转换为ASCII码对比是否与隐藏信息一致 | | | | | |
| 1. **实验过程或算法**   **实现步骤：**   1. 将图像文件中的所有像素点以RGB形式分隔开，并将各个颜色分量转换成二进制表示 2. 把每个颜色分量值的最后一位全部设置成0，对图像得影响非常细微，不会影响图像的显示格式 3. 信息嵌入：将水印字符转化为二进制字符串，并将这些信息依次填入颜色分量的最低位上，即可完成信息的嵌入 4. 信息提取：将图像像素的最低位依次提取出来，并进行拼接，即可得到原始信息   **源代码：**  from PIL import Image  *# 字符to八位二进制*  def str\_to\_binary(s):      return ''.join(format(ord(c), '08b') for c in s)  *# 二进制to字符*  def binary\_to\_str(b):      return ''.join(chr(int(b[i:i+8], 2)) for i in range(0, len(b), 8) if i+8 <= len(b))  *# 低位置0*  def setZero(rgb):      for i in range(rgb.size[0]): *# 将图片每个像素rgb三元组的最后一位全部置零*          for j in range(rgb.size[1]):              r = rgb.getpixel((i, j))[0]              g = rgb.getpixel((i, j))[1]              b = rgb.getpixel((i, j))[2]              if(r%2==1):                  r=r-1              if (g % 2 == 1):                  g = g - 1              if (b % 2 == 1):                  b = b - 1              rgb.putpixel((i,j), (r,g,b))  *# 隐藏*  def hide(rgb, output\_path, info):      cur= 0      for i in range(rgb.size[0]):          for j in range(rgb.size[1]):  *# print(rgb.getpixel((i, j)))*              if(cur >= len(info)): *# 指针越界及时跳出*                  break              r = rgb.getpixel((i, j))[0]+ (ord(info[cur])-ord('0')) *# 将水印写入最后一位*              if (cur+1 >= len(info)):                  break              g = rgb.getpixel((i, j))[1]+ (ord(info[cur+1])-ord('0'))              if (cur+2 >= len(info)):                  break              b = rgb.getpixel((i, j))[2]+ (ord(info[cur+2])-ord('0'))              cur= cur+ 3              rgb.putpixel((i, j), (r, g, b))      rgb.save(output\_path) *# 保存带水印的图片*  def extract(output\_path, info) -> str:      carrier= Image.open(output\_path) *# 打开带水印的图片*      rgb= carrier.convert("RGB") *#转为RGB形式*      output\_binary = ""      for i in range(rgb.size[0]):*# 遍历像素，提取rgb最后一位*          for j in range(rgb.size[1]):              if(len(output\_binary) >= len(info)): *# 提取完成及时跳出*                  break              output\_binary= output\_binary + str(rgb.getpixel((i, j))[0] % 2)              if (len(output\_binary) >= len(info)):                  break              output\_binary = output\_binary + str(rgb.getpixel((i, j))[1] % 2)              if (len(output\_binary) >= len(info)):                  break              output\_binary = output\_binary + str(rgb.getpixel((i, j))[2] % 2)      return output\_binary  *# 隐藏信息*  hidden\_msg = "CQUWATERMASKEXP"  input\_path = "./input.bmp"  output\_path = "./output.bmp"  *# 将隐藏信息转为二进制*  str\_to\_binary(hidden\_msg)  carrier= Image.open(input\_path)  rgb\_carrier= carrier.convert('RGB') *# 以RGB模式打开*  setZero(rgb\_carrier) *# 最后一位置零*  hide(rgb\_carrier, output\_path, str\_to\_binary(hidden\_msg))  print("隐藏成功")  ans\_binary = extract(output\_path, str\_to\_binary(hidden\_msg))  binary\_to\_str(ans\_binary)  import matplotlib.pyplot as plt  img1 = plt.imread(input\_path)  img2 = plt.imread(output\_path)  plt.subplot(121)  plt.title("原图像")  plt.imshow(img1,cmap='gray')  plt.subplot(122)  plt.title("隐藏信息后图像")  plt.imshow(img2, cmap='gray')  *# 显示对比图*  plt.show() | | | | | |
| 1. **实验结果及分析和（或）源程序调试过程** 2. 基于LSB的信息隐藏和提取       在肉眼无法分辨的情况下，将信息隐藏到图片中并成功提取。   1. 保密通信方案  * 对于隐藏信息量不可预知，可以采取协议的方式，在隐藏信息前加上其长度。 * 对于攻击者也很容易从最低位像素提取到隐藏信息，有两种方法，一是随机存储在像素点中，用[混沌序列](https://jeit.ac.cn/cn/article/doi/10.11999/JEIT210763?viewType=HTML)非线性的插入图片，导致攻击者无从发现隐藏的密文。二是[信息嵌入过程](https://www.alljournals.cn/view_abstract.aspx?pcid=5B3AB970F71A803DEACDC0559115BFCF0A068CD97DD29835&cid=1319827C0C74AAE8D654BEA21B7F54D3&jid=0DA8F102900008C100F0F9CCE3E5F1BA&aid=36A68556F57F8BA8FDE7BD8DB78D5444&yid=99E9153A83D4CB11)中,调整载体信息最低两个比特位,使它所表示那一个比特位与秘密信息的比特位相一致,避免了传统LSB算法直接在载体信息最低位上嵌入秘密信息 * 对于如何确保信息来源于正确的发送者，我们可以使用数字签名技术。发送者使用其私钥对消息进行签名，接收者使用发送者的公钥验证签名。这样，接收者可以确保消息来自于正确的发送者，而不是被篡改或伪造的。      1. 具体流程  * 发送方利用隐藏信息计算出hash，并且利用私钥加密成MAC，添加到信息尾部；在信息头部添加其长度信息；通过构造Logistic映射函数生成序列，插入到对应像素位置 * 接收方首先提取出长度信息，然后取出后续的隐藏信息和mac，利用公钥解密解密出md并和hash信息进行对比，如果相同则证明验证了数字签名并鉴别了信息。 | | | | | |