**《信息安全概论》实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **姓名** | | *李燕琴* | **年级** | | | *2019级* |
| **学号** | | *20195633* | **专业、班级** | | | *计算机科学与技术（卓越）02班* |
| **实验名称** | **信息隐藏实验** | | | | | |
| **实验时间** | **2022-04-19** | | | **实验地点** | **DS3 402** | |
| **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）： | | | | | | |
| 一、实验目的   1. 学习并掌握图像信息隐藏的基本原理和方法 2. 实现基于LSB的信息隐藏和提取算法 | | | | | | |
| 二、实验项目内容  编程实现基于LSB的信息隐藏和提取算法 | | | | | | |
| 三、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：**  **1、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  **2、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 | | | | | | |
| 四、实验过程或算法   1. 使用LSB算法在图片中隐藏如下信息：CQUWATERMASKEXP 2. 从被隐藏数据的图片中解析出如上信息，建议使用Matlab 3. 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。   LSB算法源码：   1. # 导入包 2. **import** cv2 3. **import** time 4. **import** numpy as np 5. %matplotlib inline 6. **import** matplotlib as mpl 7. **import** matplotlib.pyplot as plt 8. # 支持中文 9. plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签 10. plt.rcParams['axes.unicode\_minus'] = False  # 用来正常显示负号  13. **def** str2bit(string): 14. n = len(string)\*8 15. a = np.zeros(n).astype(np.uint8) 16. cnt = 0 17. **while** 1: 18. i = cnt//8 19. j = cnt%8 20. **if** cnt==n: 21. **break** 22. a[cnt] = ((ord(string[i])<<j)&0x80)>>7 23. cnt+=1 24. **return** a 26. **def** bit2str(a): 27. res = "" 28. ch = 0 29. cnt = 0 30. **for** i **in** a: 31. ch = ch\*2+i 32. cnt += 1 33. **if** cnt **and** cnt%8==0: 34. **if** ch==0: 35. **break** 36. res += chr(ch) 37. ch = 0 38. **return** res 40. **def** encode(copyright,img\_path,save\_path): 41. ''''' 加密 ''' 42. # 读取原始图片 43. src\_data = cv2.imread(img\_path)[:,:,[2,1,0]] # BGR(0,1,2)转为RGB(2,1,0) 44. src\_data = src\_data.astype(np.uint8) 45. # 将图片最低位数置 46. tmp\_data = (src\_data//2)\*2 47. # 二值化加密信息 48. bit\_cr = str2bit(copyright) 49. # flatten数据 50. tmp\_a = tmp\_data.flatten().astype(np.uint8) 51. **if** len(tmp\_a)<len(bit\_cr): 52. **print**("加密信息过长，请缩减加密信息") 53. **return** 54. # LSB加密 55. **for** i **in** range(0,len(tmp\_a)): 56. **if** i==len(bit\_cr): 57. **break** 58. tmp\_a[i] = tmp\_a[i] + bit\_cr[i] 59. res\_data = tmp\_a.reshape(tmp\_data.shape) 60. cv2.imwrite(save\_path,res\_data[:,:,[2,1,0]]) # RGB(0,1,2)转为BGR(2,1,0) 62. **def** uncode(img\_path): 63. # 读取图片 64. res\_data = cv2.imread(img\_path)[:,:,[2,1,0]] # BGR(0,1,2)转为RGB(2,1,0) 65. # flatten数据 66. tmp\_a = res\_data.flatten() 67. # 获取LSB 68. bit\_cr = np.zeros(len(tmp\_a)).astype(np.uint8) 69. **for** i **in** range(0,len(tmp\_a)): 70. bit\_cr[i] = tmp\_a[i]%2 71. **return** bit2str(bit\_cr)  74. st = time.time() 75. copyright = "CQUWATERMASKEXP" 76. src\_path = "./lab2\_src.jpg" 77. res\_path = "./lab2\_res.png" 78. **print**("嵌入信息: %s"%(copyright)) 79. encode(copyright,src\_path,res\_path) 80. et = time.time() 81. **print**("加密耗时: %.3fs"%((et-st)/60)) 83. st = time.time() 84. copyright = uncode(res\_path) 85. **print**("解密结果: %s"%(copyright)) 86. et = time.time() 87. **print**("解密耗时: %.3fs"%((et-st)/60)) | | | | | | |
| 五、实验过程中遇到的问题及解决情况  1. cv2读取数据，plt绘制颜色变色问题  原始图片，如图 1所示，但是通过cv2读取图片后，使用matplotlib中的pyplot绘制图片如图 2所示，经查阅资料，发现cv2读取的图片的格式为BGR图片，但是在绘制时，需要转为RGB图片，才能正常显示。    图 1 原始图片    图 2 BGR图片绘制  2. 关于cv2保存数据与读取数据不一致问题  需要存入的数据如图 3所示，存入“lab2\_res.jpg”图片后，再次读取该路径的图片，结果如图 4所示。经过查阅，cv2在存储jpg文件时会存在压缩的情况，故导致存入和读取的数据不一致。存入“lab2\_res.png”文件，问题得到解决。    图 3 待存储数据    图 4 存入后再次读取的数据 | | | | | | |
| 六、实验结果及分析和（或）源程序调试过程  1、原始图片和嵌入信息后的图片分别如下图所示：    图 5 原始图片    图 6 嵌入信息后的图片  2、嵌入信息提取结果如图 7所示，可以看到解密的结果和嵌入的信息一致，LSB算法合理。    图 7 实验运行结果  3、完整的保密通信方案设计：   1. 利用DH密钥交换协议，让发送方和接收方确定对称密钥。将嵌入的信息结合数字签名和时间戳信息，再利用密钥进行加密，再将加密结果通过LSB算法嵌入到图片中。 2. 发送方和接收方共享同一个散列函数MD5，并利用基于事件同步的口令认证技术。发送方对步骤(1)中的图片数据，进行散列，将散列结果作为口令数据，与图片数据一并发送给接收方。 3. 接收方收到信息后，先根据自己的散列函数MD5，对图片数据进行散列，将散列结果与口令数据进行对比，如果一致，说明源自可信的发送方，此时接收数据。否则，不予置理。 4. 接受数据后，提取出图片数据中的水印信息。并利用密钥进行解密，解密后验证数字签名和时间戳，如果该时间戳超过发送方约定的时效，则该数据有误。否则，得到的嵌入信息即可得到完整的该隐藏信息。   4、参考资料   1. [图像水印LSB算法讲解及python实现 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/89765000) 2. [DH（密钥交换协议/算法(Diffie-Hellman)简称）\_百度百科 (baidu.com)](https://baike.baidu.com/item/DH/4895102) 3. [信息安全技术之数字签名\_攻城狮少年的博客-CSDN博客\_信息安全技术数字签名](https://blog.csdn.net/soldier_d/article/details/119172657) 4. [【网络信息安全】鉴别和密钥分配协议\_萌宅鹿同学的博客-CSDN博客\_密钥分配协议](https://blog.csdn.net/weixin_43734095/article/details/105195264) | | | | | | |