**《信息安全概论》实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **姓名** | | **谭星** | | **年级** | | **2019级** |
| **学号** | | **20191584** | | **专业、班级** | | **计算机科学与技术卓越01班** |
| **实验名称** | **信息隐藏实验** | | | | | |
| **实验时间** | **2022.4.19** | | **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）： | | | | | | |
| 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   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。 | | | | | | |
| 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。因此，LSB加解密步骤可分解为：   1. 加密：  * 读入图片 * 准备待隐藏的信息，将其转换为二进制 * 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  1. 解密：  * 预知隐藏信息量（等同于key） * 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. 保密通信方案：  * 在图片像素最低位保存数据，在次低位保存数据长度 * 基于消息鉴别码来保证信息来源可靠且未被篡改 * 通过DES方法来加密信息，一定程度上保证了即使从图片像素最低位提取出数据，也难以破解密文。 | | | | | | |
| 四、实验过程或算法   1. 首先将待加密的明文字符串转换为二进制数据，为了后续解密的方便，每个字符都需要转换为8位二进制数据。 2. # 将字符串转换成二进制,不够8位补齐8位 3. def StringtoBin(data): 4. bin\_data = '' 5. for c in data: 6. temp = bin(ord(c)).replace('0b', '') 7. temp = temp.rjust(8, '0') 8. bin\_data = bin\_data + temp 9. return bin\_data 10. 加密过程中，遍历并替换所有像素点的最低bit位。由于最低bit位在十进制中为0或1，所以可先计算十进制像素值模2的结果来获得最低bit位，再删除该bit位并增添新的数据位。在加密后保存含密文的图片，并返回数据长度。具体实现如下： 11. def encode(path, data): 12. data = StringtoBin(data) 13. print(data) 14. # 加载图片，并读取其高度和宽度 15. image = Image.open(path) 16. height = image.size[0]  # 高度 17. width = image.size[1]  # 宽度 18. count = 0 19. # 密文数据长度须小于图片大小 20. if width \* height \* 3 < len(data): 21. print("数据过长，无法加密！") 22. return 23. image = np.array(image) 24. # 遍历该三通道图片的所有像素点 25. for h in range(height): 26. if count == len(data): 27. break 28. for w in range(width): 29. if count == len(data): 30. break 31. for c in range(3): 32. channel = image[h][w][c] 33. image[h][w][c] = channel - channel % 2 + eval(data[count]) 34. count = count + 1 35. if count == len(data): 36. break 37. image = Image.fromarray(image) 38. image.save('encode.png') 39. return count 40. 解密过程中，由于已知隐藏信息量，所以依次读取相应长度的最低bit位，然后每8位二进制数据翻译一个字符即可获取明文。 41. def decode(path, key): 42. data\_bit = "" 43. # 加载含密文的图片 44. image = Image.open(path) 45. height = image.size[0] 46. width = image.size[1] 47. image = np.array(image) 48. count = 0 49. # 遍历图片所有像素点 50. for h in range(height): 51. if count == key: 52. break 53. for w in range(width): 54. if count == key: 55. break 56. for c in range(3): 57. # 依次读取像素的最低bit位，构成的数据长度等于密文长度时停止 58. data\_bit = data\_bit + str(image[h][w][c] % 2) 59. count = count + 1 60. if count == key: 61. break 62. data = "" 63. idx = 0 64. # 每8位构成一个字符 65. while(True): 66. if idx == len(data\_bit): 67. break 68. a = chr(int(data\_bit[idx:idx+8])) 69. data = data + chr(int(data\_bit[idx:idx+8], 2)) 70. idx = idx + 8 71. print(data) 72. 保密通信方案设计 73. 发送方 74. 根据事先设定的密钥K1对明文信息进行DES加密，从而得到初始密文M，并计算数据长度L1。 75. 使用基于单向Hash函数和对称密钥体制的MAC鉴别方法，先将M经过单向Hash函数计算得到MD，再根据密钥K2计算MD的MAC值，并将MAC拼接在M后面，并计算MAC长度L2。其中在使用F(M, K2)计算消息散列值时，可在填充M数据位后对其进行分组，然后结合各分组和上一分组的结果计算，最后一个分组的结果即为整个消息的散列值。 76. 设图像的像素最低位为第1位，则将密文信息的二进制序列依次存入各像素的第1位，将L1和L2分别扩展至64位，存入图像各像素的第2位，至此完成信息的嵌入。发送的数据格式如下图：      1. 接收方 2. 读取各像素的次低位，分别获取64位的密文长度L1和64位的MAC长度L2。 3. 读取各像素的最低位，分别获取L1位的密文M'和L2位的MAC值。 4. 根据密钥K2计算M'对应的MAC值，若MAC'和提取出的MAC相等则说明信息来源正确且未被修改。 5. 最后将M'部分通过DES解密得到明文信息。   整个加密通信方案流程如下图所示： | | | | | | |
| 五、实验过程中遇到的问题及解决情况  在加密通信方案设计中，联想到了计算机网络的网络分层模型，每一层分别对数据进行封装。于是沿用此思想，第一层首先将明文进行加密，第二层使用第一层的结果来计算校验码并实现数据的拼接，第三层将所得的信息数据封装进图片中，实现加密。解密过程同理，从上到下依次解密。 | | | | | | |
| 六、实验结果及分析和（或）源程序调试过程   1. 加密前（左）和加密后（右）的图片如下，只靠肉眼无法分辨出二者的区别，证实了LSB算法的可行性。      1. 解密后的控制台输出如下所示，可知解密后的文本与密文相同，算法实现正确。   C:\Users\PH\AppData\Roaming\Tencent\Users\1447540812\QQ\WinTemp\RichOle\EUG1J7{A13_`PBX9TZ%G(B1.png | | | | | | |