**《网络空间安全概论》实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **姓名** | |  | | **年级** | | **2022** |
| **学号** | |  | | **专业、班级** | | **计算机科学与技术（卓越）01** |
| **实验名称** | **实验一加密解密算法** | | | | | |
| **实验时间** | **24/3/39** | | **实验地点** | | **DS3** | |
| **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  掌握频度分析法原理和Feistel加解密原理。 | | | | | | |
| 二、实验项目内容  1.使用频度分析法解密文本，并写出替换表  2.编程实现Feistel加密/解密 | | | | | | |
| 三、实验设计  3.1 频度分析法原理  频度分析法是一种经典的密码分析方法，基于自然语言中字母出现的频率分布特点。以英语为例，字母 'E'、'T'、'A' 等出现频率较高，通过对密文中字母频率的统计，并与标准频率对比，可以猜测加密替换关系，从而还原明文。  该实验采用统计密文字母频率后，尝试将高频字母与常见英文高频字母进行替换，逐步推理出密文的替换关系，实现解密。  3.2 Feistel网络结构原理  Feistel结构是一种对称结构，被广泛用于构建分组密码算法（如DES）。该结构将明文划分为左右两部分，每轮通过一个F函数对一部分进行加密，并与另一部分异或交叉，达到加密效果。  本实验中实现的简化Feistel算法包括以下关键模块：   * 初始置换（IP）与逆初始置换（Inv IP） * 扩展函数（E扩展） * S盒替换（用于非线性变换） * P置换（增加混淆） * 轮密钥生成（Key Schedule） * 最终组合并逆置换恢复密文或明文   所有加密操作均以64位块处理，填充机制为高位补0，使用固定密钥进行加解密测试。 | | | | | | |
| 四、实验过程或算法  4.1频度分析法   1. def frequency\_analysis(ciphertext): 2. *# 统计字母频率* 3. ciphertext\_no\_spaces = ciphertext.replace(" ", "")  *# 去掉空格* 4. letter\_counts = Counter(ciphertext\_no\_spaces) 5. total\_letters = sum(letter\_counts.values()) 6. *# 计算字母频率* 7. letter\_frequencies = {letter: count / total\_letters for letter, count in letter\_counts.items()} 8. return letter\_frequencies 9. if \_\_name\_\_ == "\_\_main\_\_": 10. ciphertext = "UZ QSO VUOHXMOPV GPOZPEVSG ZWSZ OPFPESX UDBMETSX AIZ VUEPHZ HMDZSHZO WSFP APPD TSVP QUZW YMXUZUHSX EPYEPOPDZSZUFPO MB ZWP FUPZ HMDJ UD TMOHMQ" 11. letter\_frequencies = frequency\_analysis(ciphertext) 12. *# 按照频率降序排序* 13. sorted\_frequencies = sorted(letter\_frequencies.items(), key=lambda x: x[1], reverse=True) 14. *# 按相同频率分组* 15. grouped\_frequencies = {key: [letter for letter, \_ in group] for key, group in 16. groupby(sorted\_frequencies, key=lambda x: x[1])} 17. print("Letter Frequencies:") 18. for freq, letters in grouped\_frequencies.items(): 19. print(f"{freq:.4f}: {' '.join(letters)}")  * 首先对单词、字母进行频度分析，得到以下结果：      * 由于P频率最高，Z次高，所以先尝试替换P和Z, P->E,Z->T  1. ciphertext = ciphertext.replace("P", "e") 2. ciphertext = ciphertext.replace("Z", "t") 3. *# Ut QSO VUOHXMOeV GeOteEVSG tWSt OeFeESX UDBMETSX AIt VUEeHt HMDtSHtO WSFe AeeD TSVe QUtW YMXUtUHSX EeYEeOeDtStUFeO MB tWe FUet HMDJ UD TMOHMQ*  * 发现tWe->the  1. ciphertext = ciphertext.replace("W","h") 2. *# Ut QSO VUOHXMOeV GeOteEVSG thSt OeFeESX UDBMETSX AIt VUEeHt HMDtSHtO hSFe AeeD TSVe QUth YMXUtUHSX EeYEeOeDtStUFeO MB the FUet HMDJ UD TMOHMQ*  * 发现UZ词频比较高，参考双字母表，发现it可能性比较大，尝试将U->i  1. ciphertext = ciphertext.replace("U", "i") 2. *# it QSO ViOHXMOeV GeOteEVSG thSt OeFeESX iDBMETSX AIt ViEeHt HMDtSHtO hSFe AeeD TSVe Qith YMXitiHSX EeYEeOeDtStiFeO MB the Fiet HMDJ iD TMOHMQ*  * thSt->that  1. ciphertext = ciphertext.replace("S","a") 2. *# it QaO ViOHXMOeV GeOteEVaG that OeFeEaX iDBMETaX AIt ViEeHt HMDtaHtO haFe AeeD TaVe Qith YMXitiHaX EeYEeOeDtatiFeO MB the Fiet HMDJ iD TMOHMQ*  * Qith->with  1. ciphertext = ciphertext.replace("Q","w") 2. *#it waO ViOHXMOeV GeOteEVaG that OeFeEaX iDBMETaX AIt ViEeHt HMDtaHtO haFe AeeD TaVe with YMXitiHaX EeYEeOeDtatiFeO MB the Fiet HMDJ iD TMOHMw*  * waO->was  1. ciphertext = ciphertext.replace("O","s") 2. *#it was VisHXMseV GesteEVaG that seFeEaX iDBMETaX AIt ViEeHt HMDtaHts haFe AeeD TaVe with YMXitiHaX EeYEeseDtatiFes MB the Fiet HMDJ iD TMsHMw*  * haFe->have  1. ciphertext=ciphertext.replace("F","v") 2. *# it was VisHXMseV GesteEVaG that seveEaX iDBMETaX AIt ViEeHt HMDtaHts have AeeD TaVe with YMXitiHaX EeYEeseDtatives MB the viet HMDJ iD TMsHMw*  * seveEaX->several  1. ciphertext=ciphertext.replace("E","r") 2. ciphertext=ciphertext.replace("X","l") 3. *#it was VisHlMseV GesterVaG that several iDBMrTal AIt VireHt HMDtaHts have AeeD TaVe with YMlitiHal reYreseDtatives MB the viet HMDJ iD TMsHMw*  * 由iD猜测可能为in或is，若D->n，同时可以满足reYreseDtatives->representatives,即Y->p，反之s则不行  1. ciphertext = ciphertext.replace("D","n") 2. ciphertext = ciphertext.replace("Y","p") 3. *#it was VisHlMseV GesterVaG that several inBMrTal AIt VireHt HMntaHts have Aeen TaVe with pMlitiHal representatives MB the viet HMnJ in TMsHMw*  * pMlitiHal->political，ViscloseV->disclosed  1. ciphertext = ciphertext.replace("M","o") 2. ciphertext = ciphertext.replace("H","c") 3. ciphertext = ciphertext.replace("V","d") 4. *# it was disclosed GesterdaG that several inBorTal AIt direct contacts have Aeen Tade with political representatives oB the viet conJ in Toscow*  * GesterdaG->yesterday，Tade->made  1. ciphertext = ciphertext.replace("G","y") 2. ciphertext = ciphertext.replace("T","m") 3. *#it was disclosed yesterday that several inBormal AIt direct contacts have Aeen made with political representatives oB the viet conJ in moscow*  * Aeen->been，bIt->but  1. ciphertext = ciphertext.replace("A","b") 2. ciphertext = ciphertext.replace("I","u") 3. *#it was disclosed yesterday that several inBormal but direct contacts have been made with political representatives oB the viet conJ in moscow*  * oB->on，conJ->cong  1. ciphertext = ciphertext.replace("B","n") 2. ciphertext = ciphertext.replace("J","g")   最终得到明文： it was disclosed yesterday that several innormal but direct contacts have been made with political representatives on the viet cong in moscow  4.2 Feistel加密解密   1. def main(): 2. import sys 3. *# 置换表和S盒定义* 4. ip = [ 5. 58, 50, 42, 34, 26, 18, 10, 2, 6. 60, 52, 44, 36, 28, 20, 12, 4, 7. 62, 54, 46, 38, 30, 22, 14, 6, 8. 64, 56, 48, 40, 32, 24, 16, 8, 9. 57, 49, 41, 33, 25, 17, 9, 1, 10. 59, 51, 43, 35, 27, 19, 11, 3, 11. 61, 53, 45, 37, 29, 21, 13, 5, 12. 63, 55, 47, 39, 31, 23, 15, 7 13. ] 14. inv\_ip = [ 15. 40, 8, 48, 16, 56, 24, 64, 32, 16. 39, 7, 47, 15, 55, 23, 63, 31, 17. 38, 6, 46, 14, 54, 22, 62, 30, 18. 37, 5, 45, 13, 53, 21, 61, 29, 19. 36, 4, 44, 12, 52, 20, 60, 28, 20. 35, 3, 43, 11, 51, 19, 59, 27, 21. 34, 2, 42, 10, 50, 18, 58, 26, 22. 33, 1, 41, 9, 49, 17, 57, 25 23. ] 24. e = [ 25. 32, 5, 26. 4, 9, 27. 8, 13, 28. 12, 17, 29. 16, 21, 30. 20, 25, 31. 24, 29, 32. 28, 1 33. ] 34. pc1 = [ 35. 57, 49, 41, 33, 25, 17, 9, 36. 1, 58, 50, 42, 34, 26, 18, 37. 10, 2, 59, 51, 43, 35, 27, 38. 19, 11, 3, 60, 52, 44, 36, 39. 63, 55, 47, 39, 31, 23, 15, 40. 7, 62, 54, 46, 38, 30, 22, 41. 14, 6, 61, 53, 45, 37, 29, 42. 21, 13, 5, 28, 20, 12, 4 43. ] 44. pc2 = [ 45. 14, 17, 11, 24, 1, 5, 46. 3, 28, 15, 6, 21, 10, 47. 23, 19, 12, 4, 26, 8, 48. 16, 7, 27, 20, 13, 2, 49. 41, 52, 31, 37, 47, 55, 50. 30, 40, 51, 45, 33, 48, 51. 44, 49, 39, 56, 34, 53, 52. 46, 42, 50, 36, 29, 32 53. ] 54. sbox = [ 55. [ 56. [14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7], 57. [0,15,7,4,14,2,13,1,10,6,12,11,9,5,3,8], 58. [4,1,14,8,13,6,2,11,15,12,9,7,3,10,5,0], 59. [15,12,8,2,4,9,1,7,5,11,3,14,10,0,6,13] 60. ], 61. [ 62. [15,1,8,14,6,11,3,4,9,7,2,13,12,0,5,10], 63. [3,13,4,7,15,2,8,14,12,0,1,10,6,9,11,5], 64. [0,14,7,11,10,4,13,1,5,8,12,6,9,3,2,15], 65. [13,8,10,1,3,15,4,2,11,6,7,12,0,5,14,9] 66. ], 67. [ 68. [10,0,9,14,6,3,15,5,1,13,12,7,11,4,2,8], 69. [13,7,0,9,3,4,6,10,2,8,5,14,12,11,5,1], 70. [13,6,4,9,8,15,3,0,11,1,2,12,5,10,14,7], 71. [1,10,13,0,6,9,8,7,4,15,14,3,11,5,2,12] 72. ], 73. [ 74. [7,13,14,3,0,6,9,10,1,2,8,5,11,12,4,15], 75. [13,8,11,5,6,15,0,3,4,7,2,12,1,10,14,9], 76. [10,6,9,0,12,11,7,13,15,1,3,14,5,2,8,4], 77. [3,15,0,6,10,1,13,8,9,4,5,11,12,7,2,14] 78. ], 79. [ 80. [2,12,4,1,7,10,11,6,8,5,3,15,13,0,14,9], 81. [14,11,2,12,4,7,13,1,5,0,15,10,3,9,8,6], 82. [4,2,1,11,10,13,7,8,15,9,12,5,6,3,0,14], 83. [11,8,12,7,1,14,2,13,6,15,0,9,10,4,5,3] 84. ], 85. [ 86. [12,1,10,15,9,2,6,8,0,13,3,4,14,7,5,11], 87. [10,15,4,2,7,12,9,5,6,1,13,14,0,11,3,8], 88. [9,14,15,5,2,8,12,3,7,0,4,10,1,13,11,6], 89. [4,3,2,12,9,5,15,10,11,14,1,7,6,0,8,13] 90. ], 91. [ 92. [4,11,2,14,15,0,8,13,3,12,9,7,5,10,6,1], 93. [13,0,11,7,4,9,1,10,14,3,5,12,2,15,8,6], 94. [1,4,11,13,12,3,7,4,10,15,6,8,0,5,9,12], 95. [6,11,13,8,1,4,10,7,9,5,0,15,14,2,3,12] 96. ], 97. [ 98. [13,2,8,4,6,15,11,1,10,9,3,14,5,0,12,7], 99. [1,15,13,8,10,3,7,4,12,5,6,11,0,14,9,2], 100. [1,11,4,1,9,12,14,2,0,6,10,13,15,3,5,8], 101. [2,1,14,7,4,10,8,13,15,12,9,0,3,5,6,11] 102. ] 103. ] 104. p = [ 105. 16,7,20,21,29,12,28,17, 106. 1,15,23,26,5,18,31,10, 107. 2,8,24,14,32,27,3,9, 108. 19,13,30,6,22,11,4,25 109. ] 110. *# 工具函数* 111. def to\_int(s): 112. return int(s, 2) 113. def to\_bin\_str(x, bits): 114. return format(x, f'0{bits}b')[-bits:] 115. def char\_to\_bin(ch): 116. return format(ord(ch), '08b') 117. def string\_to\_bin(s): 118. return ''.join(char\_to\_bin(c) for c in s) 119. def bin\_to\_char(b): 120. return chr(int(b, 2)) 121. def bin\_to\_str(b): 122. return ''.join(bin\_to\_char(b[i:i+8]) for i in range(0, len(b), 8)) 123. *# 置换函数* 124. def permute(bits, table): 125. return ''.join(bits[i-1] for i in table) 126. *# 密钥生成* 127. def key\_schedule(key\_bits): 128. key\_pc1 = permute(key\_bits, pc1) 129. c = key\_pc1[:28] 130. d = key\_pc1[28:] 131. keys = [] 132. shifts = [1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1] 133. for i in range(16): 134. shift = shifts[i] 135. c = c[shift:] + c[:shift] 136. d = d[shift:] + d[:shift] 137. key\_pc2 = permute(c + d, pc2) 138. keys.append(key\_pc2) 139. return keys 140. *# F函数* 141. def f\_function(r, key): 142. *# 扩展置换* 143. r\_expanded = permute(r, e) 144. *# 异或* 145. xor\_result = format(int(r\_expanded, 2) ^ int(key, 2), '048b') 146. *# S盒替换* 147. s\_out = [] 148. for i in range(8): 149. block = xor\_result[i\*6:(i+1)\*6] 150. row = int(block[0] + block[5], 2) 151. col = int(block[1:5], 2) 152. s\_val = sbox[i][row][col] 153. s\_out.append(to\_bin\_str(s\_val, 4)) 154. s\_out = ''.join(s\_out) 155. *# P置换* 156. return permute(s\_out, p) 157. *# DES加密/解密* 158. def des\_process(block, keys, is\_encrypt): 159. block = permute(block, ip) 160. l, r = block[:32], block[32:] 161. for i in range(16): 162. key = keys[i] if is\_encrypt else keys[15-i] 163. new\_r = format(int(l, 2) ^ int(f\_function(r, key), 2), '032b') 164. l, r = r, new\_r 165. combined = r + l 166. return permute(combined, inv\_ip) 167. *# 主函数* 168. def main\_des(): 169. *# 示例明文* 170. plaintext = "CQUINFORMATIONSECURITYEXP" 171. print(f"Plaintext: {plaintext}\n\nEncoding...\n") 172. *# 密钥处理* 173. key\_str = "Arknight" 174. key\_bits = string\_to\_bin(key\_str) 175. keys = key\_schedule(key\_bits) 176. *# 分组处理* 177. bin\_plain = string\_to\_bin(plaintext) 178. padding = (64 - (len(bin\_plain) % 64)) % 64 179. bin\_plain = '0' \* padding + bin\_plain  *# 高位填充* 180. groups = [bin\_plain[i:i+64] for i in range(0, len(bin\_plain), 64)] 181. *# 加密* 182. cipher\_groups = [] 183. for i, group in enumerate(groups): 184. cipher = des\_process(group, keys, True) 185. cipher\_groups.append(cipher) 186. print(f"Group {i}:\nPlain: {group}\nCipher: {cipher}") 187. *# 解密* 188. print("\nDecoding...\n") 189. decrypted\_groups = [] 190. for i, cipher in enumerate(cipher\_groups): 191. decrypted = des\_process(cipher, keys, False) 192. decrypted\_groups.append(decrypted) 193. print(f"Group {i}:\nCipher: {cipher}\nPlain: {decrypted}") 194. *# 合并结果* 195. full\_decrypted = ''.join(decrypted\_groups) 196. *# 去除填充* 197. full\_decrypted = full\_decrypted[padding:] 198. *# 转换回字符串* 199. result = bin\_to\_str(full\_decrypted) 200. print(f"\nDecrypted text: {result}") 201. main\_des() 202. if \_\_name\_\_ == "\_\_main\_\_": 203. main() | | | | | | |
| 五、实验过程中遇到的问题及解决情况  **5.1频度分析过程中替换误判问题**  在初期替换高频字符时，由于某些字母频率较为接近，出现了替换不准确的情况，导致后续文本逻辑不通。  **解决办法**：结合上下文和常见英文词组进行判断，如发现 “tWe” 可为 “the”，由此反推出 “W” 应替换为 “h”，不断修正替换表。  **5.2Feistel加密填充处理问题**  实验初期未处理明文长度非64位倍数的情况，导致程序运行报错或结果错误。  **解决办法**：增加自动高位填充功能，确保每组加密块为64位，从而程序可以正确执行。  **5.3 S盒索引错误问题**  初期S盒索引计算中的行列提取顺序搞错，导致输出不正确。  **解决办法**：重新核查Feistel结构中S盒的查表逻辑，确认行由第1、6位组成，列由中间4位组成，并进行修正。  **5.4密钥位数不足**  使用密钥 “Arknight” 时，因其为8字节不足64位比特，导致生成的子密钥不一致。  **解决办法**：将字符转换为8位二进制后拼接，确保生成64位初始密钥。 | | | | | | |
| 六、实验结果及分析和（或）源程序调试过程  6.1频度分析法  最终得到明文： it was disclosed yesterday that several innormal but direct contacts have been made with political representatives on the viet cong in moscow  得到字符替换表位：   |  |  |  |  | | --- | --- | --- | --- | | 原字符 | 替换字符 | 原字符 | 替换字符 | | P | E | Z | T | | U | I | W | H | | Q | W | S | A | | O | S | F | V | | E | R | X | L | | Y | P | D | N | | A | B | I | U | | M | O | H | C | | T | M | B | F | | V | D | G | Y | | J | G |  |  |   6.2 Feistel加密解密  最终程序运行结果：   1. Plaintext: CQUINFORMATIONSECURITYEXP 2. Encoding... 3. Group 0: 4. Plain: 0000000000000000000000000000000000000000000000000000000001000011 5. Cipher: 0100110110010000011101010110010010110011101100110100010111011010 6. Group 1: 7. Plain: 0101000101010101010010010100111001000110010011110101001001001101 8. Cipher: 1110111111111010111100111110000011101010001011011110010011010111 9. Group 2: 10. Plain: 0100000101010100010010010100111101001110010100110100010101000011 11. Cipher: 1100110001111000111110111010110100011110001111011100111111011010 12. Group 3: 13. Plain: 0101010101010010010010010101010001011001010001010101100001010000 14. Cipher: 1110010100010001111110111100111110000101000011011110000111111001 15. Decoding... 16. Group 0: 17. Cipher: 0100110110010000011101010110010010110011101100110100010111011010 18. Plain: 0000000000000000000000000000000000000000000000000000000001000011 19. Group 1: 20. Cipher: 1110111111111010111100111110000011101010001011011110010011010111 21. Plain: 0101000101010101010010010100111001000110010011110101001001001101 22. Group 2: 23. Cipher: 1100110001111000111110111010110100011110001111011100111111011010 24. Plain: 0100000101010100010010010100111101001110010100110100010101000011 25. Group 3: 26. Cipher: 1110010100010001111110111100111110000101000011011110000111111001 27. Plain: 0101010101010010010010010101010001011001010001010101100001010000 28. Decrypted text: CQUINFORMATIONSECURITYEXP   加密解密得到的文本一致，由此可知 Feistel算法的正确性 | | | | | | |