**Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования** **«Новгородский государственный университет имени Ярослава Мудрого»**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра ИТиС

Лабораторная работа №3

По дисциплине: Защита информации

Отчет

Выполнил: студент гр. 9091

\_\_\_\_\_\_\_\_\_\_\_Чалый.С.М

Проверил: Преподаватель

\_\_\_\_\_\_\_\_\_\_\_Жгун.Т.В

Великий Новгород

2022

1. **Формулировка цели и задач**

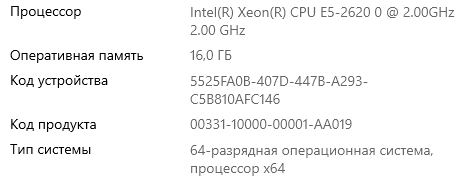
Целью данной работы является реализация работы сети Файстеля для блока 16 бит (шифрование и расшифрование). Число раундов 8. Исходный текст берется из варианта. В качестве ключа шифрования выступает пароль из работы 1. Раундовые ключи являются последовательными отрезками ключа шифрования.

Программа должна выводить промежуточные результаты каждого шага.

Построить гистограмму для зашифрованных текстов.

1. **Текст программы**

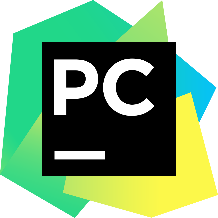
**2.1 Тестирование и запуск.**



Необходимые средства применяемые мною при выполнении данной работы:

PyCharm

* Version: 2021.3.2
* Build: 213.6777.50
* 31 January 2022
* Community
* Python 3.10



"**2.1 Тестирование и запуск**".

# -\*- coding: utf8 -\*-  
import module  
import matplotlib.pyplot as plt  
  
key = "Пустой колос голову кверху носит"  
key = key.replace(" ", "")  
key = key.lower()  
text = "Есть дети, есть отцы, деды а собственно мужчины героинь будто бы запропали куда то.Видимо, это миф, когда говорят, что мужчины вполне могут обойтись без женщин, зато вот женщины без сильного пола не могут совсем, никак.Могут.По крайней мере, с недавнего времени.Мир, описанный в этой книге, движим женщиной. Женщины здесь тихие подвижники не быта или не только быта, а бытия.Не сказать, что мужчины вовсе ни при чем: их, в общем то, еще ждут, по ним даже скучают но они все равно где то на периферии женского зрения и сознания.На мужчин будто бы махнули рукой: что взять с них нечего.Мужчина никакой не центр женского мироздания как зачастую кажется самим мужчинам. В лучшем случае это нелишний атрибут. Только непонятно чего. Никак не счастья.Читая эту книжку, очень хотелось понять, как эпоха отражается в женских зрачках, что за вкус у нее, что за цвет. Какие детали эпохи наглядны, каких нет вовсе.И в первую очередь выяснилось, говорю, что тут нет мужчины героя. Во всех смыслах слова герой.Нету его.Есть Аркаша в повести «Вариант нормы» Анны Андроновой но он скорей сын своей матери, зато муж никакой, да и будущая жена тоже воспринимает его как большого непричесанного ребенка.Муж время от времени бросает реплики в другой повести Андроновой Я не зайчик, но он во всей этой круговерти событий все равно эпизодическое лицо, он почти случаен в семье за все время повествования его с трудом хватает на то, чтоб приделать уши маске зайца. Дальше он опять засыпает и спит. Все время спит.Гриша в Кукуше Майи Кучерской тоже почти сын героини любом случае не муж точно. Муж, правда, у нее тоже есть, но он в буквальном смысле убегает. Не смог перенести ожидания жена застряла в лифте, и пришлось полчаса ее ждать. Разве это возможно? Сбежал."  
# text = text.replace(" ", "")  
text = text.replace(":", "")  
text = text.replace(".", "")  
text = text.replace(",", "")  
text = text.replace("(", "")  
text = text.replace(")", "")  
text = text.replace("?", "")  
text = text.replace(";", "")  
text = text.lower()  
  
# text = "абра кадабра бадабра кадабра."  
encryptedMessage = []  
encryptedMessageAlp = []  
decryptedMessage = []  
Rzero = []  
  
  
# translate key into bin:  
def keyToBin():  
 keyStr = ""  
 for ch in key:  
 keyStr += module.getSymbol(ch)  
 return keyStr  
  
  
Name = []  
Values = []  
  
  
def makeGraph(str):  
 Name.clear()  
 Values.clear()  
 fig, ax = plt.subplots()  
 for i in str:  
 counter = 0  
 for g in str:  
 if i == g:  
 counter += 1  
 Name.append(i)  
 Values.append(counter)  
  
 plt.bar(Name, Values)  
 plt.show()  
  
  
def encrypt(txt):  
 # 8 rounds  
 if len(txt) % 2 != 0:  
 txt = "0" + txt  
 middle = int(len(txt) / 2)  
 L = []  
 R = []  
 L.append(txt[0:middle])  
 R.append(txt[middle:len(txt)])  
 Rzero.append(R[0])  
  
 counter = 0  
 while counter < 8:  
 # one round description  
 # 1.  
 L.append(R[counter])  
 # 2. [XoR R[current step], Keys[current step]]  
 resWithXoRKey = module.getXorResult(R[counter], Keys[counter])  
 # 3.[XoR res 2step, L[current step]]  
 resWithXoRLeft = module.getXorResult(resWithXoRKey, L[counter])  
 # 4. res 3 step << 1  
 R.append(module.bitwiseShift(resWithXoRLeft))  
  
 # print(L[counter] + R[counter])  
  
 # drawing gist  
 # makeGraph(transformBinIntoText(L[counter] + R[counter]))  
 # for ch in str:  
 counter += 1  
 encryptedMessage.append(L[len(L) - 1] + R[len(R) - 1])  
  
  
def transformBinIntoText(value):  
 endString = ""  
 timeLine = ""  
 for ch in value:  
 timeLine += ch  
 if len(timeLine) == 6:  
 endString += module.getAlp(timeLine)  
 timeLine = ""  
 return endString  
  
  
def decrypt(value, RzeroCount):  
 middle = int(len(value) / 2)  
 L = []  
 R = []  
 L.append(value[0: middle])  
 R.append(value[middle: len(value)])  
 counter = 0  
 keyInd = len(Keys) - 1  
  
 while counter < 8:  
 if counter == 7:  
 R.append(Rzero[RzeroCount])  
 else:  
 # 1  
 R.append(L[counter])  
  
 # 2  
 resBitwise = module.bitwiseShiftL(R[counter])  
 # 3  
 resWithXorKey = module.getXorResult(R[counter + 1], Keys[keyInd])  
 # 4  
 L.append(module.getXorResult(resWithXorKey, resBitwise))  
 keyInd -= 1  
 counter += 1  
 decryptedMessage.append(L[len(L) - 1] + R[len(R) - 1])  
  
  
# format key values  
keyBinArr = keyToBin()  
len(keyBinArr)  
Keys = []  
Keys.append(keyBinArr[0:16])  
Keys.append(keyBinArr[16:32])  
Keys.append(keyBinArr[32:48])  
Keys.append(keyBinArr[48:64])  
Keys.append(keyBinArr[64:80])  
Keys.append(keyBinArr[80:96])  
Keys.append(keyBinArr[96:112])  
Keys.append(keyBinArr[112:128])  
  
# format text  
textDouble = text  
textDouble = textDouble.replace(" ", "")  
textDouble = textDouble.lower()  
makeGraph(textDouble)  
text = text + "."  
binText = []  
timeLine = ""  
for ch in text:  
 if (ch == " " or ch == "."):  
 binText.append(timeLine)  
 timeLine = ""  
 else:  
 timeLine += str(module.getSymbol(ch))  
# encrypting text  
for txt in binText:  
 encrypt(txt)  
# transforming encrypting text  
endString = ""  
for value in encryptedMessage:  
 timeLine = ""  
 newLine = ""  
 if len(value) % 6 != 0:  
 maximum = 0  
 for i in range(0, 100):  
 if (i + len(value)) % 6 == 0:  
 maximum = i  
 break  
 while len(newLine) != maximum:  
 newLine += "0"  
 value = newLine + value  
 for ch in value:  
 timeLine += ch  
 if len(timeLine) == 6:  
 endString += module.getAlp(timeLine)  
 timeLine = ""  
print(endString)  
makeGraph(endString)  
counter = 0  
for value in encryptedMessage:  
 decrypt(value, counter)  
 counter += 1  
  
StartText = []  
for value in decryptedMessage:  
 allString = ""  
 newLine = ""  
 maxCount = len(value) - 1  
 while maxCount >= 0:  
 newLine = value[maxCount] + newLine  
 if len(newLine) == 6:  
 allString += module.getAlp(newLine)  
 newLine = ""  
 maxCount -= 1  
 StartText.append(allString[::-1])  
graphStr = ""  
for value in StartText:  
 print(value, " ", end="")

Symbols = {}  
#################################A\_L\_P###############################  
Symbols["а"] = "000001" #a  
Symbols["б"] = "000010" #б  
Symbols["в"] = "000011" #в  
Symbols["г"] = "000100" #г  
Symbols["д"] = "000101" #д  
Symbols["е"] = "000110" #е  
Symbols["ё"] = "000111" #ё  
Symbols["ж"] = "001000" #ж  
Symbols["з"] = "001001" #з  
Symbols["и"] = "001010" #и  
Symbols["й"] = "001011" #й  
Symbols["к"] = "001100" #к  
Symbols["л"] = "001101" #л  
Symbols["м"] = "001110" #м  
Symbols["н"] = "001111" #н  
Symbols["о"] = "010000" #о  
Symbols["п"] = "010001" #п  
Symbols["р"] = "010010" #р  
Symbols["с"] = "010011" #с  
Symbols["т"] = "010100" #т  
Symbols["у"] = "010101" #у  
Symbols["ф"] = "010110" #ф  
Symbols["х"] = "010111" #х  
Symbols["ц"] = "011000" #ц  
Symbols["ч"] = "011001" #ч  
Symbols["ш"] = "011010" #ш  
Symbols["щ"] = "011011" #щ  
Symbols["ъ"] = "011100" #ъ  
Symbols["ы"] = "011101" #ы  
Symbols["ь"] = "011110" #ь  
Symbols["э"] = "011111" #э  
Symbols["ю"] = "100000" #ю  
Symbols["я"] = "100001" #я

Octs = {}  
Octs["000001"] = 1  
Octs["000010"] = 2  
Octs["000011"] = 3  
Octs["000100"] = 4  
Octs["000101"] = 5  
Octs["000110"] = 6  
Octs["000111"] = 7  
Octs["001000"] = 8  
Octs["001001"] = 9  
Octs["001010"] = 10  
Octs["001011"] = 11  
Octs["001100"] = 12  
Octs["001101"] = 13  
Octs["001110"] = 14  
Octs["001111"] = 15  
Octs["010000"] = 16  
Octs["010001"] = 17  
Octs["010010"] = 18  
Octs["010011"] = 19  
Octs["010100"] = 20  
Octs["010101"] = 21  
Octs["010110"] = 22  
Octs["010111"] = 23  
Octs["011000"] = 24  
Octs["011001"] = 25  
Octs["011010"] = 26  
Octs["011011"] = 27  
Octs["011100"] = 28  
Octs["011101"] = 29  
Octs["011110"] = 30  
Octs["011111"] = 31  
Octs["100000"] = 32  
Octs["100001"] = 33  
#################################O\_C\_T###############################  
def getSymbol(sym):  
 if isinstance(sym, str) == False:  
 sym = str(sym)  
 for key in Symbols:  
 if sym == key:  
 return Symbols[key]  
  
def getAlp(value):  
 returningSym = ""  
 for key in Symbols:  
 if Symbols[key] == value:  
 returningSym = key  
 break  
  
 return returningSym  
  
def getOctFromBinValue(value):  
 if isinstance(value, str) == False:  
 value = str(value)  
 for key in Octs:  
 if key == value:  
 return Octs[key]  
  
def getBinFromOctValue(value):  
 print(value)  
 if isinstance(value, int) == False:  
 value = int(value)  
 for key in Octs:  
 if Octs[key] == value:  
 return key  
  
def getXorResult(arg1, arg2):  
 if len(arg2) > len(arg1):  
 newValue = ""  
 middlePoint = len(arg2) - len(arg1)  
 while len(newValue) != middlePoint:  
 newValue += "0"  
 arg1 = newValue + arg1  
 elif len(arg1) > len(arg2):  
 newValue = ""  
 middlePoint = len(arg1) - len(arg2)  
 while len(newValue) != middlePoint:  
 newValue += "0"  
 arg2 = newValue + arg2  
 return getXor(arg1, arg2)  
  
def getXor(arg1, arg2):  
 res = ""  
 for i in range(0, len(arg1)):  
 if arg1[i] != arg2[i]:  
 res += "1"  
 elif arg1[i] == arg2[i]:  
 res += "0"  
 return res  
  
def bitwiseShift(str, count = 1):  
 swap = str[count : len(str)]  
 swap += str[0 : count]  
 return swap  
def bitwiseShiftL(str, count = 1):  
 swap = str[0 : len(str) - 1]  
 swap = str[len(str) - 1 : len(str)] + swap  
 return swap  
  
  
def getStringAllValues():  
 str = ""  
 for key in Symbols:  
 str += Symbols[key] + "\n"  
 return str

