Федеральное государственное автономное

образовательное учреждение

высшего профессионального образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

институт

Прикладная математика и компьютерная безопасность

кафедра

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ 13**

по Криптографические методы защиты информации

наименование дисциплины

Частотный криптоанализ

тема

Преподаватель В.И. Вайнштейн

подпись**,** дата инициалы, фамилия

Студент КИ17-01, 031722011 К.А. Василенко

номер группы, зачетной книжки подпись**,** дата инициалы, фамилия

Красноярск 2021

# ВВЕДЕНИЕ

Цель работы: получение практических навыков в криптоанализе шифрованного теста на основе частотного анализа. реализовать метод частотного анализа шифротекста на любом языке программирования. Предусмотреть графический интерфейс.

Задачи:

1. реализовать программно метод частотного анализа шифротекста, предусматривая графический интерфейс;
2. провести тесты на работоспособность программы;
3. сделать отчёт о проделанной работе.

**Реализация шифра**

1. **Описание метода**

Частотный анализ, частотный криптоанализ — один из методов криптоанализа, основывающийся на предположении о существовании статистического распределения отдельных символов и их последовательностей как в открытом тексте, так и в шифротексте, которое, с точностью до замены символов, будет сохраняться в процессе шифрования и дешифрования.

Упрощённо, частотный анализ предполагает, что частота появления заданной буквы алфавита в достаточно длинных текстах одна и та же для разных текстов одного языка. При этом в случае моноалфавитного шифрования если в шифротексте будет символ с аналогичной вероятностью появления, то можно предположить, что он и является указанной зашифрованной буквой.

Приблизительные частоты распределения букв уже давно составлены практически для всех языков мира (см. таблицы распределения букв).

Алгоритм для подсчета частоты появления символов алфавита в блоке исходного текста:

На первом шаге вводится символы исходного текста из файла в массив символов. Файл, содержащий исходный текст, должен иметь текстовый формат. Одновременно определяем принадлежность вводимого символа к множеству букв используемого алфавита или цифр. Если введенный символ не является таковым, то исключаем его из рассмотрения.

В дальнейшем для всех букв/цифр просматриваем массив и подсчитываем число появлений каждой из них, а также определяем относительные частоты появления букв по формуле: (частота появления i-го символа/общее количество символов текста)\*100. После чего для криптоанализа и сравним их со среднестатистическими.

Достоверность получаемых в ходе эксперимента относительных частот повышается с увеличением размерности анализируемого блока текста.

Также необходимо принимать во внимание особенности текста (например, литературный или технический), для которых относительные частоты букв могут несколько различаться.

|  |  |  |  |
| --- | --- | --- | --- |
| А | 0.062 | A | 0.081 |
| Б | 0.014 | B | 0.016 |
| В | 0.038 | C | 0.032 |
| Г | 0.013 | D | 0.036 |
| Д | 0.025 | E | 0.123 |
| Е | 0.072 | F | 0.023 |
| Ё | 0.0001 | G | 0.016 |
| Ж | 0.007 | H | 0.051 |
| З | 0.016 | I | 0.071 |
| И | 0.062 | J | 0.001 |
| Й | 0.01 | K | 0.005 |
| К | 0.028 | L | 0.04 |
| Л | 0.035 | M | 0.022 |
| М | 0.026 | N | 0.072 |
| Н | 0.053 | O | 0.079 |
| О | 0.09 | P | 0.023 |
| П | 0.023 | Q | 0.002 |
| Р | 0.04 | R | 0.06 |
| С | 0.045 | S | 0.066 |
| Т | 0.053 | T | 0.096 |
| У | 0.021 | U | 0.031 |
| Ф | 0.002 | V | 0.009 |
| Х | 0.009 | W | 0.02 |
| Ц | 0.004 | X | 0.002 |
| Ч | 0.012 | Y | 0.001 |
| Ш | 0.006 |  |  |
| Щ | 0.003 |  |  |
| Ъ | 0.0004 |  |  |
| Ы | 0.016 |  |  |
| Ь | 0.014 |  |  |
| Э | 0.003 |  |  |
| Ю | 0.006 |  |  |
| Я | 0.018 |  |  |

{\displaystyle c\_{j}=(m\_{j}+k\_{j})\mod {n}}{\displaystyle m\_{j}=(c\_{j}+n-k\_{j})\mod {n}}**Программная реализация**

Программа написана на языке Python с использованием библиотеки PyQt5 и Qt designer для отрисовки графического интерфейса.

1. **Листинг с описанием основного алгоритма программы**

**chastota.py**

# -\*- coding: utf-8 -\*-  
**from** PyQt5 **import** QtWidgets  
**import** matplotlib.pyplot **as** plt  
**import** numpy **as** np  
**from** termcolor **import** colored  
  
**def** oneRegistr(text, dict):  
 dictEng = "abcdefghijklmnopqrstuvwxyz"  
 dictRus = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя"  
 dictEngUp = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"  
 dictRusUp = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ"  
 **if** dict == dictRusUp:  
 **for** i **in** range(len(text)):  
 **for** j **in** range(len(dictRus)):  
 **if** text[i] == dictRus[j]:  
 text = text[:i] + dictRusUp[dictRus.index(text[i])] + text[i + 1:]  
 **break  
 elif** dict == dictEngUp:  
 **for** i **in** range(len(text)):  
 **for** j **in** range(len(dictEng)):  
 **if** text[i] == dictEng[j]:  
 text = text[:i] + dictEngUp[dictEng.index(text[i])] + text[i + 1:]  
 **break** print(text)  
 **return** text  
  
text = "АББВВВ"  
  
**class** IshodnayaBukva:  
 bukva = ""  
 chastota = 0  
 nearbyChastot = [] #буква - разница с этой буквой  
 bukvaInSootv = ""  
  
 **def** \_\_init\_\_(self, bukva):  
 self.bukva = bukva  
 self.chastota = 0  
 self.nearbyChastot = []  
  
 **def** print(self):  
 print(self.bukva, self.chastota)  
 print('{', end=' ')  
 **for** i **in** self.nearbyChastot:  
 print(i.bukva, i.raznitsa, end=' ')  
 print('}')  
  
 **def** sort(self):  
 **for** i **in** range(len(self.nearbyChastot) - 1):  
 **for** j **in** range(len(self.nearbyChastot) - 1):  
 **if** self.nearbyChastot[j].raznitsa > self.nearbyChastot[j + 1].raznitsa:  
 self.nearbyChastot[j], self.nearbyChastot[j + 1] = self.nearbyChastot[j + 1], self.nearbyChastot[j]  
  
**class** RaznitsaWithIshodnayaBukva():  
 bukva = "" #буква из алфавита  
 raznitsa = 0  
  
 **def** \_\_init\_\_(self, bukva, raznitsa):  
 self.bukva = bukva  
 self.raznitsa = raznitsa  
  
  
**def** SearchChastotaInText(text, bukvaArray, dict):  
 **for** i **in** text:  
 **if** i **in** dict:  
 var = bukvaArray[dict.index(i)]  
 var.chastota += 1/len(text)  
 **return None  
  
def** SearchRaznitsaForEachBukva(bukvaWithChastota, freqObr, dict):  
 **for** i **in** range(len(dict)):  
 obj = RaznitsaWithIshodnayaBukva(dict[i], abs(bukvaWithChastota.chastota - freqObr[i]))  
 bukvaWithChastota.nearbyChastot.append(obj)  
 **return None  
  
def** chastota(text):  
 dictRus = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ"  
 dictEngStr = "abcdefghijklmnopqrstuvwxyz"  
 dictRusStr = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя"  
 freqObrRus = [0.062,0.014,0.038,0.013,0.025,0.072,0.0001,0.007,0.016,0.062,0.01,0.028,0.035,0.026,0.053,  
 0.09,0.023,0.04,0.045,0.053,0.021,0.002,0.009,0.004,0.012,0.006,0.003,0.0004,0.016,0.014,0.003,0.006,0.018]  
 dictEng = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"  
 freqObrEng = [0.081,0.016,0.032,0.036,0.123,0.023,0.016,0.051,0.071,0.001,0.005,0.04,0.022,0.072,  
 0.079,0.023,0.002,0.06,0.066,0.096,0.031,0.009,0.02,0.002,0.019,0.001]  
  
 #dictEngFreq = "ETAONISRHLDCUPFMWYBGVKQXJZ"  
 dictEngFreq = "ETAOINSHRDLCUMWFGYPBVKXJQZ"  
 # dictRusFreq = "ОЕАИНТСРВЛКМДПУЯЫЗЬБГЧЙХЖЁЮШЦЩЭФЪ"  
 dictRusFreq = "ОЕАИНТСРВЛКМДПУЯЫЬГЗБЧЙХЖШЮЦЩЭФЪЁ"  
  
 **if not** text:  
 msgBox = QtWidgets.QMessageBox()  
 msgBox.setWindowTitle("Ошибка")  
 msgBox.setText("Введите текст!")  
 msgBox.exec\_()  
 **return** ("")  
  
 dict = ""  
 dict1 = ""  
 freqObr = []  
 # какой алфавит первый символ из алфавита  
 **for** i **in** text:  
 **if** i **in** dictEng **or** i **in** dictEngStr:  
 dict = dictEng  
 dict1 = dictEngFreq  
 freqObr = freqObrEng  
 **break  
 elif** i **in** dictRus **or** i **in** dictRusStr:  
 dict = dictRus  
 dict1 = dictRusFreq  
 freqObr = freqObrRus  
 **break** # else:  
 # msgBox = QtWidgets.QMessageBox()  
 # msgBox.setWindowTitle("Ошибка")  
 # msgBox.setText("Введите русские или английские заглавные буквы!")  
 # msgBox.exec\_()  
 # return ("")  
  
 text = oneRegistr(text, dict)  
  
 #массив объектов, букв по алфавиту  
 bukvaArray = []  
 **for** i **in** dict:  
 obj = IshodnayaBukva(i)  
 bukvaArray.append(obj)  
  
 #относительные частоты  
 SearchChastotaInText(text, bukvaArray, dict)  
  
 **for** i **in** bukvaArray:  
 SearchRaznitsaForEachBukva(i, freqObr, dict)  
  
 **for** i **in** bukvaArray:  
 #i.sort()  
 i.print()  
  
 **for** i **in** range(len(bukvaArray) - 1):  
 **for** j **in** range(len(bukvaArray) - 1):  
 **if** bukvaArray[j].chastota < bukvaArray[j + 1].chastota:  
 bukvaArray[j], bukvaArray[j + 1] = bukvaArray[j + 1], bukvaArray[j]  
  
 print('After sort.')  
 **for** i **in** bukvaArray:  
 #i.sort()  
 i.print()  
  
 **for** i **in** range(len(bukvaArray)):  
 print(bukvaArray[i].bukva)  
 bukvaArray[i].bukvaInSootv = dict1[i]  
  
 # for i in range(len(dict)):  
 # minRaznitsa = 1  
 # minBukva = ""  
 # for j in range(len(bukvaArray)):  
 # if bukvaArray[j].nearbyChastot[i].raznitsa < minRaznitsa and bukvaArray[j].bukvaInSootv == "":  
 # minRaznitsa = bukvaArray[j].nearbyChastot[i].raznitsa  
 # minBukva = bukvaArray[j]  
 # minBukva.bukvaInSootv = dict[i]  
  
 **for** i **in** bukvaArray:  
 print(i.bukva, i.bukvaInSootv)  
  
 result = ""  
 **for** i **in** text:  
 **if** i **in** dict:  
 **for** j **in** bukvaArray:  
 **if** i == j.bukva:  
 result+=j.bukvaInSootv  
 **break  
 else**:  
 result+=i  
 print(result)  
 **return** result, bukvaArray  
  
**def** gistogramma(text, bukvaArray):  
 dictRus = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ"  
 freqObrRus = [0.062,0.014,0.038,0.013,0.025,0.072,0.0001,0.007,0.016,0.062,0.01,0.028,0.035,0.026,0.053,  
 0.09,0.023,0.04,0.045,0.053,0.021,0.002,0.009,0.004,0.012,0.006,0.003,0.0004,0.016,0.014,0.003,0.006,0.018]  
 dictEng = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"  
 freqObrEng = [0.081,0.016,0.032,0.036,0.123,0.023,0.016,0.051,0.071,0.001,0.005,0.04,0.022,0.072,  
 0.079,0.023,0.002,0.06,0.066,0.096,0.031,0.009,0.02,0.002,0.019,0.001]  
 dictEngStr = "abcdefghijklmnopqrstuvwxyz"  
 dictRusStr = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя"  
  
 **if not** text **or not** bukvaArray:  
 msgBox = QtWidgets.QMessageBox()  
 msgBox.setWindowTitle("Ошибка")  
 msgBox.setText("Введите текст!")  
 msgBox.exec\_()  
 **return** ("")  
  
 dict = ""  
 freqObr = []  
 # какой алфавит первый символ из алфавита  
 **for** i **in** text:  
 **if** i **in** dictEng **or** i **in** dictEngStr:  
 dict = dictEng  
 freqObr = freqObrEng  
 **break  
 elif** i **in** dictRus **or** i **in** dictRusStr:  
 dict = dictRus  
 freqObr = freqObrRus  
 **break** index = np.arange(len(dict))  
 values1 = []  
 values2 = []  
  
 # for i in range(len(dict)):  
 # values2.append(bukvaArray[i].chastota)  
  
 tmp = []  
 **for** i **in** range(len(dict)):  
 **for** j **in** range(len(bukvaArray)):  
 **if** bukvaArray[j].bukva == dict[i]:  
 res = bukvaArray[j].bukva + str(bukvaArray[j].bukvaInSootv)  
 values2.append(freqObr[dict.index(str(bukvaArray[j].bukvaInSootv))])  
 tmp.append(res)  
 values1.append(bukvaArray[j].chastota)  
 **break** plt.figure(figsize=(12, 7))  
 bw = 0.4  
 plt.title("Frequency", fontsize=20)  
 plt.bar(index, values1, bw, color='b')  
 plt.bar(index + bw, values2, bw, color='g')  
 plt.xticks(index + bw, tmp)  
 plt.show()  
 **return None  
  
def** printBukva(bukvaArray:list[IshodnayaBukva]):  
 **if not** bukvaArray:  
 msgBox = QtWidgets.QMessageBox()  
 msgBox.setWindowTitle("Ошибка")  
 msgBox.setText("Заполните поля!")  
 msgBox.exec\_()  
 **return** ("")  
  
 **for** i **in** bukvaArray:  
 i.sort()  
 res = ""  
 **for** i **in** bukvaArray:  
 **if** i.chastota != 0:  
 i.sort()  
 res += i.bukva + " -> "  
 **for** k **in** i.nearbyChastot:  
 res += str(k.bukva) + " "  
 res += "**\n**"  
 **return** res  
  
**def** zamena(a, b, bukvaArray, restext):  
 **global** newSootv  
 dictRus = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ"  
 dictEng = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"  
 dict = ""  
 **for** i **in** bukvaArray:  
 **if** i.bukva **in** dictRus:  
 dict = dictRus  
 **else**:  
 dict = dictEng  
  
 **if** a **not in** dict **or** b **not in** dict:  
 msgBox = QtWidgets.QMessageBox()  
 msgBox.setWindowTitle("Ошибка")  
 msgBox.setText("Буквы только из алфавита!")  
 msgBox.exec\_()  
 **return** ("")  
  
 **if not** a **or not** b **or not** restext **or not** bukvaArray:  
 msgBox = QtWidgets.QMessageBox()  
 msgBox.setWindowTitle("Ошибка")  
 msgBox.setText("Заполните все поля!")  
 msgBox.exec\_()  
 **return** ("")  
  
  
 **for** i **in** bukvaArray:  
 **if** i.bukva == a:  
 newSootv = i.bukvaInSootv  
 i.bukvaInSootv = b  
 **elif** i.bukvaInSootv == b **and** i.bukva != a:  
 i.bukvaInSootv = newSootv  
 **for** i **in** range(len(restext)):  
 **if** restext[i] == newSootv:  
 restext = restext[:i] + b + restext[i + 1:]  
 **elif** restext[i] == b:  
 restext = restext[:i] + newSootv + restext[i + 1:]  
 **return** restext, bukvaArray  
  
 # for i in bukvaArray:  
 # if i.bukvaInSootv == a:  
 # i.bukvaInSootv = b  
 # elif i.bukvaInSootv == b:  
 # i.bukvaInSootv = a  
 # for i in range(len(restext)):  
 # if restext[i] == a:  
 # restext = restext[:i] + b + restext[i + 1:]  
 # elif restext[i] == b:  
 # restext = restext[:i] + a + restext[i + 1:]  
 # return restext, bukvaArray

1. **Примеры работы программы**

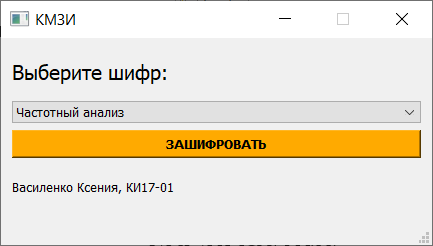


Рисунок 1 – Главное окно.

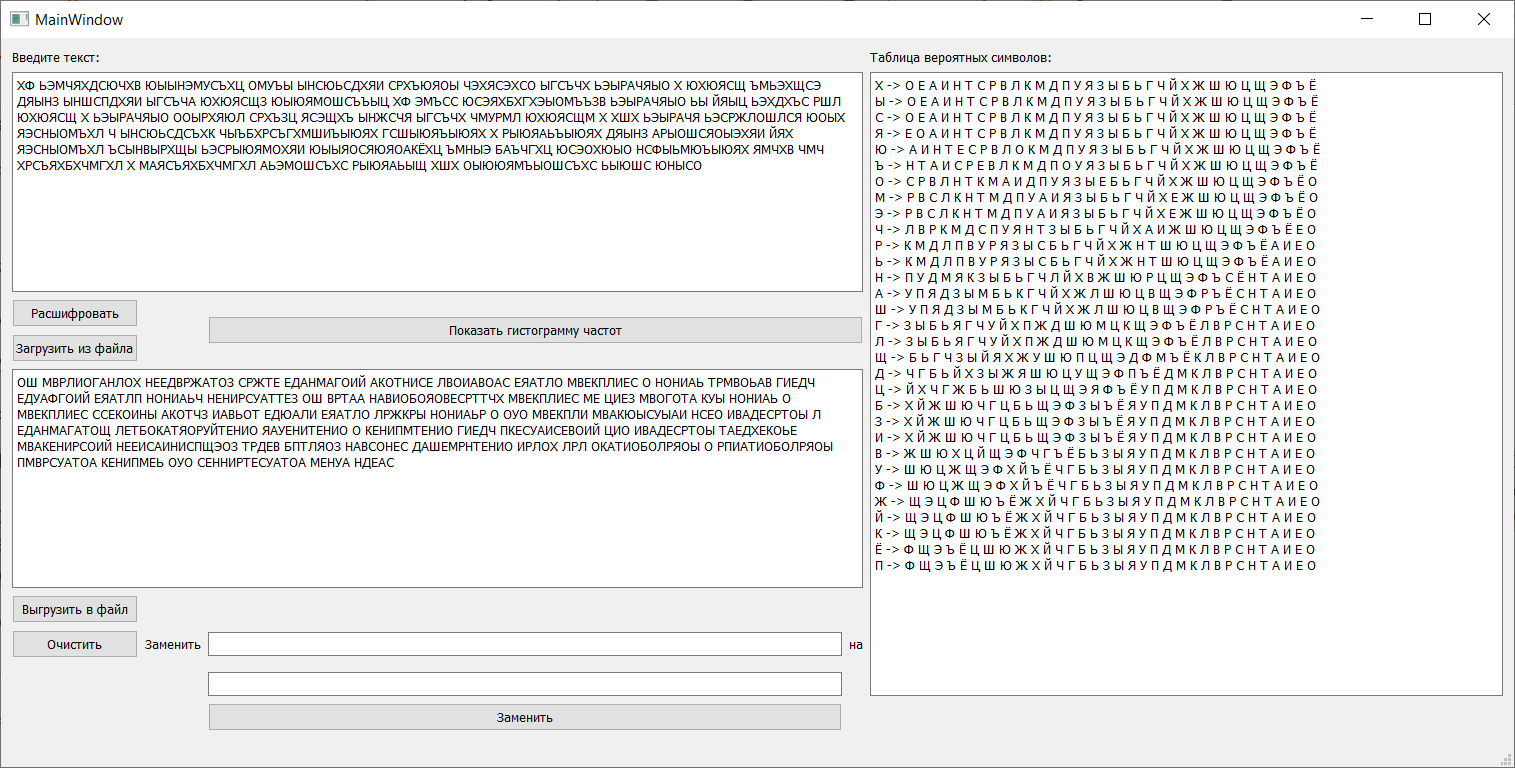


Рисунок 2 – Метод частотного анализа шифротекста.

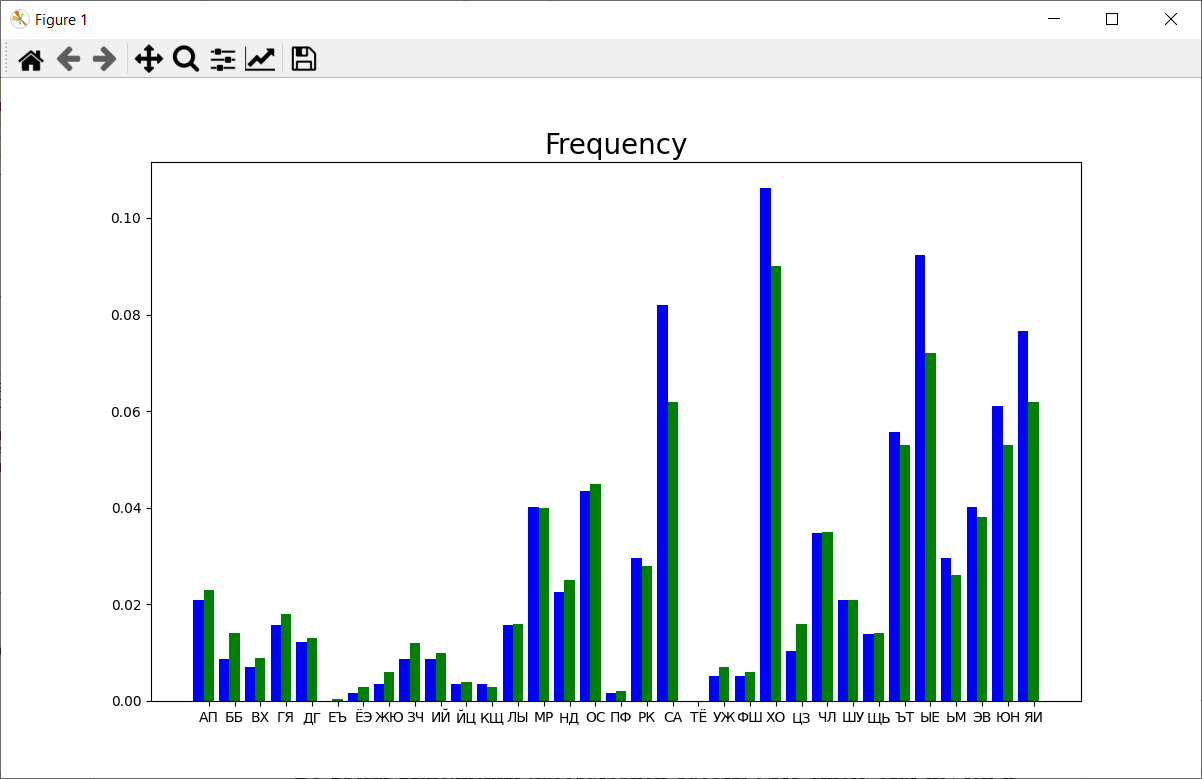


Рисунок 3 – Гистограмма частот.

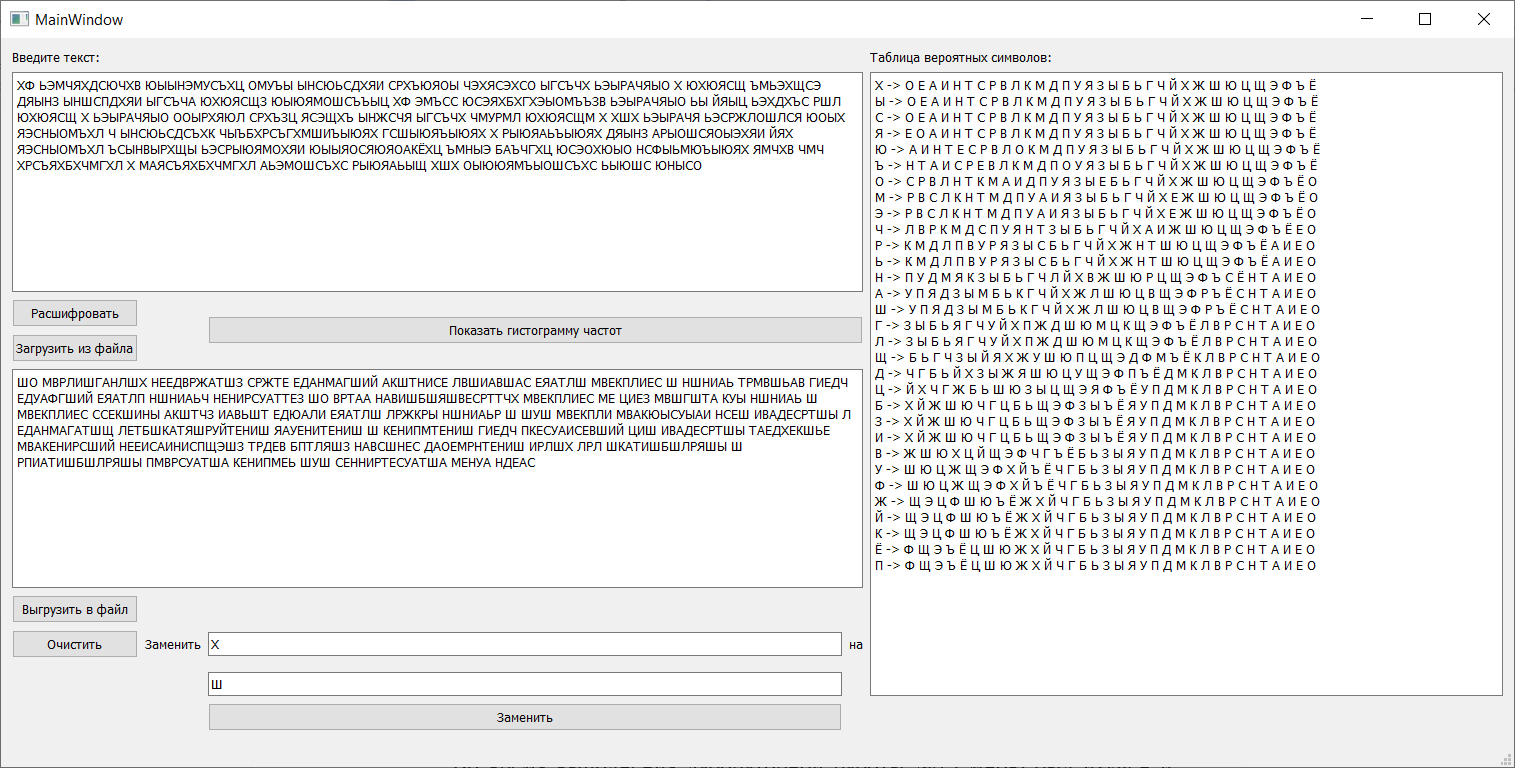


Рисунок 4 – Замена Х на Ш.

# ЗАКЛЮЧЕНИЕ

Во время выполнения лабораторной работы №13 мною был изучен и реализован программно метод частотного анализа шифротекста, а также предусмотрен графический интерфейс с помощью PyQt5 и Qt designer.