Министерство образования и науки Российской Федерации  
ФГАОУ ВПО «Уральский федеральный университет имени первого  
Президента России Б.Н. Ельцина»

Научно-учебный центр «Информационная Безопасность»

Отчет

По дисциплине «Криптографические методы защиты информации»

Лабораторная работа №1

Подпись ФИО

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Агафонов

Студенты \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Л.В. Родионов

И.А. Ибрагимов

Группа РИ-481220

Екатеринбург 2022

Цель работы: изучить исторически значимые методы шифрования на основе замен и их недостатки.

Задачи:

1. Создать программу, которая зашифровывает и расшифровывает файлы с

использованием шифра Цезаря (с. 11 пособия «Основы криптографии» Алферова и

др., далее - пособия). При этом имена входного и выходного файлов, а также режим

работы (зашифрование/расшифрование) и ключ (в качестве которого используется

величина смещения алфавита) задаются параметрами командной строки программы

(подсказка: sys.argv).

2. Зашифровать и расшифровать произвольный текстовый файл.

3. Изучить главу 7 пособия.

4. Выполнить криптоанализ на основе частот появления символов: построить 2

гистограммы, показывающие доли каждого из символов: 1) в каком-либо объемном

текстовом файле (предлагается использовать текст «Войны и мира» Л.Н. Толстого из

прилагаемого файла) и 2) в зашифрованном файле. Убедиться в совпадении кодов

наиболее часто встречающихся зашифрованных символов с соответствующими

кодами открытого текста.

Ход работы:

Шифр Цезаря основывается на сдвиге каждой буквы сообщения на определенное количество символов по алфавиту. Так, при кодировании буквы «А» со смещением в 3 символа получится «Д». В первой лабораторной работе по курсу «Криптографические методы защиты информации» нашей задачей является реализовать шифрование файлов методом Цезаря. Для решения данной задачи был использован ЯП Python. Python является высокоуровневым языком программирования, который отлично подходит для работы с данными с помощью встроенных библиотек. Для написания рабочей программы требовалось вспомнить/ изучить такие разделы языка, как:

* Работа с файлами (открытие, закрытие, чтение, запись)
* Работа со строками (Перебор символов строки, добавление символов и строк в конец строки)
* Словари (ассоциативные массивы) – обновление словарей, инициализация, получение ключей и значений
* Работа с циклами while и for
* Работа с пользовательскими функциями
* Работа с библиотеками numpy, sys и matplotlib

Код программы с комментариями представлен ниже:

Файл cezar.py принимает в качестве аргументов консоли файл, который необходимо зашифровать(infile) , имя зашифрованного файла (outfile), режим работы: encode или decode (mode) и смещение (step).

import sys  
  
  
def change\_char(c: str, d: dict, d1: dict, n: int): # Функция замены символа  
 # c - char, d - словарь типа a0 , d1 - словарь типа 0a, n - шаг  
 sym = d1[(d[c] + n) % len(d)]  
 return sym  
  
  
infile = sys.argv[1]  
outfile = sys.argv[2]  
mode = sys.argv[3]  
step = int(sys.argv[4])  
match mode:  
 case "encode":  
 step = abs(step)  
 case "decode":  
 step = - abs(step)  
 case \_:  
 print("Допустимые режимы работы: encode - зашифровать, decode - расшифровать")  
 exit(1)  
  
  
  
  
# Словарь строчных букв кириллицы  
  
rus\_a0 = {}  
rus\_0a = {}  
for k in range(1072, 1104):  
 rus\_a0.update({chr(k): k-1072})  
 rus\_0a.update({k - 1072: chr(k)})  
  
  
# Словарь прописных букв кириллицы  
RUS\_a0 = {}  
RUS\_0a = {}  
for k in range(1040, 1072):  
 RUS\_a0.update({chr(k): k-1040})  
 RUS\_0a.update({k - 1040: chr(k)})  
  
  
# Словарь строчных букв латиницы  
eng\_a0 = {}  
eng\_0a = {}  
for k in range(97, 123):  
 eng\_a0.update({chr(k): k-97})  
 eng\_0a.update({k - 97: chr(k)})  
  
# Словарь прописных букв латиницы  
ENG\_a0 = {}  
ENG\_0a = {}  
for k in range(65, 91):  
 ENG\_a0.update({chr(k): k-65})  
 ENG\_0a.update({k - 65: chr(k)})  
  
  
# Открываем файл для чтения  
f = open(infile, 'r', encoding='utf-8')  
# Открываем файл для записи  
f2 = open(outfile, 'w', encoding='utf-8')  
# Бесконечный цикл для считывания строк из файла  
while True:  
 line = f.readline() # Считываем строку  
 if not line:  
 break # Если строка пустая - конец файла - выход из цикла  
 line2 = ''  
 for i in line: # Проверяем каждый символ считанной строки  
 if (ord(i) >= 1072) and (ord(i) <= 1103):  
 line2 = line2 + change\_char(i, rus\_a0, rus\_0a, step)  
 elif (ord(i) >= 1040) and (ord(i) <= 1071):  
 line2 = line2 + change\_char(i, RUS\_a0, RUS\_0a, step)  
 elif (ord(i) >= 97) and (ord(i) <= 122):  
 line2 = line2 + change\_char(i, eng\_a0, eng\_0a, step)  
 elif (ord(i) >= 65) and (ord(i) <= 90):  
 line2 = line2 + change\_char(i, ENG\_a0, ENG\_0a, step)  
 else:  
 line2 = line2 + i # если символ не относится ни к одному словарю - переносим без изменений  
 f2.write(line2) # Записываем измененную строку в файл  
f.close() # Закрываем файлы  
f2.close()

Для тестирования работоспособности программы был создан файл test1.txt со следующим содержимым:

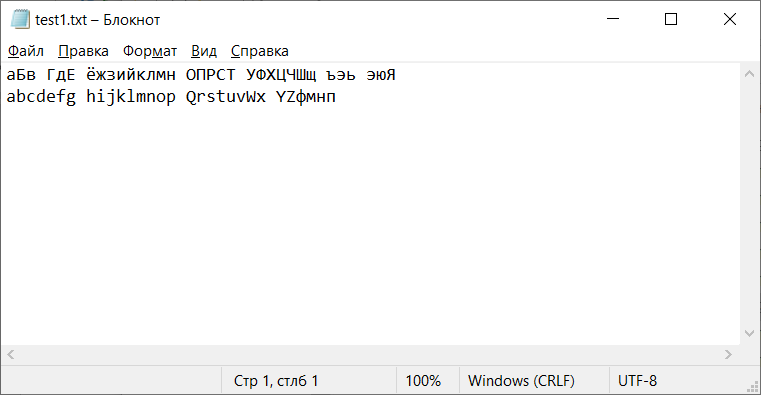


Рисунок 1 – Шифруемый текстовый файл

Для того, чтобы зашифровать данный файл с помощью сценария cezar.py необходимо ввести следующую команду в консоль:



Рисунок 2 – кодирование тестового файла

В результате зашифрованный файл выглядит так:

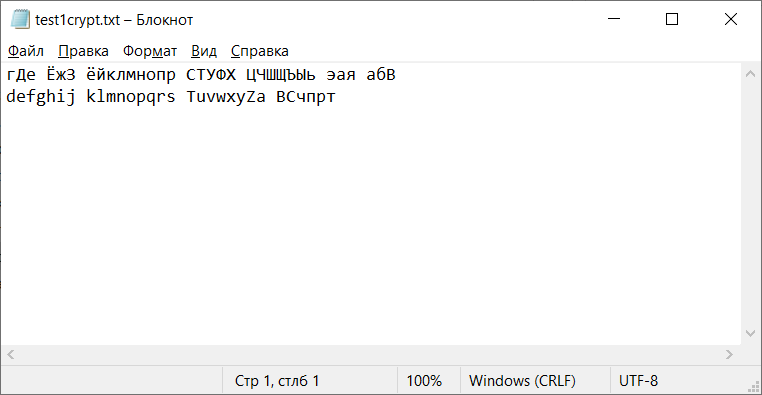


Рисунок 3 – зашифрованный тестовый файл

Для проверки работоспособности написанного сценария, был зашифрован файл, содержащий текст романа Л.Н. Толстого «Война и мир»

Так выглядят первые строки текстового файла:



Рисунок 4 – «Война и мир» оригинал

Команда для шифрования файла: python .\cezar.py .\war\_and\_peace.txt .\crypt.txt encode 31



Рисунок 5 – Зашифрованный файл «Война и мир»

Проведение криптоанализа

В качестве уязвимости шифра Цезаря можно выделить метод, основанный на частотах появления букв в тексте.

Если сравнить гистограммы зашифрованного методом Цезаря и оригинального файлов, то можно заметить, что число вхождений определенной буквы у зашифрованного файла сдвинулось на определенное количество символов по алфавиту.

Для данной задачи был написан отдельный скрипт, использующий в качестве аргумента консоли имя файла, частоту вхождений символов которого мы хотим узнать. Для построения гистограммы нужно подключить библиотеки matplotlib и numpy.

Код сценария выглядит следующим образом:

import sys  
import matplotlib.pyplot as plt  
import numpy as np  
  
# Считываем файл из консоли  
file = sys.argv[1]  
  
# Задаем словарь и заполняем его буквами  
letters = {}  
#for k in range(65, 91):  
 # letters.update({chr(k): 0})  
#for k in range(97, 123):  
 #letters.update({chr(k): 0})  
for k in range(1040, 1104): # 1040 и 1104 - границы интервала таблицы ASCII для строчных и прописных букв кириллицы  
 letters.update({chr(k): 0}) # Добавление в конец словаря пары  
f = open(file, 'r', encoding='utf-8') # Читаем файл в кодировке utf-8  
while True:  
 line = f.readline() # Читаем файл построчно  
 if line == '':  
 f.close()  
 break # Если достигнута пустая строка - конец файла - прекращаем чтение файла и закрываем его  
 for i in line:  
 if i in letters: # Если символ считанной строки найден в словаре, у соответствующей пары вырастает значение на 1  
 letters[i] += 1  
keys = letters.keys()  
values = letters.values()  
plt.bar(keys, np.divide(list(values), sum(values)), color='green', label='Частотный анализ') # Строим гистограмму, np.divide находит отношение вхождения каждой буквы к общему числу букв  
plt.ylim(0, 1) # Нормируем вертикальную ось к единице  
plt.ylabel('Доля вхождений символа')  
plt.xlabel('Символы')  
plt.show()

Так, для оригинального файла Война и Мир гистограмма выглядит следующим образом:

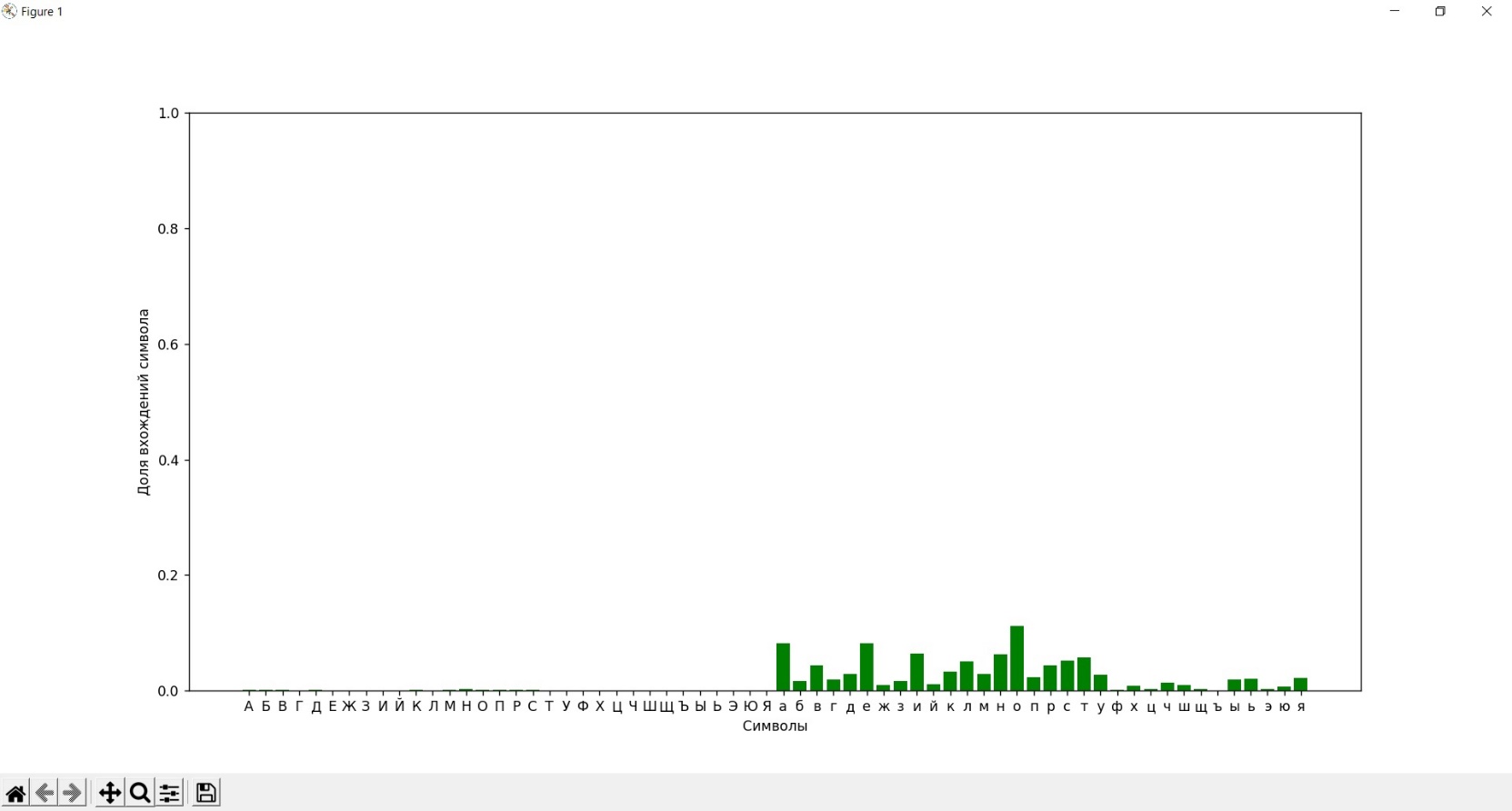


Рисунок 6 – Гистограмма вхождения букв в оригинальном файле Война и мир

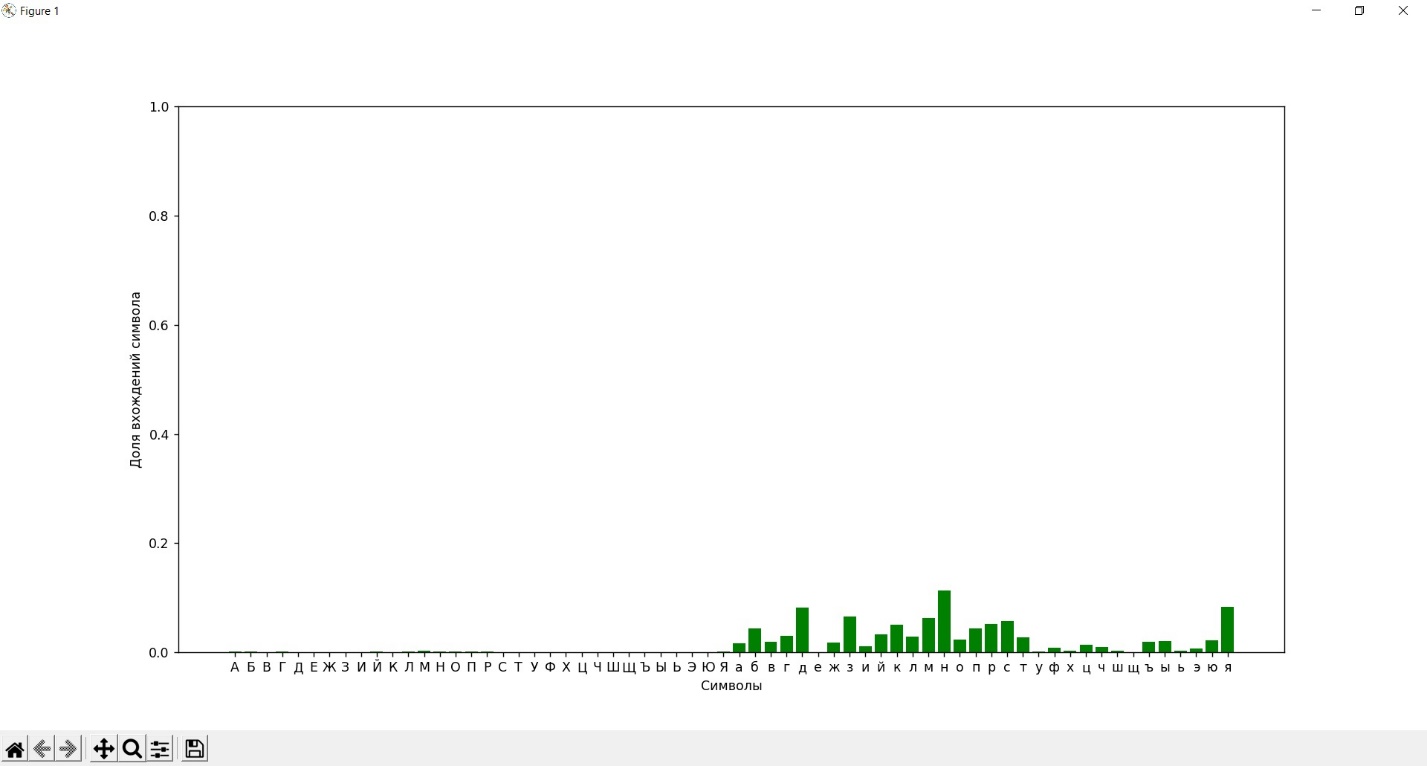


Рисунок 7 - Гистограмма вхождения букв в зашифрованном файле Война и мир

Команда для получения гистограммы выглядит так:

python .\main.py .\war\_and\_peace.txt

Как мы можем заметить, на первой гистограмме максимальный уровень был в столбце над буквой «о». На второй же гистограмме этот уровень уже соответствует букве «н». Соотнеся две данные гистограммы, легко понять на сколько символов был сдвинут текст.

Выводы

В ходе выполнения данной лабораторной работы были получены и улучшены навыки разработки сценариев на языке программирования Python