1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт компьютерных наук и кибербезопасности

**ЛАБОРАТОРНАЯ РАБОТА № 2**

1. «**Основы частотного криптоанализа**»
2. по дисциплине «Основы информационной безопасности»
3. Выполнил
4. студент гр. 5151001/40001 Волошкевич М.А.

<*подпись*>

1. Преподаватель
2. асс. преподавателя Орел Е.М.

<*подпись*>

1. Санкт-Петербург
2. 2025
3. **Цель работы**

В работе необходимо приобрести навыки криптоанализа, ознакомиться со способом дешифрования криптограмм на примере применения метода частного криптоанализа.

1. **Постановка задачи**

Необходимо разработать программу, реализующую функции инструмента криптоаналитика. Кроме того, с использованием данной программы нужно расшифровать данную криптограмму.

1. **Теоретические исследования**

Работа алгоритма шифрования при помощи шифра моноалфавитной подстановки заключается в следующем:

- выбирается набор символов, которые будут использоваться для составления текста, требующего шифрования;

- выбирается алфавит шифрования;

- устанавливается взаимно однозначное соответствие между символами нормативного алфавита и символами алфавита шифрования.

Моноалфавитные подстановки обладают важным свойством – они не нарушают частот появления символов, характерных для данного языка. Все естественные языки имеют характерное частотное распределение символов. Например, буква «О» встречается в русском языке чаще других, а буква «Ъ» − самая редкая.

Для того, чтобы получить открытый текст при помощи частотного анализа, необходимо сопоставить частоты появления символов шифра с вероятностями появления букв используемого алфавита. После этого наиболее частые символы криптограммы заменяются наиболее вероятными символами алфавита, остальные замены производятся на основе вероятных слов и знания синтаксических правил используемого языка.

1. **Описание решения**

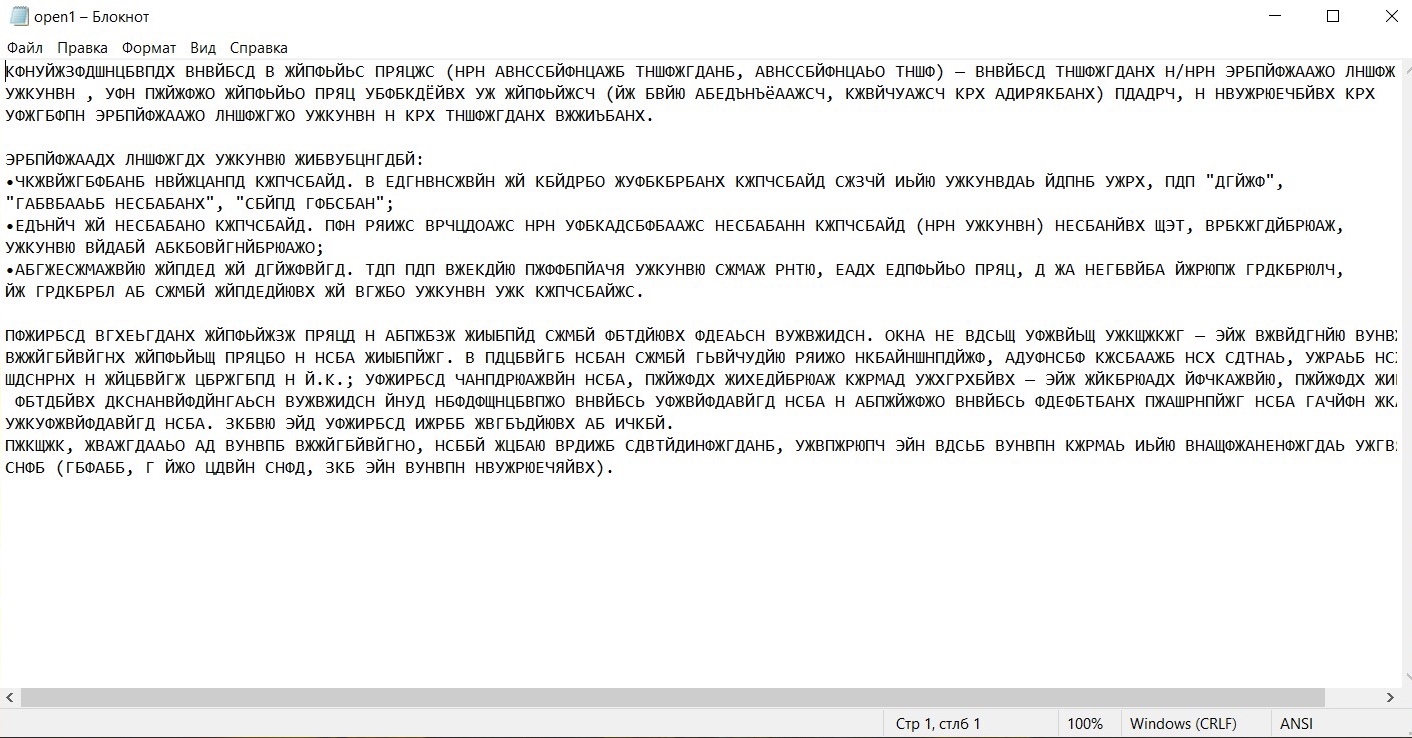
* Исходная криптограмма

Рисунок 1 – Исходная криптограмма

* Реализация:

Программа для работы с криптограммой начинается с определения глобальных переменных: алфавита русского языка в верхнем регистре (alphabet), частотности букв (frequens), словаря соответствия букв и их частот (alph\_freq), а также списка символов, которые нужно удалить из текста (chars\_to\_remove). Также создается список history для хранения истории замен.

Функция frequence\_anal(str) выполняет частотный анализ текста: подсчитывает количество вхождений каждого символа, вычисляет их процентное соотношение и возвращает отсортированный по убыванию частоты словарь.

Функция read\_file(name) читает файл с криптограммой, удаляет ненужные символы, разделяет текст на слова и возвращает три варианта данных: оригинальный текст, текст без пробелов и список уникальных слов.

Программа сразу же считывает и обрабатывает файл 'in.txt', сохраняя результаты в переменные cryptogramm, cryptogramm\_str и cryptogramm\_words, а также сохраняет оригинальный текст в переменную original.

Функция replace\_letter(what, to) выполняет замену символов во всех вариантах криптограммы (как в полном тексте, так и в списке слов) и сохраняет изменения в глобальных переменных.

Основной цикл программы ожидает ввода команд от пользователя и выполняет соответствующие действия:

q - завершение работы программы

freq - частотный анализ криптограммы с выводом предполагаемых замен

mlen - сортировка слов по длине (от самого длинного к короткому)

r - замена одной буквы на другую (например: "r А б")

unenc - сортировка слов по количеству нерасшифрованных (прописных) букв

h - вывод истории замен

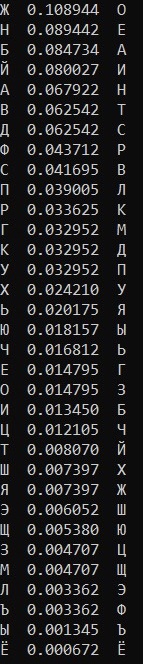
undo - отмена последней замены (можно указать номер замены для отмены)

print - вывод текущего состояния криптограммы

orig - вывод оригинального текста криптограммы

auto - автоматическая замена букв на основе частотного анализа

Все замены автоматически сохраняются в истории, что позволяет в любой момент просмотреть или отменить их. Программа работает с русскими буквами в обоих регистрах, учитывая, что прописные буквы считаются нерасшифрованными, а строчные - расшифрованными.

1. **Тестирование и результаты работы программы**

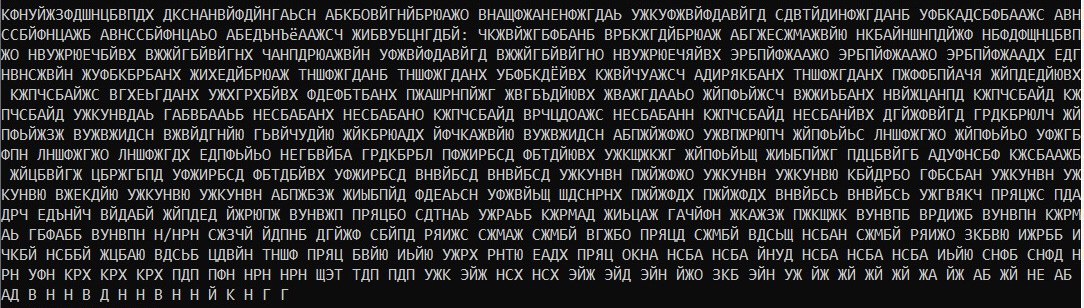
Рисунок 2 – Частотный анализ криптограммы

Рисунок 3 – Сортировка по кол-ву букв

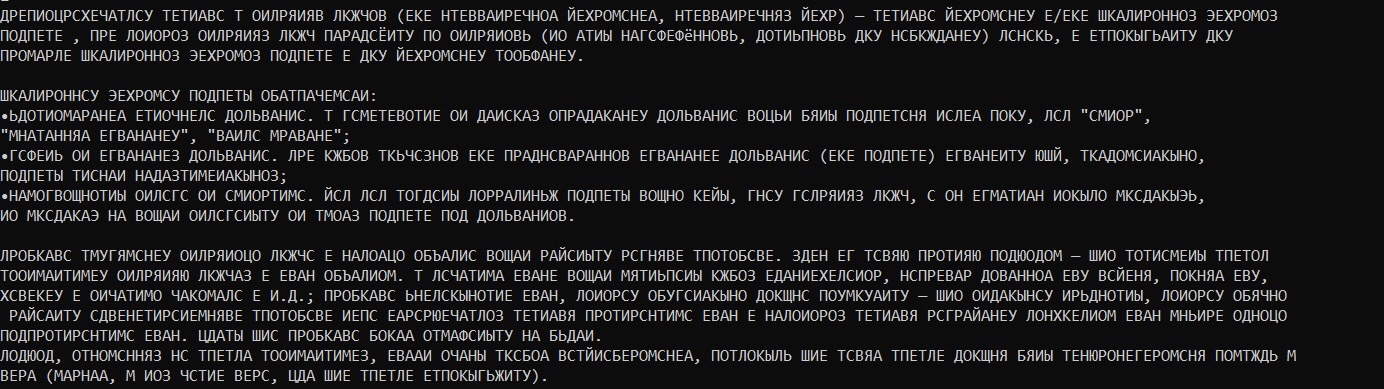


Рисунок 4 – Криптограмма после автозамены

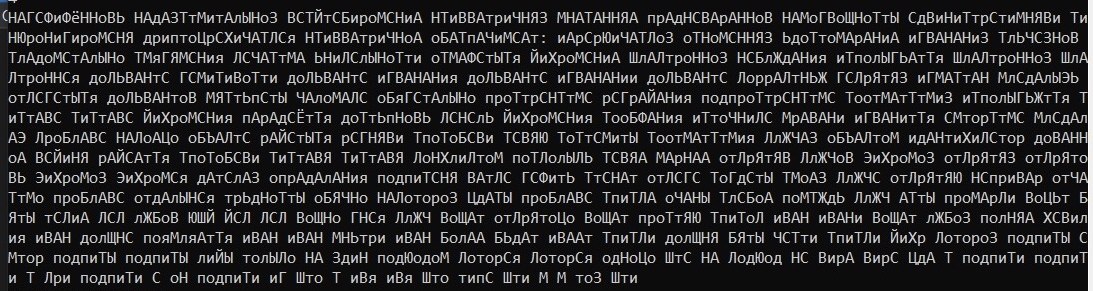
Рисунок 5 – Сортировка по кол-ву нерасшифрованных букв на некоторой интеграции цикла

Рисунок 6 – История замены букв (по красной полоске видно где был откат действия)

Порядок действий с пояснением:

Автозамена  
Е-и  
К-л (из-за Е/ЕКЕ, скорее всего это и/или)  
Д-д  
У-я (ДлУ скорее всего это для)  
П-п  
Р-р (ПРи скорее всего при)  
О- о (пОд скорее всего под)  
И-т (Ио и оИ это то и от)  
З-й (тоЗ это той)  
Л-к (Лоторой это которой)  
В-м (иВя это имя)  
С-а (мирС это мира)  
Ч-ч  
Т-с (ЧаТти это части)  
А-е (систАма это система)  
М-в (аМтор это автор)  
Ж-ю (клЖч это ключ)  
Ц-г  
Х-ф (криптоЦраХическая это криптографическая)  
Я-ы (открЯтЯм это открытым)  
Н-н (верНее это вернее)  
Й-ш (Йифр это шифр)  
Ш-э (Шти это эти)  
Э-ц (Эфровой это цифровой)  
Б-б  
Ъ-ъ (оБЪекта это объекта)  
Ы-ь  
Г-з  
Ь-у (исполЫГЬются это используются)  
Ю-х (один из самыЮ простыЮ поЮдов это один из самых простых подходов)  
Ф-п (ФодФиси это подписи)  
Щ-ж

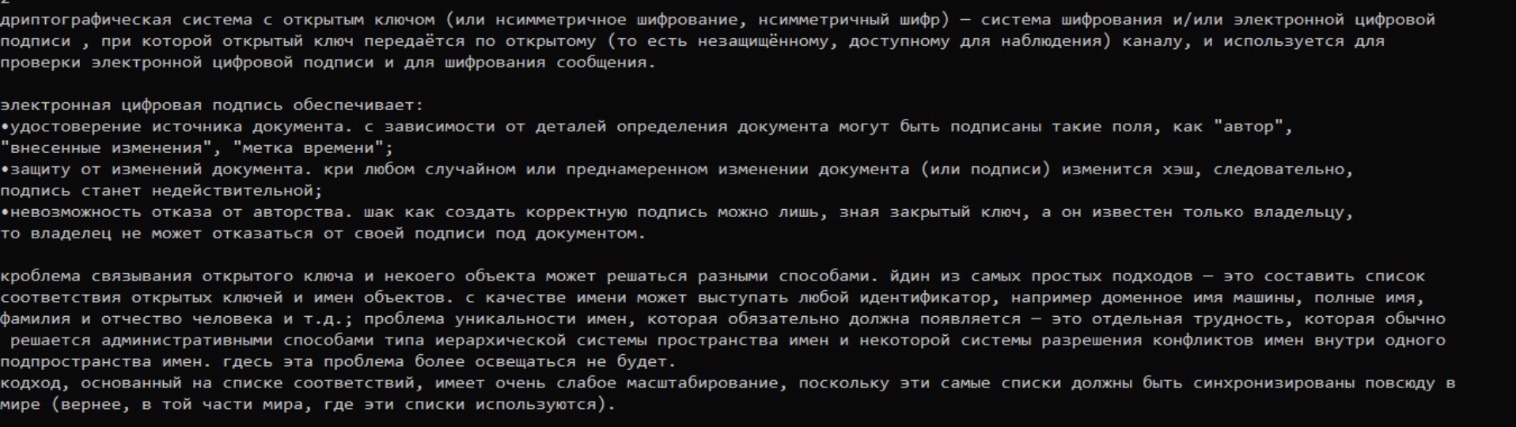


Рисунок 7 – Расшифрованная криптограмма

Таблица 1 – Ключ

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| А | Б | В | Г | Д | Е | Ж | З | И | Й | К | Л | М | Н | О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ |
| **д** | **и** | **г** | **з** | **к** | **б** | **м** | **е** | **н** | **о** | **п** | **р** | **с** | **а** | **ж** | **у** | **ф** | **в** | **й** | **ч** | **ш** | **щ** | **л** | **ц** | **т** | **ъ** |
| Ъ | Ы | Ь | Э | Ю | Я |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **ю** | **ь** | **ю** | **э** | **я** | **х** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**6. Контрольные вопросы**

Что такое шифр моноалфавитной подстановки?

Моноалфавитный шифр подстановки — шифр, при котором каждый символ открытого текста заменяется на некоторый, фиксированный при данном ключе, символ того же алфавита.

Укажите недостатки шифра моноалфавитной подстановки.

Так как буквы в исходном тексте и зашифрованном тексте находятся в отношении один к одному, данный текст легко расшифровать при помощи частотного анализа

Какова сложность дешифрации методом прямого перебора для сообщения, зашифрованного шифром моноалфавитной подстановки?

Сложность дешифрации методом прямого перебора зависит от количества всех возможных решений задачи.

В зависимости от количества всех возможных решений прямой перебор может потребовать экспоненциального времени работы.

Какие условия упрощают частотный анализ?

Частотный анализ может зависеть от особенностей языка шифрования: частота встречаемости пар букв, союзы, предлоги и т.д.

Получится ли правильно расшифрованная криптограмма, если произвести все замены в соответствии с частотами появления букв в русском языке?

Нет, так как частота употребления символа может зависеть от характера текста и не соответствовать частоте появления букв в русском языке, поэтому последующие замены должны производится на основе вероятных слов и знаний синтаксических правил языка.

1. **Выводы**

В ходе работы была разработана программу, реализующую функции инструмента криптоаналитика. И с использованием данной программы была расшифрована криптограмма. Были получены знания в сфере дешифровки и изучен шифр моноалфавитной подстановки.

ПРИЛОЖЕНИЕ А

Листинг программы «Основы частотного криптоанализа»

import re

alphabet = 'абвгдеёжзийклмнопрстуфхцчшщъыьэюя'.upper()

frequens = [8.01, 1.59, 4.54, 1.7, 2.98, 8.45, 0.04, 0.94, 1.65, 7.35, 1.21,

3.49, 4.4, 3.21, 6.7, 10.97, 2.81, 4.73, 5.47, 6.26, 2.62, 0.26,

0.97, 0.48, 1.44, 0.73, 0.36, 0.04, 1.90, 1.74, 0.32, 0.64, 2.01

]

alph\_freq = {k: v for k, v in zip(alphabet, frequens)}

alph\_freq = dict(sorted(alph\_freq.items(), key= lambda item: item[1], reverse=True))

chars\_to\_remove = [",", "?", "!", ":", ";", "'", '"', '\n', ".", "—"]

history = []

def frequence\_anal(str):

symbols = [symbol for symbol in str]

letter\_counts = {k: 0 for k in symbols}

for l in str:

letter\_counts[l] += 1

letter\_counts = dict(sorted(letter\_counts.items(), key= lambda item: item[1], reverse=True))

for k, v in letter\_counts.items():

letter\_counts[k] = round(letter\_counts[k] / len(str) \* 100, 2)

return letter\_counts

def read\_file(name):

f = open(name, 'r')

cry = first = f.read().upper()

for char in chars\_to\_remove:

cry = cry.replace(char, "")

word = list(set(re.split(" |\n", cry)))

if '' in word:

word.remove('')

cry = cry.replace(' ', '').replace('\n', '')

return first, cry, word

cryptogramm, cryptogramm\_str, cryptogramm\_words = read\_file('in.txt')

original = cryptogramm

def replace\_letter(what, to):

global cryptogramm, cryptogramm\_words

cryptogramm = cryptogramm.replace(what, to)

for i in range(0, len(cryptogramm\_words)):

cryptogramm\_words[i] = cryptogramm\_words[i].replace(what, to)

while(True):

command = input()

command = command.split(' ')

match command[0]:

case 'q':

break

case 'freq':

cryptogramm\_freq = frequence\_anal(cryptogramm\_str)

keys\_list\_1 = list(alph\_freq.keys())

keys\_list\_2 = list(cryptogramm\_freq.keys())

for i in range(0, len(cryptogramm\_freq)):

print(keys\_list\_2[i], " ", list(cryptogramm\_freq.values())[i], " ", keys\_list\_1[i])

case 'mlen':

temp = sorted(cryptogramm\_words, key=lambda item: len(item), reverse=True)

for word in temp:

print(word)

case 'r':

replace\_letter(command[1], command[2])

history.append(command)

case 'unenc':

len\_unencode = []

for word in cryptogramm\_words:

len\_unencode.append(sum(1 for ch in word if not ch.isupper()))

temp = {k: v for k, v in zip(cryptogramm\_words, len\_unencode)}

temp = dict(sorted(temp.items(), key= lambda item: item[1], reverse=True))

for item in temp:

print(item)

case 'h':

i = 1

for command in history:

print(i, command[1], '->', command[2])

i += 1

case 'undo':

if history:

last = history.pop(int(command[1]) - 1)

replace\_letter(last[2].lower(), last[1])

case 'print':

print(cryptogramm)

case 'orig':

print(original)

case 'auto':

crypt\_freq = frequence\_anal(cryptogramm\_str)

keys\_list\_1 = list(alph\_freq.keys())

keys\_list\_2 = list(crypt\_freq.keys())

for i in range(0, len(crypt\_freq)):

replace\_letter(keys\_list\_2[i], keys\_list\_1[i].lower())

history.append(['r', keys\_list\_2[i], keys\_list\_1[i]])