Федеральное государственное автономное образовательное учреждение высшего образования

Университет ИТМО

Дисциплина: Информационная безопасность (Криптография)

Лабораторная работа 1.1

Вариант 2

Работу выполнил студент группы P34111:
Кривоносов Егор Дмитриевич

Преподаватель:
Маркина Татьяна Анатольевна

2022 г.

г. Санкт-Петербург

Оглавление

Цель работы

Изучение основных принципов шифрования информации, знакомство с широко известными алгоритмами шифрования, приобретение навыков их программной реализации.

Задание

Реализовать шифрование и дешифрацию файла по методу Виженера. Ключевая фраза вводится. Реализовать в программе частотный криптоанализ зашифрованного текста.

Листинг разработанной программы

```
from collections import Counter
from tabulate import tabulate


# Считывает текст из файла
def read_file():
    while True:
        path = input("Введите путь к файлу: ").strip()
        try:
            with open(path, encoding='utf-8') as f:
                data = f.readlines()
                return data[0]
        except FileNotFoundError:
            print("Файл не найден!", '\n')
        except ValueError:
            print("Неправильный формат файла!", '\n')


# Проверяет какой язык используется
def check_lang(plaintext, lang_1, lang_2):
    num = 0
    spec_character = ",.:;'\"-?!/1234567890[]= …“—«»”*&^(){}"
    upper_text = ""
    answer = []
    for letter in plaintext:
        upper_text += letter.upper()
        if spec_character.find(letter) != -1:
            continue
        num = lang_1.find(letter.upper())
    if num != -1:
        answer.append("EN")
```

```
    for letter in plaintext:
        if spec_character.find(letter) != -1:
            continue
        num = lang_2.find(letter.upper())
    if num != -1:
        answer.append("RU")
    answer.append(upper_text)
    return answer


def encrypt(plaintext, key, ALPHABET):
    result_text = ""
    key_ind = 0

    for letter in plaintext:
        num = ALPHABET.find(letter.upper())
        if num != -1:
            num = (ALPHABET.find(letter.upper()) + ALPHABET.find(key[key_ind % len(key)])) %
len(ALPHABET)
            key_ind += 1
            if letter.islower():
                result_text += ALPHABET[num].lower()
            else:
                result_text += ALPHABET[num]
        else:
            result_text += letter

    return result_text


def decrypt(ciphertext, key, ALPHABET):
    result_text = ""
    key_ind = 0

    for letter in ciphertext:
        num = ALPHABET.find(letter.upper())
        if num != -1:
            num = (ALPHABET.find(letter.upper()) - ALPHABET.find(key[key_ind % len(key)])) %
len(ALPHABET)
            key_ind += 1
            if letter.islower():
                result_text += ALPHABET[num].lower()
            else:
                result_text += ALPHABET[num]
        else:
            result_text += letter

    return result_text


# Формируем словарь сопоставляющий по частоте буквы в тексте и в языке
def mapping_dict(counter, DICT):
    decrypt_arr = []

    # Упорядочиваем по частоте появление букв в тексте
```

```
    for i in range(len(counter)):
        decrypt_arr.append(counter.most_common(len(counter))[i][0])
    return dict(zip(decrypt_arr, DICT))


# Простой частотный анализ
def freq_analysis(ciphertext, ALPHABET, DICT):
    spec_character = ",.:;'\"-?!/1234567890[]= …‟—«»”*&^(){}"
    modified_ciphertext = "".join([(char if char not in spec_character else '') for char in
ciphertext])
    modified_ciphertext = modified_ciphertext.replace('\n', '')

    chars_counter = Counter(modified_ciphertext.upper())

    amount = 0
    for i in range(len(ALPHABET)):
        amount += chars_counter[ALPHABET[i]]

    table = []
    for i in range(len(ALPHABET)):
        table.append([ALPHABET[i], chars_counter[ALPHABET[i]], round(chars_counter[ALPHABET[i]]
/ amount * 100, 3)])

    print("\n Частотный анализ:")
    print(tabulate(table, headers=["Буква", "Количество", "Частота"], tablefmt="grid",
floatfmt="2.5f"), "\n")

    new_DICT = mapping_dict(chars_counter, DICT)
    print(new_DICT)

    # Если в исходном зашифрованном тексте встречается символ, то сохраняется, иначе буква
сопоставляется по словарю
    new_result = ""
    j = 0
    for i in ciphertext:
        num = ALPHABET.find(i.upper())
        if num != -1:
            if ciphertext[j].islower():
                new_result += (new_DICT.get(modified_ciphertext[j].upper())).lower()
            else:
                new_result += new_DICT.get(modified_ciphertext[j].upper())
            j += 1
        else:
            new_result += i

    return new_result


# Запускает основную программу
if __name__ == '__main__':
    # Алфавит для сдвига
    ALPHABET_EN = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    ALPHABET_RU = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ"

    # Упорядоченные по частоте появления в языке буквы
    DICT_EN = [*'ETAOINSHRDLCUMWFGYPBVKJXQZ']
```

```
    DICT_RU = [*'ОЕЁАИТНСРВЛКМДПУЯЫГЗБЧЙХЪЖЬЮШЦЩЭФ']

    mode = ""
    key = ""
    key_lang = ""
    ALPHABET = ALPHABET_RU
    DICT = DICT_RU

    # Считываем текст с файла
    text = read_file()
    text_lang = check_lang(text, ALPHABET_EN, ALPHABET_RU)[0]

    print(f"Исходный текст: {text} \n")

    # Выбираем режим Шифрование / Расшифрование
    while True:
        mode = input("Введите 'enc' для шифрования или 'dec' для расшифрования: ")
        if mode == "enc" or mode == "dec":
            break
        print("Выбран неправильный режим!", '\n')

    key = input("Введите ключ: ")
    key_lang, new_key = check_lang(key, ALPHABET_EN, ALPHABET_RU)

    # Проверяем языки ключа и исходного текста
    if key_lang != text_lang:
        print("Язык ключа и текста не совпадают, попробуйте снова!")
        print(f"key_lang: {key_lang} != {text_lang} :text_lang")
        exit(-1)

    if text_lang == "EN":
        ALPHABET = ALPHABET_EN
        DICT = DICT_EN

    # Шифрование и расшифрование исходного текста
    result = encrypt(text, new_key, ALPHABET) if mode == "enc" else decrypt(text, new_key,
ALPHABET)
    print(f"Результат: {result}")

    # Если у нас было шифрование, тогда производим частотный анализ
    if mode == "enc":
        print(f"Частотный анализ результат: {freq_analysis(result, ALPHABET, DICT)}")
```

Результаты работы программы

Шифровка

Пример работы:

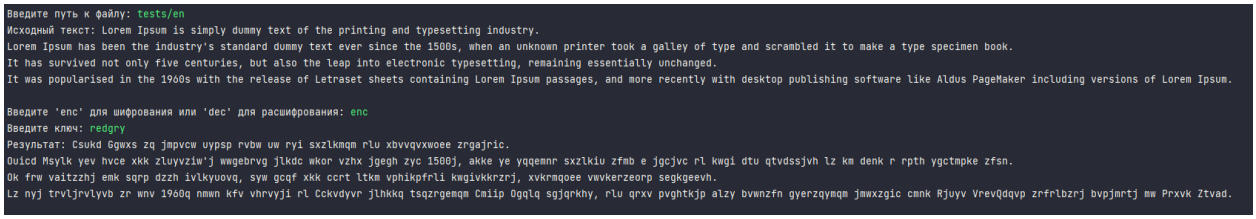
Исходный текст: Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Ключ: redgry

Результат: Csukd Ggwxs zq jmpvcw uypsp rvbw uw ryi sxzlkmqm rlu xbvvqvxwoee zrgajric. Ouicd Msylk yev hvce xkk zluyvziw'j wwgebrvg jlkdc wkor vzhx jgegh zyc 1500j, akke ye yqqemnr sxzlkiu zfmb e jgcjvc rl kwgi dtu qtvdssjvh lz km denk r rpth ygctmpke zfsn. Ok frw vaitzzhj emk sqrp dzzh ivlkyuovq, syw gcqf xkk ccrt ltkm vphikpfrli kwgivkkrzrj, xvkrmqoee vwvkerzeorp segkgeevh. Lz nyj trvljrvlyvb zr wnv 1960q nmwn kfv vhrvyji rl Cckvdyvr jlhkkq tsqzrgemqm Cmiip Ogqlq sgjqrkhy, rlu qrxv pvghtkjp alzy bvwnzfn gyerzqymqm jmwxzgic cmnk Rjuyv VrevQdqvp zrfrlbzrj bvpjmrtj mw Prxvk Ztvad.

Скриншоты вывода:

Шифрование

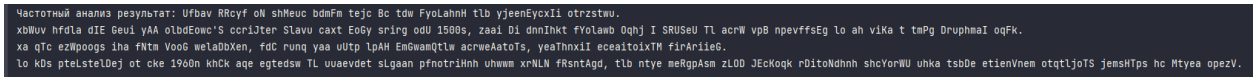


Частотный анализ

Частотный анализ:

Буква	Количество	Частота
A	5	1.07100
B	8	1.71300
C	18	3.85400
D	10	2.14100
E	23	4.92500
F	8	1.71300
G	20	4.28300
H	13	2.78400
I	15	3.21200
J	22	4.71100
K	35	7.49500
L	20	4.28300
M	21	4.49700
N	10	2.14100
O	8	1.71300
P	15	3.21200
Q	22	4.71100
R	38	8.13700
S	14	2.99800
T	12	2.57000
U	11	2.35500
V	41	8.77900
W	19	4.06900
X	12	2.57000
Y	20	4.28300
Z	27	5.78200

Расшифровка обычным частотным анализом



Расшифровка

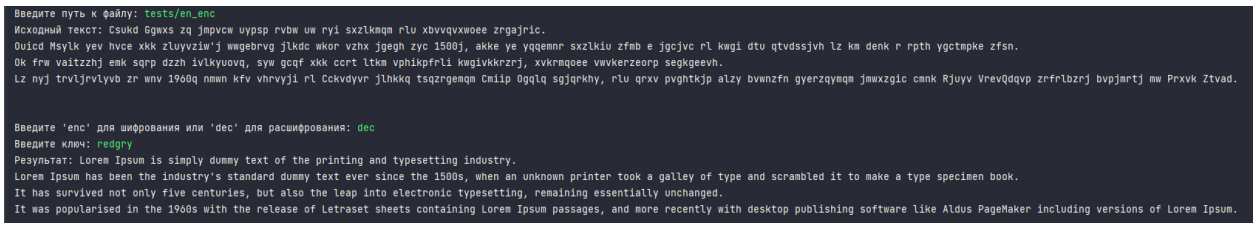
Пример работы:

Исходный текст: Csukd Ggwxs zq jmpvcw uypsp rvbw uw ryi sxzlkmqm rlu xbvvqvxwoee zrgajric. Ouicd Msylk yev hvce xkk zluyvziw'j wwgebrvg jlkdc wkor vzhx jgegh zyc 1500j, akke ye yqqemnr sxzlkiu zfmb e jgcjvc rl kwgi dtu qtvdssjvh lz km denk r rpth ygctmpke zfsn. Ok frw vaitzzhj emk sqrp dzzh ivlkyuovq, syw gcqf xkk ccrt ltkm vphikpfrli kwgivkkrzrj, xvkrmqoee vwvkerzeorp segkgeevh. Lz nyj trvljrvlyvb zr wnv 1960q nmwn kfv vhrvyji rl Cckvdyvr jlhkkq tsqzrgemqm Cmiip Ogqlq sgjqrkhy, rlu qrxv pvghtkjp alzy bvwnzfn gyerzqymqm jmwxzgic cmnk Rjuyv VrevQdqvp zrfrlbzrj bvpjmrtj mw Prxvk Ztvad.

Ключ: redgry

Результат: Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Скриншоты вывода:



Вывод

В результате выполнения лабораторной работы были получены навыки программной реализации алгоритма Виженера на языке Python и проведения его частотного анализа простым способом. Обычный частотный анализ очевидно не даёт внятного результата, в силу того, что шифр Виженера по сути состоит из нескольких частей текста, каждая из

которых имеет свой сдвиг, а значит и анализ надо проводить для каждого из блоков текста по отдельности.

Полезные ссылки

https://habr.com/ru/post/517410/ - частотный анализ
https://gist.github.com/dssstr/aedbb5e9f2185f366c6d6b50fad3e4a4 - реализации метода Виженера
Как работает метод Виженера
https://planetcalc.ru/2468/ - шифр Виженера онлайн