Ковкель Никита, ФИТ 3-4

Информационная безопасность

Отчет по лабораторной работе № 4.

**Цель**: изучение и приобретение практических навыков разработки и использования приложений для реализации подстановочных шифров

**1. Практическая часть**

В данной лабораторной работе необходимо разработать авторское приложение, которое будет реализовывать зашифрование/расшифрование текстовых документов (объемом не менее 5 тысяч знаков), созданных на основе польского алфавита.

Для зашифрования и расшифрования были разработаны функции Substitution и Trisemus, приведённые в листингах 1.1 и 1.2 соответственно.

|  |
| --- |
| def Substitution(text, alphabet, k, decrypt=False):      k = -k if decrypt else k      new\_text = [          alphabet[(alphabet.index(char) + k) % len(alphabet)] if char in alphabet else char          for char in text      ]      return ''.join(new\_text) |

Листинг 4.1 – Substitution

|  |
| --- |
| def Trisemus(text, key, alphabet, decrypt=False):      key = ''.join(sorted(set(key), key=key.index))      grid = key + ''.join([char for char in alphabet if char not in key])      new\_text = [          grid[new\_index(char, grid, key, decrypt)] if char in grid else char          for char in text      ]      return ''.join(new\_text)  def new\_index(char, grid, key, decrypt):      index = grid.index(char)      row = index // len(key)      col = index % len(key)      if decrypt:          return (row - 1) \* len(key) + col if row > 0 else (len(grid) // len(key) - 1) \* len(key) + col      else:          return ((row + 1) % (len(grid) // len(key))) \* len(key) + col |

Листинг 4.2 – Trisemus

Функция Substitution реализует алгоритм шифрования и дешифрования методом подстановки (шифр Цезаря). Она заменяет каждую букву в тексте на другую букву из алфавита с определенным сдвигом

Функция возвращает новый текст, полученный после применения шифра подстановки к входному тексту. Функция Trisemus реализует алгоритм шифрования и дешифрования с использованием Трисемус-квадрата. Этот метод шифрования использует матрицу, составленную из ключа и оставшихся символов алфавита.

Анализ гистограмм

Гистограммы исходного текста и расшифрованного полностью совпадают. Также размеры столбцов в исходным тесте и зашифрованном совпадают, но располагаются в другом порядке.

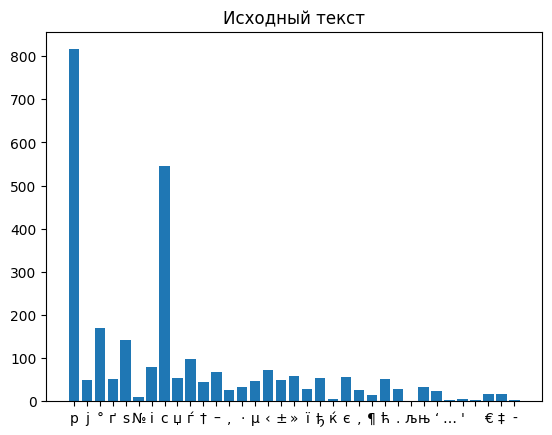


Рисунок 1.1 – Гистограмма исходного текста

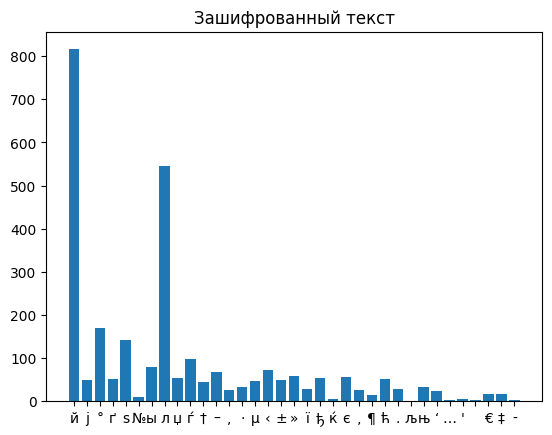


Рисунок 1.2 – Гистограмма зашифрованного текста

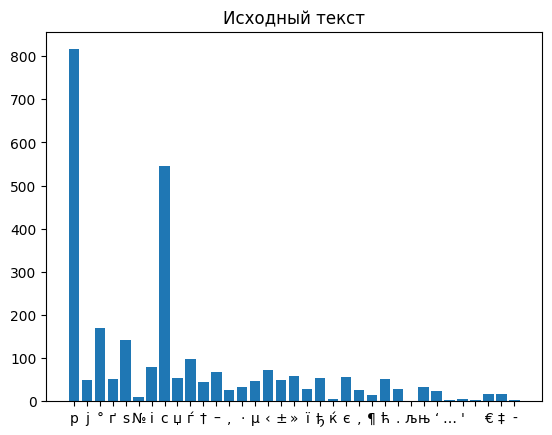


Рисунок 2.3 – Гистограмма расшифрованного текста

Расчёт времени шифрования и расшифрования

Декоратор timer\_decorator представляет собой обертку для функций, которая измеряет время выполнения этих функций. Давайте рассмотрим, как он работает и для чего он нужен.

|  |
| --- |
| import time  def timer\_decorator(func):      def wrapper(\*args, \*\*kwargs):          start\_time = time.time()          result = func(\*args, \*\*kwargs)          end\_time = time.time()          if kwargs.get('decrypt'):              print(f"Время дешифрования {func.\_\_name\_\_}: {end\_time - start\_time} секунд")          else:              print(f"Время шифрования {func.\_\_name\_\_}: {end\_time - start\_time} секунд")          return result      return wrapper |

Листинг 4.3 – Код для подсчёта времени

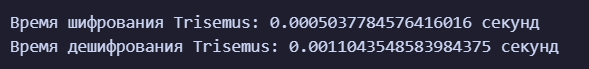


Рисунок 1.4 – Время выполнения кода

**Вывод:** в этой лабораторной работе мыизучили и приобрели практические навыки разработке и использовании приложений для реализации подстановочных шифров.