

Оглавление

Шифры моноалфавитной подстановки	2
Шифр Цезаря	3
Шифрование и расшифрование файла шифром Цезаря	4
Дешифрование текстового файла, зашифрованного шифром Цезаря	10
Дешифрование изображения, зашифрованного шифром Цезаря	12
Задание 1	13
Задание 2 (для 1 варианта)	13
Задание 3 (для 2 варианта)	13
Задание 4	13
Аффинный шифр	15
Задание 5	17
Задание 6 (вариант 1)	17
Задание 7 (вариант 2)	17
Полиалфавитные подстановки	18
Задание 8	20
Дешифрование - метод вероятных слов	20
Задание 9	21
Задание 10	21
Задание 11	22
Приложение. Форматы графических файлов	23
Литература	24

Шифры моноалфавитной подстановки

Шифр моноалфавитной подстановки - это один из самых древних шифров на Земле. Частным случаем этого шифра для шифровки секретных сообщений пользовался еще Гай Юлий Цезарь.

Рассмотрим, как используют этот шифр.

Прежде всего, выбирается *нормативный алфавит*, т.е. набор символов, которые будут использоваться при составлении сообщений, требующих зашифровки. Допустим, это будут прописные буквы русского алфавита (исключая буквы “Ё” и “Ъ”) и пробел. Таким образом, наш нормативный алфавит состоит из 32 символов. Затем выбирается *алфавит шифрования* и устанавливается взаимно однозначное соответствие между символами нормативного алфавита и символами алфавита шифрования. Алфавит шифрования может состоять из произвольных символов, в том числе и из символов нормативного алфавита.

Чтобы зашифровать исходное сообщение, каждый символ открытого текста заменяется на соответствующий ему символ алфавита шифрования.

Таблица 1

Нормативный алфавит	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	...
Алфавит шифрования	Н	К	А	Л	З	Т	П	И	О	Р	Б	Г	...

Зашифруем, например, слово “звезда”. Если использовать алфавиты, приведенные в таблице 1, то получится следующее:

Исходное сообщение: З В Е З Д А

Шифрованный текст: И А Т И З Н

Метод моноалфавитной подстановки можно представить как числовые преобразования символов исходного текста. Для этого каждой букве нормативного алфавита ставится в соответствие некоторое число, называемое *числовым эквивалентом* этой буквы. Например, для букв русского алфавита и пробела это выглядит так:

Таблица 2

Нормативный алфавит	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
Числовые эквиваленты	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Таблица 2 (продолжение)

Нормативный алфавит	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ы	Ь	Э	Ю	Я	“ – “
Числовые эквиваленты	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Шифр Цезаря

Простейшим примером моноалфавитных подстановок является шифр Цезаря. В этом шифре каждый символ открытого текста заменяется третьим после него символом в алфавите, замкнутом в кольцо, т.е. после пробела следует буква “А”. Таким образом, шифр Цезаря описывается так:

$$E_i = (M_i + S) \bmod L \quad (1)$$

где S - коэффициент сдвига, одинаковый для всех символов, M_i - числовой эквивалент символа нормативного алфавита, S_i - числовой эквивалент символа алфавита шифрования, L - количество символов в алфавите.

Цезарь использовал величину сдвига $S=3$, но, конечно, можно использовать любое целое S : $1 \leq S \leq (L-1)$.

Зашифруем, например, текст “ШИФР_ЦЕЗАРЯ”, используя коэффициент сдвига $S=2$.

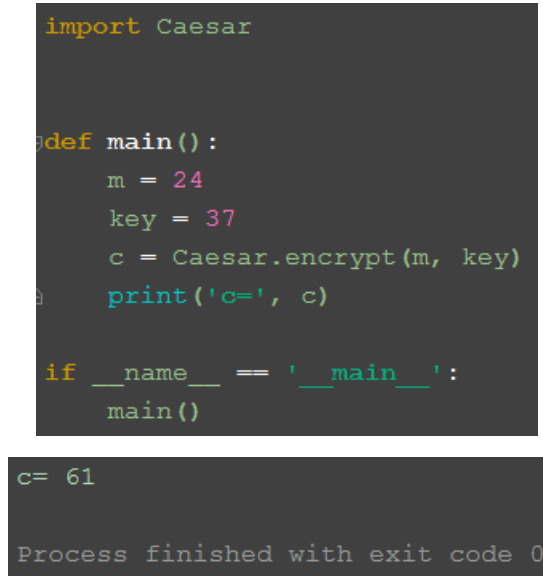
Открытый текст:	Ш	И	Ф	Р	_	Ц	Е	З	А	Р	Я
Шифрованный текст:	Ы	К	Ц	Т	Б	Ш	З	Й	В	Т	А

Шифрование и расшифрование файла шифром Цезаря

1. Создаем файл Caesar.py
2. Вставляем в Caesar.py функцию шифрования одного числа:

```
def encrypt(m, key):  
    c = (m + key) % 256  
    return c
```

3. Проверяем: создаем файл main_caesar.py (рис.1)



```
import Caesar  
  
def main():  
    m = 24  
    key = 37  
    c = Caesar.encrypt(m, key)  
    print('c=', c)  
  
if __name__ == '__main__':  
    main()  
  
c= 61  
  
Process finished with exit code 0
```

Рисунок 1 – Вызов функции encrypt

4. Вставляем в Caesar.py функцию расшифрования одного числа:

```
def decrypt(m, key):  
    c = (m - key) % 256  
    return c
```

5. Проверяем (рис.2)

```
import Caesar

def main():
    m = 24
    key = 37
    c = Caesar.encrypt(m, key)
    print('c=', c)
    m1 = Caesar.decrypt(c, key)
    print('m1=', m1)

if __name__ == '__main__':
    main()

c= 61
m1= 24

Process finished with exit code 0
```

Рисунок 2 – Вызов функции decrypt

6. Для шифрования массива (списка) чисел используем функцию:

```
def encrypt_data(data, key):
    cypher_data = []
    for m in data:
        c = encrypt(m, key)
        cypher_data.append(c)
    return cypher_data
```

7. Проверяем (рис. 3)

```
import Caesar

def main():
    m = 24
    key = 37
    c = Caesar.encrypt(m, key)
    print('c=', c)
    m1 = Caesar.decrypt(c, key)
    print('m1=', m1)
    data = [34, 67, 123, 79, 201]
    encrypt_data = Caesar.encrypt_data(data, key)
    print('encrypt_data =', encrypt_data)

if __name__ == '__main__':
    main()

c= 61
m1= 24
encrypt_data = [71, 104, 160, 116, 238]

Process finished with exit code 0
```

Рисунок 3 – Вызов функции encrypt_data

8. Для расшифрования массива (списка) чисел используем функцию:

```
def decrypt_data(data_c, key):  
    data = []  
    for c in data_c:  
        m = decrypt(c, key)  
        data.append(m)  
    return data
```

9. Проверяем (рис. 4)

```
import Caesar  
  
def main():  
    m = 24  
    key = 37  
    c = Caesar.encrypt(m, key)  
    print('c=', c)  
    m1 = Caesar.decrypt(c, key)  
    print('m1=', m1)  
    data = [34, 67, 123, 79, 201]  
    encrypt_data = Caesar.encrypt_data(data, key)  
    print('encrypt_data =', encrypt_data)  
    decrypt_data = Caesar.decrypt_data(encrypt_data, key)  
    print('decrypt_data =', decrypt_data)  
  
if __name__ == '__main__':  
    main()
```

```
c= 61  
m1= 24  
encrypt_data = [71, 104, 160, 116, 238]  
decrypt_data = [34, 67, 123, 79, 201]  
  
Process finished with exit code 0
```

Рисунок 4 – Вызов функции decrypt_data

10. Теперь массив (список) чисел формируем не сами, а берем из файла 'f1.txt' (находится в папке с заданием). В этом файле содержится текст:

He was a burly man of an exceedingly dark complexion, with an exceedingly large head and a corresponding large hand. He took my chin in his large hand and turned up my face to have a look at me by the light of the candle. He was prematurely bald on the top of his head, and had bushy black eyebrows that wouldn't lie down but stood up bristling. His eyes were set very deep in his head, and were disagreeably sharp and suspicious. He had a large watchchain, and strong black dots where his beard and whiskers would have been if he had let them. He was nothing to me, and I could have had no foresight then, that he ever would be anything to me, but it happened that I had this opportunity of observing him well.

Подключаем модуль read_write_file (рис. 5). Модуль в виде файла «read_write_file.py» также содержится в папке с заданием.

```
import Caesar
import read_write_file

def main():
    data = read_write_file.read_data_1byte('f1.txt')
    print('data=', data[0:15])

if __name__ == '__main__':
    main()
```

```
data= [72, 101, 32, 119, 97, 115, 32, 97, 32, 98, 117, 114, 108, 121, 32]
Process finished with exit code 0
```

Рисунок 5 – Побайтовое чтение файла "f1.txt" и вывод первых 15 байт

Чтоб сделать код наглядней, можно импортировать функции для чтения и записи данных из файла, например, таким образом:

```
from read_write_file import read_data_1byte as read
from read_write_file import write_data_1byte as write
```

Тогда, чтобы прочитать данные из файла достаточно написать:

```
data = read('f1.txt')
```

11. Убедимся, что полученные числа – это наш текст (рис. 6):

```
import Caesar
import read_write_file

def main():
    data = read_write_file.read_data_1byte('f1.txt')
    print('data=', data[0:15])
    txt = ''
    for n in data[0:15]:
        txt += chr(n)
    print('text=', txt)

if __name__ == '__main__':
    main()
```

```
data= [72, 101, 32, 119, 97, 115, 32, 97, 32, 98, 117, 114, 108, 121, 32]
text= He was a burly
```

Рисунок 6

12. Или более коротко (рис. 7):

```
import Caesar
import read_write_file

def main():
    data = read_write_file.read_data_1byte('f1.txt')
    print('data=', data[0:15])
    txt = ''.join([chr(s) for s in data[0:15]])
    print('text=', txt)

if __name__ == '__main__':
    main()
```

```
data= [72, 101, 32, 119, 97, 115, 32, 97, 32, 98, 117, 114, 108, 121, 32]
text= He was a burly
```

Рисунок 7

13. Теперь зашифруем весь файл. Результат шифрования запишем в файл 'fl_encrypt.txt' (Рис. 8)

```
import Caesar
import read_write_file

def main():
    data = read_write_file.read_data_1byte('f1.txt')
    print('data=', data[0:15])

    encrypt_data = Caesar.encrypt_data(data, key=67)
    print('encrypt_data=', encrypt_data[0:15])

    txt = ''.join([chr(s) for s in encrypt_data[0:15]])
    print('encrypt_text=', txt)

    read_write_file.write_data_1byte('f1_encrypt.txt', encrypt_data)

if __name__ == '__main__':
    main()

data= [72, 101, 32, 119, 97, 115, 32, 97, 32, 98, 117, 114, 108, 121, 32]
encrypt_data= [139, 168, 99, 186, 164, 182, 99, 164, 99, 165, 184, 181, 175, 188, 99]
encrypt_text= <"c?xqcxç,µ"çc

Process finished with exit code 0
```

Рисунок 8 – Шифрование файла «f1.txt»

Содержимое файла 'fl_encrypt.txt' показано на рис. 9.

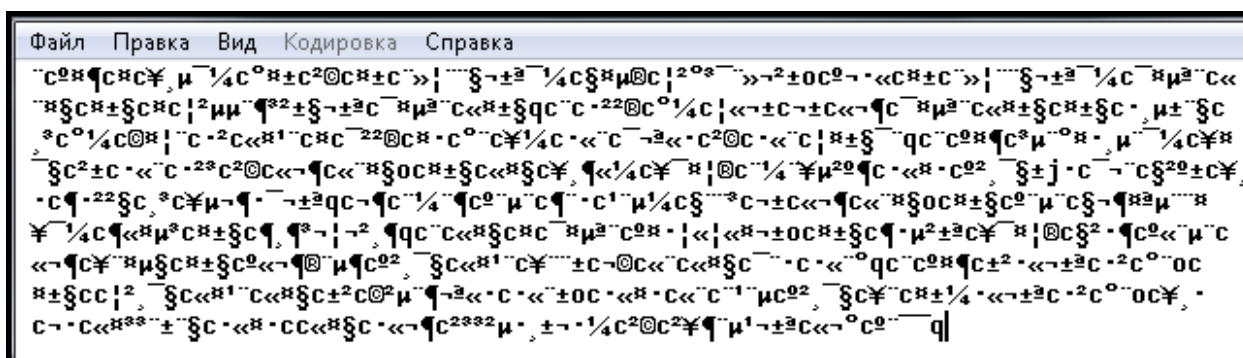


Рисунок 9 – Результат шифрования файла «f1.txt»

14. Теперь откроем зашифрованный файл 'f1_encrypt.txt', расшифруем его и результат запишем в файл 'f1_decrypt.txt' (рис. 10):

```

import Caesar
import read_write_file

def main():
    encrypt_data = read_write_file.read_data_1byte('f1_encrypt.txt')
    print('encrypt_data=', encrypt_data[0:15])

    decrypt_data = Caesar.decrypt_data(encrypt_data, key=67)
    print('decrypt_data=', decrypt_data[0:15])

    txt = ''.join([chr(s) for s in decrypt_data[0:15]])
    print('decrypt_data=', txt)

    read_write_file.write_data_1byte('f1_decrypt.txt', decrypt_data)

if __name__ == '__main__':
    main()

```

```

encrypt_data= [139, 168, 99, 186, 164, 182, 99, 164, 99, 165, 184, 181, 175, 188, 99]
decrypt_data= [72, 101, 32, 119, 97, 115, 32, 97, 32, 98, 117, 114, 108, 121, 32]
decrypt_data= He was a burly
Process finished with exit code 0

```

Рисунок 10 – Расшифрование зашифрованного файла

Исходный файл «f1.txt» и файл «f1_decrypt.txt» совпадают.

Дешифрование текстового файла, зашифрованного шифром Цезаря

Рассмотрим такую задачу: дан файл 'f1_encrypt.txt' – зашифрованное шифром Цезаря текстовое сообщение на английском языке. Ключ (т.е. значение сдвига) неизвестен. Надо дешифровать сообщение из этого файла, т.е. прочитать сообщение не зная ключа.

Последовательность действий такая: перебираем все возможные ключи (от 0 до 255), т.е. расшифровываем сообщение на этих ключах и проверяем полученное текстовое сообщение: является ли оно написанным на английском языке. Если является, то нашли ключ и расшифровали сообщение. Чтобы определить, что сообщение написано на английском языке

используем модуль detectEnglish (рис.11). Этот модуль в виде файла «detectEnglish.py» содержится в папке с заданием. Для корректной работы этого модуля необходим словарь английский слов, который в виде файла «dictionary.txt» находится в папке с заданием.

На рис.11 показан фрагмент программы, в котором выполняется расшифрование файла на очередном ключе k. Полученные данные преобразуются в текст, который с помощью функции isEnglish из модуля detectEnglish проверяется, является ли он, написанным на английском языке или нет. Как раз для этого в функции isEnglish и используется словарь английский слов (файл “dictionary.txt”).

```
import Caesar
import read_write_file
import detectEnglish

# расшифровываем
decrypt_data = Caesar.decrypt_data(encrypt_data[0:15], key=k)
# смотрим, что получилось
txt = ''.join([chr(s) for s in decrypt_data])
print('decrypt_data=', txt)
# проверяем, полученный текст - английский или нет
is_english = detectEnglish.isEnglish(txt)
```

Рисунок 11 – В переменной is_english содержится true, если текст английский, false – в противном случае (Модуль detectEnglish.py вместе со словарем должен располагаться в папке вместе с программой)

Дешифрование изображения, зашифрованного шифром Цезаря

Дан файл 'f2.png'. В нем содержится следующее изображение (рис.12):



Рисунок 12

Содержимое этого файла в шестнадцатеричном виде показано на рис.

13.

[illegible]

Рисунок 13

Видно, что первые два байта изображения имеют значения: 0x89, 0x50 в шестнадцатеричной системе счисления. Нетрудно убедиться, что у всех изображений в формате PNG первые два байта имеют данные значения. Зная эту информацию, можно выполнить дешифрование зашифрованного изображения в формате PNG.

Заголовочная информация графических файлов других типов представлена в Приложении 2.

Но прежде, зашифруем изображение, например, на ключе 143 (рис. 14):

```

import Caesar
import read_write_file
import detectEnglish

def main():
    data = read_write_file.read_data_1byte('f2.png')
    encrypt_data = Caesar.encrypt_data(data, key=143)
    read_write_file.write_data_1byte('f2_encrypt.png', encrypt_data)

```

Рисунок 14 – Шифрование изображения

После выполнения приведенного кода зашифрованное изображение будет записано в файл «f2_encrypt.png».

Задание 1

Теперь задача: есть зашифрованное изображение в файле «f2_encrypt.png». Предположим, что ключ (143) забыли. Требуется, не зная ключа, получить исходное изображение, т.е. дешифровать зашифрованное сообщение.

Задание 2 (для 1 варианта)

Дешифровать файл t3_caesar_c_all.txt.

Задание 3 (для 2 варианта)

Дешифровать файл c4_caesar_c_all.bmp. Зашифровать, оставив первые 50 байт без изменения. Сравнить с оригинальным изображением.

Задание 4

В общем случае ключом в моноалфавитном шифре является таблица замен. Таблицу замен можно сформировать следующим образом

```

k = list(range(256))
print(k)
random.shuffle(k)
print(k)

```

Само шифрование можно реализовать, например, так:

```
cypher_data = []  
for m in data:  
    c = k[m]  
    cypher_data.append(c)
```

Известна таблица замен:

```
k=[179, 109, 157, 182, 126, 141, 251, 220, 169, 237, 188, 131, 207, 22, 32, 242, 208, 68, 216, 170, 249, 199, 44,  
198, 206, 8, 148, 197, 136, 195, 159, 98, 175, 53, 123, 212, 233, 150, 6, 243, 38, 79, 156, 153, 2, 134, 47, 215, 102,  
15, 57, 110, 236, 24, 184, 72, 137, 113, 171, 70, 161, 64, 252, 247, 49, 103, 105, 138, 119, 213, 87, 130, 203, 90,  
167, 238, 231, 116, 78, 86, 173, 250, 200, 239, 178, 97, 114, 94, 166, 142, 104, 31, 75, 89, 106, 56, 128, 69, 164, 67,  
26, 228, 61, 181, 125, 227, 54, 96, 168, 107, 17, 14, 37, 190, 219, 211, 121, 112, 35, 18, 143, 158, 193, 129, 71, 23,  
101, 191, 41, 241, 82, 201, 223, 120, 59, 177, 58, 63, 151, 42, 36, 183, 226, 127, 172, 202, 84, 132, 3, 45, 73, 30,  
235, 50, 189, 4, 1, 43, 221, 205, 83, 232, 46, 147, 93, 192, 124, 244, 12, 21, 80, 55, 160, 145, 245, 209, 88, 204, 176,  
13, 253, 11, 99, 165, 140, 19, 224, 111, 27, 185, 65, 62, 16, 163, 210, 115, 217, 34, 92, 187, 152, 155, 108, 5, 122,  
229, 174, 118, 162, 95, 100, 7, 66, 29, 230, 144, 149, 52, 9, 91, 117, 214, 76, 48, 33, 194, 254, 10, 234, 218, 40, 133,  
196, 139, 135, 240, 60, 25, 225, 85, 255, 246, 51, 28, 146, 74, 222, 186, 39, 77, 0, 20, 180, 154, 81, 248]
```

Расшифровать файл c3_subst_c_all.png.

Для обратной замены можно по значению в списке определить индекс этого значения. Если есть список:

```
k=[179, 109, 157, 182, 126, 141, 251]
```

то `k.index(157)` возвратит 2, `k.index(182)` возвратит 3, `k.index(251)` возвратит 6.

Аффинный шифр

Здесь буквы исходного сообщения преобразуются следующим образом:

$$E_i = (AM_i + B) \bmod L \quad (2)$$

где A, B – целые числа, причем A и L взаимно простые (наибольший общий делитель равен 1).

Зашифруем фразу КОРАБЛИ ОТПЛЫВАЮТ ВЕЧЕРОМ, используя аффинную систему подстановок при $A=13$, $B=5$. Размер алфавита $L=32$ (будем считать, что в исходном алфавите в качестве буквы Й используется И, а в качестве Ё – Е, и добавим 32-ым символом пробел). В результате преобразований получим:

Сообщение К О Р А Б Л И О Т П Л Ы В А Ю Т В Е Ч Е Р О М
Шифртекст Ы П И Е У З О Щ П В Ъ З Ш Е Я В Щ Ж Г Ж И П Х

В нашем случае алфавит состоит из $L=256$ элементов, следовательно, надо выбрать A таким образом, чтобы $\text{НОД}(A, 256)=1$.

Алгоритм Евклида поиска наибольшего общего делителя двух чисел приведен на рис.15:

```
EUCLID(a, b)
1.  A ← a; B ← b
2.  if B = 0 return A = gcd(a, b)
3.  R = A mod B
4.  A ← B
5.  B ← R
6.  goto 2
```

Рисунок 15 – Алгоритм Евклида [2]

Пример расчета. Найти $\text{НОД}(1970, 1066)$:

1970 = 1 × 1066 + 904	gcd(1066, 904)
1066 = 1 × 904 + 162	gcd(904, 162)
904 = 5 × 162 + 94	gcd(162, 94)
162 = 1 × 94 + 68	gcd(94, 68)
94 = 1 × 68 + 26	gcd(68, 26)
68 = 2 × 26 + 16	gcd(26, 16)
26 = 1 × 16 + 10	gcd(16, 10)
16 = 1 × 10 + 6	gcd(10, 6)
10 = 1 × 6 + 4	gcd(6, 4)
6 = 1 × 4 + 2	gcd(4, 2)
4 = 2 × 2 + 0	gcd(2, 0)

Следовательно, НОД(1970, 1066) = 2.

Ниже приведена функция gcd, которая реализует алгоритм Евклида для определения НОД.

```
def gcd(a, b):
    while a != 0:
        a, b = b % a, a
    return b
```

Обратное преобразование выглядит следующим образом:

$$M_i = (E_i - B) \cdot A^{-1} \bmod L$$

где A^{-1} - обратное значение по умножению: $A \cdot A^{-1} \equiv 1 \bmod L$.

Для поиска обратного значения по умножению применяют расширенный алгоритм Евклида (рис.16).

```
EXTENDED EUCLID(m, b)
1. (A1, A2, A3) ← (1, 0, m); (B1, B2, B3) ← (0, 1, b)
2. if B3 = 0 return A3 = gcd(m, b); no inverse
3. if B3 = 1 return B3 = gcd(m, b); B2 = B-1 mod m

4.  $Q = \left\lfloor \frac{A3}{B3} \right\rfloor$ 

5. (T1, T2, T3) ← (A1 - QB1, A2 - QB2, A3 - QB3)
6. (A1, A2, A3) ← (B1, B2, B3)
7. (B1, B2, B3) ← (T1, T2, T3)
8. goto 2
```

Рисунок 16 – Расширенный алгоритм Евклида [2]

Реализация в Python:

```
def findModInverse(a, m):
    # Returns the modular inverse of a % m, which is
    # the number x such that a*x % m = 1
    if gcd(a, m) != 1:
        return None # no mod inverse if a & m aren't relatively prime
    # Calculate using the Extended Euclidean Algorithm:
    u1, u2, u3 = 1, 0, a
    v1, v2, v3 = 0, 1, m
    while v3 != 0:
        q = u3 // v3 # // is the integer division operator
        v1, v2, v3, u1, u2, u3 = (u1 - q * v1), (u2 - q * v2), (u3 - q * v3), v1, v2, v3
    return u1 % m
```

Задание 5

Расшифровать файл ff2_affine_c_all.bmp. Шифр аффинный. $a=167$, $b=35$. Зашифровать, оставив первые 50 байт без изменения. Сравнить с оригинальным изображением.

Задание 6 (вариант 1)

Дешифровать файл text10_affine_c_all.txt. Шифр аффинный. Подсчитать сколько перебрали вариантов ключей.

Задание 7 (вариант 2)

Дешифровать файл b4_affine_c_all.png. Шифр аффинный. Подсчитать сколько перебрали вариантов ключей.

Полиалфавитные подстановки

В случае моноалфавитных подстановок используется только один алфавит шифрования. Существуют шифры, где используется целый набор алфавитов шифрования. Такие шифры называются полиалфавитными и позволяют, в отличие от моноалфавитных подстановок, скрыть естественную частоту появления символов в тексте.

Простая полиалфавитная подстановка (или шифр Вижинера) последовательно и циклически меняет используемые алфавиты шифрования. Число используемых алфавитов называется периодом шифра. Для шифрования используется ключ - слово или бессмысленный набор символов нормативного алфавита. Каждая буква ключа определяет свой алфавит шифрования, который получается из нормативного циклическим сдвигом на количество символов, равное числовому эквиваленту буквы ключа (табл. 5). Очевидно, что длина ключа равна периоду шифра.

Таблица 5

Ключ	<u>А</u>	<u>Б</u>	<u>В</u>	<u>Г</u>	<u>Д</u>	...	<u>Э</u>	<u>Ю</u>	<u>Я</u>
0	А	Б	В	Г	Д	...	Э	Ю	Я
1	Б	В	Г	Д	Е	...	Ю	Я	А
2	В	Г	Д	Е	Ж	...	Я	А	Б
3	Г	Д	Е	Ж	З	...	А	Б	В
...
30	Ю	Я	А	Б	В	...	Ы	Ь	Э
31	Я	А	Б	В	Г	...	Ь	Э	Ю

Чтобы зашифровать сообщение шифром Вижинера, поступают следующим образом. Под каждой буквой открытого текста помещается буква ключа. Ключ циклически повторяется необходимое число раз. Чтобы вычислить числовой эквивалент буквы шифртекста, числовой эквивалент буквы ключа складывается по модулю L с числовым эквивалентом буквы открытого текста, где L - мощность нормативного алфавита. Т.е. шифр Вижинера описывается следующим выражением:

$$E_i = (M_i + K_{i \bmod U}) \bmod L \quad (3)$$

где

- E_i, M_i - числовые эквиваленты символов криптограммы и открытого текста соответственно,
 $K_{i \bmod U}$ - числовой эквивалент буквы ключа,
 L - мощность нормативного алфавита.
 U - длина ключа или период шифра

Буквы ключа определяют величину смещения символов криптограммы относительно символов открытого текста.

Зашифруем, например, текст “полиалфавитная_подстановка” с ключом “краб”. Будем использовать алфавит, приведенный в табл. 6. Процесс шифрования приведен в табл. 7.

Таблица 6

Нормативный Алфавит	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
Числовые Эквиваленты	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Таблица 6 (продолжение)

Нормативный Алфавит	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ы	Ь	Э	Ю	Я	“ _ ”
Числовые Эквиваленты	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Таблица 7

П	15	К	10	$(15 + 10) \bmod 32$	25	Щ
О	14	Р	16	$(14 + 16) \bmod 32$	30	Я
Л	11	А	0	$(11 + 0) \bmod 32$	11	Л
И	8	Б	1	$(8 + 1) \bmod 32$	9	Й
А	0	К	10	$(0 + 10) \bmod 32$	10	К
Л	11	Р	16	$(11 + 16) \bmod 32$	27	Ь
Ф	20	А	0	$(20 + 0) \bmod 32$	20	Ф
А	0	Б	1	$(0 + 1) \bmod 32$	1	Б
В	2	К	10	$(2 + 10) \bmod 32$	12	М
И	8	Р	16	$(8 + 16) \bmod 32$	24	Ш
Т	18	А	0	$(18 + 0) \bmod 32$	18	Т
Н	13	Б	1	$(13 + 1) \bmod 32$	14	О
А	0	К	10	$(0 + 10) \bmod 32$	10	К
Я	30	Р	16	$(30 + 16) \bmod 32$	14	О
_	31	А	0	$(31 + 0) \bmod 32$	31	_
П	15	Б	1	$(15 + 1) \bmod 32$	16	Р
О	14	К	10	$(14 + 10) \bmod 32$	24	Ш

Д	4	Р	16	$(4 + 16) \bmod 32$	20	Ф
С	17	А	0	$(17 + 0) \bmod 32$	17	С
Т	18	Б	1	$(18 + 1) \bmod 32$	19	У
А	0	К	10	$(0 + 10) \bmod 32$	10	К
Н	13	Р	16	$(13 + 16) \bmod 32$	29	Ю
О	14	А	0	$(14 + 0) \bmod 32$	14	О
В	2	Б	1	$(2 + 1) \bmod 32$	3	Г
К	10	К	10	$(10 + 10) \bmod 32$	20	Ф
А	0	Р	16	$(0 + 16) \bmod 32$	16	Р

В результате получилась криптограмма: “ЩЯЛЙКЪФБМШТОКО_РШФСУКЮОГФР”.

Задание 8

Расшифровать файл im6_vigener_c_all.bmp. Шифр Виженера. Ключ: magistr. Зашифровать, оставив первые 50 байт без изменения. Сравнить с оригинальным изображением.

Преобразовать ключ в численное значение:

```
key = 'magistr'
k = [ord(s) for s in key]
```

Дешифрование - метод вероятных слов

Метод вероятных слов основан на том, что чаще всего заранее известна область применения криптограммы, а, значит, и слова, которые могут встретиться в тексте. Например, если известно, что в криптограмме зашифрован финансовый отчет, вероятно, в тексте встречаются слова “дебет”, “кредит”, “баланс” и т.п.

Т.к. в полиалфавитных подстановках криптограмма является суммой открытого текста и ключа по модулю L , то, чтобы проверить наличие вероятного слова в тексте, необходимо вычесть его из криптограммы по модулю L во всех возможных позициях, где L - мощность исходного алфавита.

Если данное вероятное слово присутствует в тексте и вычитается в правильной позиции, то результатом такого вычитания будет ключ шифрования или его часть. Если слово испытывается в неправильной позиции, то результатом вычитания будет бессмысленный набор букв. Понятно, что, если этого слова нет в тексте, все позиции будут неправильными.

Полученная в результате вычитания хотя бы часть осмысленного слова - показатель успеха. Далее надо попытаться расширить открытый текст или ключ в этом направлении.

Конечно, будут возникать и “ложные тревоги”, особенно в случае коротких вероятных слов, но эти варианты будут легко отбрасываться в процессе продвижения анализа, т.к. они не дадут разумных расширений.

Задание 9

Дешифровать `text4_vigener_c.txt` – зашифрованное шифром Вижинера текстовое сообщение. Известно, что сообщение начинается со строки из одних пробелов.

Задание 10

Дешифровать этот же файл (`text4_vigener_c.txt`) – зашифрованное шифром Вижинера текстовое сообщение. Известно, что в сообщении присутствует слово `housewives`.

Надо взять первые `len(housewives)` значений зашифрованных данных. Вычесть из них численные значения `housewives`. Полученные 10 значений представить в виде текста и напечатать. Взять `len(housewives)` значений зашифрованных данных, но уже не с самого начала, а выполнить сдвиг на одно значение. Вычесть, распечатать. Взять `len(housewives)` значений после

сдвига на два значения и т.д. Просмотреть напечатанные сообщения на предмет поиска ключевого выражение – оно должно читаться.

```

+%  0  0+0/"
, &00) 0*000
-000/0+000
000!.0,0 0
000 /000)!
00/
,$0" 00%0.!
+%  0  0+0/"
, &00) 0*000
-000/0+000
000!.0,0 0
000 /000)!
00/
,$0" 00%000
+%  0  0+
0N
, &00) 000\`
-000/  0Kn`
000!00X]nb
0Jj]p!
&%00\\j_/w
,0  Nn\l0..b
00L`n^+tpn
0R^`p0 _|\
Yd^b/slkje
kd`!..^xYs0
kf0wpjfb)b
m%ub|Xo0pt
f`x0p0t

```

Задание 11

Дешифровать файл (text1_vigener_c.txt) – зашифрованное шифром Вижинера текстовое сообщение. Известно, что в сообщении присутствуют слова it therefore.

Приложение. Форматы графических файлов

BMP

00000000:	42	4D	7E	1B	00	00	00	00	00	00	3E	00	00	00	28	00		ВМ~.....>...(. 00000010:	00	00	E6	00	00	00	DA	00	00	00	01	00	01	00	00	00		..ж...ь..... 00000020:	00	00	40	1B	00	00	C4	0E	00	00	C4	0E	00	00	00	00		..@...Д...Д.... 00000030:	00	00	00	00	00	00	00	00	00	00	FF	FF	FF	00	FF	FF	яяя.яя 00000040:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	F0	7F	FF		яяяяяяяяяяяяяя■я 00000050:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FC	00	00	00	7F	FF			яяяяяяяяяяъ...яя 00000060:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			яяяяяяяяяяяяяя■я
-----------	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	--	-----------------------------	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	--	---------------------------	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	--	------------------------------	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	--	--------------------------	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	--	-------------------------------	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	--	--	-------------------------------	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	--	--	------------------

PNG

Файл	Правка	Вид	Кодировка	Справка
00000000:	89 50 4E 47 0D 0A 1A 0A	00 00 00 0D 49 48 44 52		%PNG.....IHDR
00000010:	00 00 00 6F 00 00 00 7D	08 06 00 00 00 AC DB 3D		...o...}.....w=
00000020:	F0 00 00 00 01 73 52 47	42 00 AE CE 1C E9 00 00		p.....sRGB.@.й..
00000030:	00 04 67 41 4D 41 00 00	B1 8F 0B FC 61 05 00 00		..gAMA...t.ьa...
00000040:	00 09 70 48 59 73 00 00	0E C3 00 00 0E C3 01 C7		..pHYs...Г...Г.3
00000050:	6F A8 64 00 00 60 5D 49	44 41 54 78 5E ED BD 05		oEd...`]IDATx^NS.
00000060:	78 1D 47 B6 35 9A EF BD	7B FF FB CF C5 E1 99 CC		x.G45nS{yмПЕб™M

JPG

Файл	Правка	Вид	Кодировка	Справка
000000000:	FF D8 FF E0 00 10 4A 46	49 46 00 01 00 01 00 60		Имя..JFIF.....`
000000010:	00 60 00 00 FF FE 00 1F	4C 45 41 44 20 54 65 63		...я..LEAD Tec
000000020:	68 6E 6F 6C 6F 67 69 65	73 20 49 6E 63 2E 20 56		hnologies Inc. U
000000030:	31 2E 30 31 00 FF DB 00	84 00 05 05 05 08 05 08		1.01.я.,,.....
000000040:	0C 07 07 0C 0C 09 09 09	0C 0D 0C 0C 0C 0C 0D 0D	
000000050:	0D 0D 0D 0D 0D 0D 0D 0D	0D 0D 0D 0D 0D 0D 0D 0D	
000000060:	0D 0D 0D 0D 0D 0D 0D 0D	0D 0D 0D 0D 0D 0D 0D 0D	
000000070:	0D 0D 0D 0D 0D 0D 0D 0D	0D 0D 01 05 08 08 0A 07	

TIFF

Файл	Правка	Вид	Кодировка	Справка
00000000:	49	49	2A 00 08 00 00 00 00	11 00 FE 00 04 00 01 00 II*.....ю.....
00000010:	00	00	02 00 00 00 00 00 01	04 00 01 00 00 00 BE 00 S.
00000020:	00	00	01 01 04 00 01 00 00	00 00 AE 00 00 00 02 01 ®.....
00000030:	03	00	04 00 00 00 DA 00 00	00 00 03 01 03 00 01 00 b.....
00000040:	00	00	01 00 00 00 06 01 00	03 00 01 00 00 00 02 00 T.....
00000050:	00	00	11 01 04 00 01 00 00	00 00 F2 00 00 00 12 01 T.....
00000060:	03	00	01 00 00 00 01 00 00	00 00 15 01 03 00 01 00 ®.....
00000070:	00	00	04 00 00 00 16 01 00	04 00 01 00 00 00 AE 00 ®.....

Литература

- [1] Al Sweigart. Hacking Secret Ciphers with Python
- [2] Stallings W, “Cryptography And Network Security. Principles And Practice”, 5th Edition, 2011.