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.

Первая работа посвящена изучению моноалфавитных подстановок. 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 руского 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 1.1

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 1.1, 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 1.2

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 1.2 (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 :

$Ei = (Mi + S) \bmod L$, где (1.2)

S - коэффициент сдвига, одинаковый для всех символов.

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

Зашифруем, например, текст “ШИФР_ЦЕЗАРЯ” , использовав 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

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
```

Рис. 1.

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

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

4 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
```

Рис. 2.

5. 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
```

6. 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
```

Рис. 3.

7. 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
```

8. 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
```

Рис. 4.

9. Теперь массив (список) чисел формируем не сами, а берем из файла.

Формируем файл ‘f1.txt’. В него заносим следующую информацию:

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):

```
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
```

Рис. 5.

10. 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
```

Рис. 6.

11. 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
```

Рис. 7.

12. Теперь зашифруем весь файл. Результат шифрования запишем в файл ‘f1_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= ‹¨c²¤¶c¤c¥,µ¯¾c

Process finished with exit code 0
```

Рис.8.

Содержимое файла ‘f1_encrypt.txt’ показано на рис. 9.

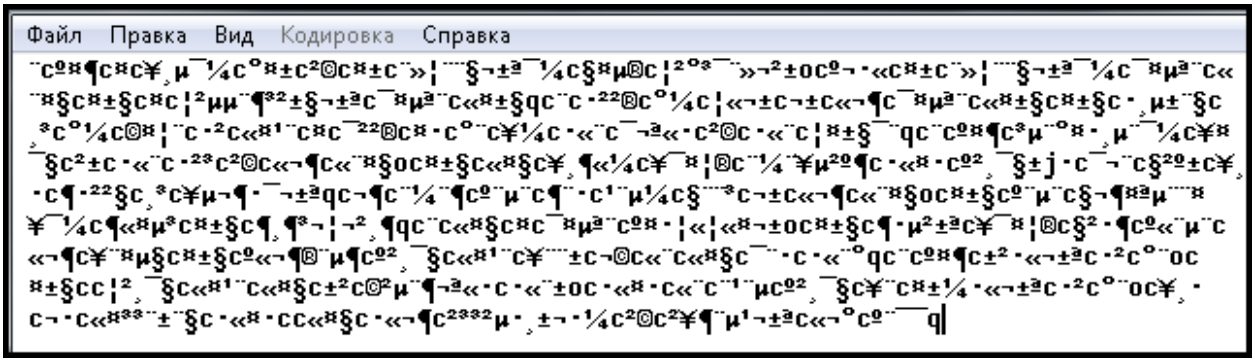

Рис. 9.

13. 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
```

Рис. 10.

Исходный файл ‘f1.txt’ и ‘f1_decrypt.txt’ совпадают.

14. Рассмотрим такую задачу: дан файл ‘f1_encrypt.txt’ – зашифрованное шифром Цезаря текстовое сообщение. Ключ неизвестен. Надо дешифровать сообщение из этого файла.

Последовательность действий такая: перебираем все возможные ключи (от 0 до 255), расшифровываем сообщение на этих ключах, проверяем полученное текстовое сообщение: является оно написанным на английском языке, если является, то нашли ключ и расшифровываем сообщение. Чтобы определить, что сообщение написано на английском языке используем модуль detectEnglish (рис. 11):

```
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)
```

Рис. 11. В is_english содержится true, если текст английский, false – в противном случае

Модуль detectEnglish.py вместе со словарем (файл dictionary.txt) должен быть в папке вместе с программой.

15. Дан файл ‘f2.png’. В нем содержится следующее изображение (рис. 12):

Рис. 12.

Содержимое этого файла в шестнадцатеричном виде показано на рис. 13:

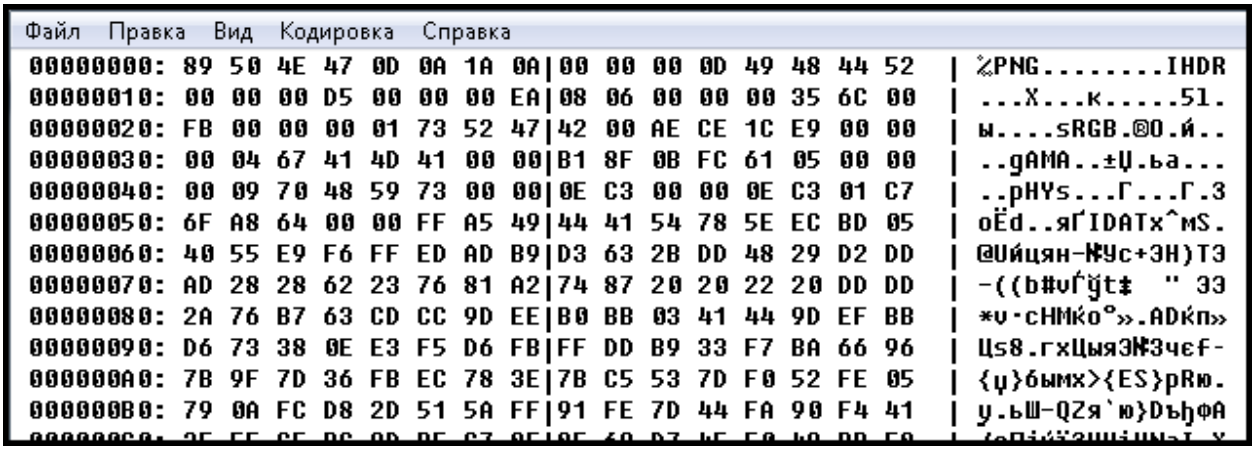

Рис. 13

Видно, что первые два байта изображения имеют значения: 0x89, 0x50. 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.

Но прежде, зашифруем изображение (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)
```

Рис. 14.

Теперь задача: дан файл ‘f2_encrypt.png’. Считаем, что ключ не известен. Дешифровать.

16. Дешифровать файл t3_caesar_c_all.txt.

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

18. 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)
```

В модуле substitution.py реализовать функции шифрования и расшифрования данных. 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. Зашифровать, оставив первые 350 байт без изменения. Сравнить с оригинальным изображением.

19. **Афинный шифр**

Здесь буквы исходного сообщения преобразуются следующим образом:

$$Ei = (AMi + B) \bmod L \qquad (1.3)$$

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

Пример.

Зашифруем фразу КОРАБЛИ ОТПЛЫВАЮТ ВЕЧЕРОМ, используя афинную 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 Ь 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 НОД(A, 256)=1. Ниже приведена функция gcd, которая реализует известный алгоритм Евклида для определения НОД.

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

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

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

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

22. 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. 1). Очевидно, что длина ключа равна периоду шифра.

Таблица 1

Ключ	А	Б	В	Г	Д	…	Э	Ю	Я
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	Ъ	Э
31	Я	А	Б	В	Г	…	Ъ	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 :

$$E_i = (M_i + K_{i \bmod U}) \bmod L, \qquad (1)$$

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” ключом “краб”. Будем использовать алфавит, приведенный в таблице 2. Процесс шифрования приведен в таблице 3.

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

Таблица 3 .

П	**15**	**К**	**10**	**(15 + 10) mod 32**	**25**	**Щ**
О	**14**	**Р**	**16**	**(14 + 16) mod 32**	**30**	**Я**
Л	**11**	**А**	**0**	**(11 + 0) mod 32**	**11**	**Л**
И	**8**	**Б**	**1**	**(8 + 1) mod 32**	**9**	**Й**
А	**0**	**К**	**10**	**(0 + 10) mod 32**	**10**	**К**
Л	**11**	**Р**	**16**	**(11 + 16) mod 32**	**27**	**Ь**
Ф	**20**	**А**	**0**	**(20 + 0) mod 32**	**20**	**Ф**
А	**0**	**Б**	**1**	**(0 + 1) mod 32**	**1**	**Б**
В	**2**	**К**	**10**	**(2 + 10) mod 32**	**12**	**М**
И	**8**	**Р**	**16**	**(8 + 16) mod 32**	**24**	**Ш**
Т	**18**	**А**	**0**	**(18 + 0) mod 32**	**18**	**Т**
Н	**13**	**Б**	**1**	**(13 + 1) mod 32**	**14**	**О**
А	**0**	**К**	**10**	**(0 + 10) mod 32**	**10**	**К**
Я	**30**	**Р**	**16**	**(30 + 16) mod 32**	**14**	**О**
_	**31**	**А**	**0**	**(31 + 0) mod 32**	**31**	**_**
П	**15**	**Б**	**1**	**(15 + 1) mod 32**	**16**	**Р**
О	**14**	**К**	**10**	**(14 + 10) mod 32**	**24**	**Ш**
Д	**4**	**Р**	**16**	**(4 + 16) mod 32**	**20**	**Ф**
С	**17**	**А**	**0**	**(17 + 0) mod 32**	**17**	**С**
Т	**18**	**Б**	**1**	**(18 + 1) mod 32**	**19**	**У**
А	**0**	**К**	**10**	**(0 + 10) mod 32**	**10**	**К**
Н	**13**	**Р**	**16**	**(13 + 16) mod 32**	**29**	**Ю**
О	**14**	**А**	**0**	**(14 + 0) mod 32**	**14**	**О**
В	**2**	**Б**	**1**	**(2 + 1) mod 32**	**3**	**Г**
К	**10**	**К**	**10**	**(10 + 10) mod 32**	**20**	**Ф**
А	**0**	**Р**	**16**	**(0 + 16) mod 32**	**16**	**Р**

В результате получилась криптограмма: “ЩЯЛЙКЬФБМШТОКО_Р ШФСУКЮОГФР”.

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

23. 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.

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

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

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

26. Шифры перестановки

Шифры замены относятся к т.н. поточным шифрам. В поточных шифрах шифрование происходит посимвольно. Если уподобить поточные шифры - а,вернее, шифраторы - черному ящику и подавать на вход поток символов открытого текста, то на выходе практически без задержек будут появляться соответствующие им символы криптограммы.

Кроме поточных, существуют еще блочные шифры. При использовании блочных шифров текст предварительно разбивается на блоки и шифрование происходит поблочно.

Перестановки относятся к блочным шифрам. Текст делится на блоки по N символов, и в каждом блоке символы переставляются в соответствии с некоторым правилом (ключом). Таким образом ключ задает порядок символов в блоке.

Кроме того, известны т.н. табличные перестановки. Например, исходный текст вписывают в таблицу по столбцам, а затем строки таблицы переставляются в соответствии с ключом. Размер таблицы оговаривается заранее.

1. Использование подстановки

Существует два способа использования ключа.

Пусть

Ключ K = $k_1 k_2 \ldots k_i \ldots k_N$

Текст M = $m_1 m_2 \ldots m_i \ldots m_N$

k_i- целое число : $k_i \in [1,N]$.

Способ 1.

Чтобы зашифровать M, нужно i-ый символ текста поставить на k_i-ое место. При расшифровке на i-ое место ставим k_i-ый символ криптограммы.

N = 5; Ключ 3-1-5-4-2

Исходный Текст : ИНФОР | МАЦИЯ

Криптограмма : НРИОФ | АЯМИЦ

Способ 2.

При шифровке на i-ое место ставим k_i-ый символ текста. При расшифровке i-ый символ криптограммы ставится на k_i-ое место.

N = 5; Ключ 3-1-5-4-2

Исходный Текст : ИНФОР | МАЦИЯ

Криптограмма : ФИРОН | ЦМЯИА

Такая таблица называется подстановкой степени n.

Зная подстановку, задающую преобразование, можно осуществить как зашифрование, так и расшифрование текста.

Дешифровать text2_permutation_c.txt – зашифрованное шифром перестановки текстовое сообщение. Известно, что длина перестановки 6. Правило перестановки: на i-ю позицию ставится п(i) символ, где п – перестановка.