МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

**ЛАБОРАТОРНЫЕ РАБОТЫ ПО ПРЕДМЕТУ**

«Программирование криптографических алгоритмов»

**Выполнил:**

Барышников С.С. гр. 191-351

**Преподаватель:**

Бутакова Н.Г.

Москва 2021 г.

**Содержание**

[Аннотация 3](#_Toc84365951)

[Постоянный модуль 4](#_Toc84365952)

[**E: ШИФРЫ ГАММИРОВАНИЯ** 5](#_Toc84365953)

[13. Одноразовый блокнот К.Шеннона 5](#_Toc84365962)

[14. Гаммирование ГОСТ 28147-89 10](#_Toc84365963)

# Аннотация

**Среда программирования:** Visual Studio Code

**Язык программирования:** Python 3

**Процедуры для запуска программы:** $ python3 <имя\_файла>.py

**Пословица-тест:** Время, приливы и отливы не ждут человека.

**Текст для проверки работы:** Вот пример статьи на тысячу символов. Это достаточно маленький текст, оптимально подходящий для карточек товаров в интернет или магазинах или для небольших информационных публикаций. В таком тексте редко бывает более двух или трёх абзацев и обычно один подзаголовок. Но можно и без него. На тысячу символов рекомендовано использовать один или два ключа и одну картину. Текст на тысячу символов это сколько примерно слов? Статистика показывает, что тысяча включает в себя сто пятьдесят или двести слов средней величины. Но, если злоупотреблять предлогами, союзами и другими частями речи на один или два символа, то количество слов неизменно возрастает. В копирайтерской деятельности принято считать тысячи с пробелами или без. Учет пробелов увеличивает объем текста примерно на сто или двести символов именно столько раз мы разделяем слова свободным пространством. Считать пробелы заказчики не любят, так как это пустое место. Однако некоторые фирмы и биржи видят справедливым ставить стоимость за тысячу символов с пробелами, считая последние важным элементом качественного восприятия. Согласитесь, читать слитный текст без единого пропуска, никто не будет. Но большинству нужна цена за тысячу знаков без пробелов.

**Интерфейс:** #в разработке#

# Постоянный модуль

Код модуля base.py используемый для предотвращения дублирования кода, используется во всех последующих программах:

import re

alphabet = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя"

dict = {'.': 'тчк', ',': 'зпт'}

def replace\_all\_to(input\_text, dict):

    input\_text = input\_text.replace(' ', '')

    for i, j in dict.items():

        input\_text = input\_text.replace(i, j)

    return input\_text

def replace\_all\_from(input\_text, dict):

    for i, j in dict.items():

        input\_text = input\_text.replace(j, i)

    return input\_text

def file\_to\_string(name):

    with open(name) as f:

        input\_short\_text = " ".join([l.rstrip() for l in f]) + ' '

    return input\_short\_text.lower()

def input\_for\_cipher\_short():

    return replace\_all\_to(file\_to\_string('short.txt'), dict)

def input\_for\_cipher\_long():

    return replace\_all\_to(file\_to\_string('long.txt'), dict)

def output\_from\_decrypted(decrypted\_text):

    return replace\_all\_from(decrypted\_text, dict)

**E: ШИФРЫ ГАММИРОВАНИЯ**



## 

## Одноразовый блокнот К.Шеннона

Популярность поточных шифров можно связывать с работой Клода Шеннона, посвященной анализу одноразовых гамма-блокнотов. Название «одноразовый блокнот» стало общепринятым в годы Второй мировой войны, когда для шифрования широко использовались бумажные блокноты.

Одноразовый блокнот использует длинную шифрующую последовательность, которая состоит из случайно выбираемых бит или наборов бит (символов). Шифрующая последовательность побитно или посимвольно накладывается на открытый текст, имеет ту же самую длину, что и открытое сообщение, и может использоваться только один раз (о чем свидетельствует само название шифрсистемы); ясно, что при таком способе шифрования требуется огромное количество шифрующей гаммы.

Открытый текст сообщения ш записывают как последовательность бит или символов m = momi...mn\_i, а двоичную или символьную шифрующую последовательность к той же самой длины - как k = koki...k„\_|.

Шифртекст с = c0cl...cn.i определяется соотношением Cj = mi Шк, при 0

**Код программы:**

import random

from base import alphabet, input\_for\_cipher\_short, input\_for\_cipher\_long, output\_from\_decrypted

alphabet = alphabet.replace(' ', '')

alphabet\_lower = {}

i = 0

while i < (len(alphabet)):

    alphabet\_lower.update({alphabet[i]: i})

    i += 1

def get\_key(d, value):

    for k, v in d.items():

        if v == value:

            return k

def shenon\_encode(msg):

    msg\_list = list(msg)

    msg\_list\_len = len(msg\_list)

    msg\_code\_bin\_list = list()

    for i in range(len(msg\_list)):

        msg\_code\_bin\_list.append(alphabet\_lower.get(msg\_list[i]))

    key\_list = list()

    for i in range(msg\_list\_len):

        key\_list.append(random.randint(0, 32))

    cipher\_list = list()

    for i in range(msg\_list\_len):

        m = int(msg\_code\_bin\_list[i])

        k = int(key\_list[i])

        cipher\_list.append(int(bin(m ^ k), base=2))

    return cipher\_list, key\_list

def shenon\_decode(msg, key\_list):

    decipher\_list = list()

    msg\_list\_len = len(msg)

    for i in range(msg\_list\_len):

        c = int(msg[i])

        k = int(key\_list[i])

        decipher\_list.append(int(bin(c ^ k), base=2))

    deciphered\_str = ""

    for i in range(len(decipher\_list)):

        deciphered\_str += get\_key(alphabet\_lower, decipher\_list[i])

    return deciphered\_str

short\_encoded = shenon\_encode(input\_for\_cipher\_short())

short\_decoded = shenon\_decode(short\_encoded[0], short\_encoded[1])

long\_encoded = shenon\_encode(input\_for\_cipher\_long())

long\_decoded = shenon\_decode(long\_encoded[0], long\_encoded[1])

print(f'''

Одноразовый блокнот:

КОРОТКИЙ ТЕКСТ:

Зашифрованный текст:

{short\_encoded[0]}

Ключ:

{short\_encoded[1]}

Расшифрованный текст:

{output\_from\_decrypted(short\_decoded)}

ДЛИННЫЙ ТЕКСТ:

Зашифрованный текст:

{long\_encoded[0]}

Ключ:

{long\_encoded[1]}

Расшифрованный текст:

{output\_from\_decrypted(long\_decoded)}

''')

**Тестирование:**

/bin/python3 /root/mospolytech-education-crypt-dev-2021-1/lab05\_13\_shenon.py

Одноразовый блокнот:

КОРОТКИЙ ТЕКСТ:

Зашифрованный текст:

[20, 24, 3, 1, 54, 21, 5, 27, 10, 6, 19, 22, 12, 23, 30, 22, 12, 20, 31, 13, 30, 6, 17, 14, 21, 0, 0, 4, 20, 10, 7, 10, 0, 3, 15, 9, 26, 31, 26]

Ключ:

[22, 9, 6, 12, 22, 29, 21, 8, 26, 23, 26, 26, 5, 21, 2, 31, 3, 7, 19, 4, 28, 26, 31, 11, 18, 4, 20, 23, 12, 15, 11, 5, 2, 6, 4, 9, 9, 7, 17]

Расшифрованный текст:

время,приливыиотливынеждутчеловека.

ДЛИННЫЙ ТЕКСТ:

Зашифрованный текст:

[12, 47, 30, 29, 1, 28, 12, 13, 17, 28, 14, 8, 8, 13, 23, 25, 7, 14, 25, 7, 49, 56, 27, 11, 13, 16, 10, 24, 9, 14, 3, 26, 22, 6, 9, 6, 7, 0, 6, 5, 7, 31, 7, 12, 12, 11, 1, 15, 32, 25, 24, 4, 31, 17, 7, 9, 2, 8, 23, 9, 17, 17, 48, 4, 3, 19, 1, 10, 1, 4, 27, 9, 25, 23, 15, 17, 29, 12, 13, 3, 47, 21, 22, 12, 19, 19, 58, 10, 8, 12, 11, 14, 31, 31, 31, 9, 6, 18, 1, 49, 3, 10, 27, 3, 19, 21, 26, 16, 13, 2, 5, 21, 15, 26, 5, 5, 5, 32, 29, 28, 9, 14, 25, 3, 22, 15, 6, 14, 34, 14, 7, 15, 18, 0, 29, 1, 27, 14, 12, 46, 16, 27, 2, 4, 27, 16, 16, 10, 12, 0, 21, 1, 28, 24, 17, 1, 8, 27, 19, 13, 11, 16, 20, 9, 1, 34, 23, 19, 25, 7, 20, 11, 1, 14, 23, 9, 10, 21, 4, 10, 29, 6, 3, 6, 6, 9, 23, 0, 4, 0, 14, 14, 9, 12, 15, 1, 20, 14, 27, 5, 20, 0, 13, 13, 25, 27, 1, 7, 7, 2, 15, 6, 7, 27, 10, 23, 2, 9, 31, 21, 22, 25, 22, 23, 36, 23, 31, 10, 18, 8, 19, 28, 21, 8, 4, 23, 1, 22, 29, 31, 28, 27, 31, 22, 21, 3, 23, 30, 7, 11, 0, 18, 31, 20, 4, 20, 25, 27, 19, 8, 45, 26, 9, 14, 7, 16, 20, 15, 23, 26, 11, 16, 29, 23, 7, 24, 8, 4, 4, 9, 9, 14, 22, 19, 15, 23, 3, 2, 15, 9, 19, 1, 9, 11, 9, 17, 14, 27, 12, 25, 7, 21, 2, 8, 31, 13, 23, 44, 13, 5, 9, 29, 23, 20, 11, 25, 2, 5, 2, 4, 26, 23, 20, 27, 9, 27, 30, 24, 14, 7, 17, 23, 4, 0, 13, 17, 33, 22, 10, 20, 9, 13, 24, 16, 27, 1, 3, 11, 21, 9, 17, 20, 17, 20, 19, 26, 21, 24, 1, 17, 0, 9, 22, 9, 19, 4, 44, 22, 25, 28, 26, 28, 5, 29, 12, 3, 12, 8, 22, 18, 5, 7, 14, 8, 27, 16, 12, 15, 29, 30, 6, 20, 17, 29, 15, 14, 18, 24, 7, 22, 28, 38, 15, 28, 6, 16, 4, 6, 21, 23, 9, 22, 9, 1, 18, 26, 52, 22, 19, 21, 25, 37, 15, 28, 31, 8, 30, 36, 19, 17, 14, 12, 26, 5, 4, 30, 6, 0, 5, 25, 0, 14, 25, 27, 1, 36, 14, 15, 6, 11, 31, 13, 17, 1, 31, 27, 28, 7, 12, 22, 8, 28, 17, 22, 16, 25, 25, 28, 21, 29, 13, 0, 20, 17, 13, 23, 23, 37, 15, 8, 58, 4, 8, 9, 27, 7, 5, 19, 21, 13, 21, 25, 24, 19, 4, 2, 26, 25, 12, 11, 24, 26, 8, 19, 11, 11, 31, 35, 1, 9, 26, 22, 2, 25, 26, 60, 24, 2, 49, 15, 22, 27, 14, 3, 29, 12, 7, 13, 7, 31, 2, 12, 22, 13, 14, 17, 5, 6, 31, 21, 21, 14, 14, 25, 4, 7, 30, 25, 27, 10, 8, 16, 7, 14, 6, 29, 29, 17, 2, 31, 15, 12, 14, 19, 17, 31, 21, 25, 16, 10, 26, 19, 9, 31, 22, 15, 32, 18, 1, 27, 3, 0, 17, 5, 47, 10, 11, 16, 4, 29, 4, 4, 17, 21, 10, 14, 10, 10, 13, 50, 15, 15, 2, 19, 15, 10, 18, 7, 29, 48, 13, 15, 13, 42, 7, 1, 13, 7, 0, 27, 4, 0, 16, 13, 17, 4, 34, 1, 12, 7, 6, 8, 19, 13, 10, 9, 22, 25, 10, 23, 12, 23, 30, 29, 40, 10, 16, 2, 16, 2, 7, 13, 31, 27, 15, 16, 11, 0, 19, 4, 10, 31, 26, 16, 24, 56, 2, 18, 8, 0, 30, 24, 5, 11, 0, 6, 31, 19, 10, 31, 8, 23, 5, 24, 26, 6, 30, 11, 13, 25, 16, 17, 13, 6, 30, 21, 15, 7, 1, 9, 2, 15, 12, 14, 0, 10, 6, 31, 5, 29, 26, 5, 30, 12, 28, 4, 18, 8, 1, 13, 24, 24, 7, 8, 21, 27, 25, 17, 18, 2, 27, 16, 3, 4, 12, 22, 45, 15, 0, 10, 14, 29, 34, 31, 12, 22, 31, 21, 26, 2, 16, 16, 11, 15, 0, 6, 23, 26, 25, 2, 2, 28, 16, 3, 24, 2, 25, 17, 26, 10, 26, 16, 21, 25, 27, 4, 18, 21, 23, 6, 0, 18, 30, 6, 24, 17, 1, 16, 25, 22, 20, 15, 30, 37, 29, 4, 21, 54, 6, 18, 3, 8, 6, 12, 23, 27, 15, 27, 4, 51, 29, 30, 7, 14, 31, 5, 0, 18, 18, 29, 31, 5, 27, 19, 21, 30, 4, 4, 21, 31, 31, 46, 3, 16, 18, 4, 12, 25, 5, 22, 28, 7, 11, 9, 29, 23, 29, 22, 11, 12, 10, 7, 24, 29, 35, 7, 13, 10, 12, 12, 17, 17, 1, 14, 17, 22, 19, 21, 20, 16, 0, 20, 17, 51, 13, 30, 10, 24, 2, 25, 20, 21, 22, 17, 13, 23, 13, 25, 29, 41, 25, 20, 26, 23, 25, 12, 20, 1, 26, 34, 11, 16, 20, 29, 20, 17, 44, 5, 21, 13, 27, 7, 17, 50, 21, 0, 13, 31, 53, 31, 27, 21, 19, 26, 24, 17, 41, 5, 30, 21, 8, 14, 16, 45, 3, 22, 37, 17, 21, 20, 25, 12, 45, 26, 14, 14, 1, 50, 18, 21, 22, 5, 18, 24, 17, 24, 13, 28, 21, 10, 16, 6, 52, 1, 30, 35, 17, 23, 2, 25, 1, 20, 19, 9, 20, 7, 18, 26, 3, 9, 11, 31, 11, 22, 26, 7, 11, 15, 16, 31, 20, 8, 11, 20, 11, 5, 0, 27, 24, 1, 9, 30, 22, 23, 14, 7, 41, 28, 31, 16, 30, 21, 10, 28, 19, 10, 14, 25, 10, 2, 19, 21, 27, 0, 11, 26, 26, 27, 30, 12, 21, 4, 10, 0, 10, 25, 13, 11, 20, 13, 31, 19, 27, 15, 29, 22, 8, 21, 31, 21, 11, 9, 0, 15, 10, 7, 4, 17, 14, 16, 24, 9, 22, 3, 55, 17, 3, 6, 46, 23, 21, 14, 18, 18, 1, 4, 18, 25, 25, 0, 16, 7, 8, 19, 51, 25, 30]

Ключ:

[14, 32, 13, 13, 16, 21, 1, 8, 0, 14, 29, 8, 27, 16, 30, 23, 7, 29, 5, 21, 17, 32, 15, 25, 4, 29, 8, 23, 5, 1, 1, 9, 14, 13, 23, 21, 8, 4, 9, 23, 20, 31, 20, 3, 20, 5, 14, 2, 32, 21, 29, 10, 2, 26, 14, 3, 17, 13, 28, 27, 2, 25, 32, 23, 12, 3, 18, 3, 12, 4, 23, 20, 23, 24, 31, 30, 25, 26, 2, 7, 15, 15, 31, 6, 23, 31, 26, 1, 8, 29, 24, 1, 7, 26, 20, 26, 9, 16, 1, 32, 12, 8, 25, 10, 29, 6, 31, 1, 3, 7, 22, 28, 3, 19, 8, 5, 6, 32, 21, 21, 7, 14, 15, 10, 26, 6, 2, 2, 2, 0, 2, 14, 29, 12, 0, 24, 18, 24, 5, 32, 5, 20, 19, 9, 27, 7, 25, 5, 2, 14, 9, 23, 12, 12, 16, 13, 1, 16, 19, 26, 2, 26, 7, 17, 10, 32, 4, 19, 18, 8, 25, 24, 4, 5, 5, 26, 15, 4, 1, 14, 22, 9, 2, 26, 4, 9, 18, 19, 5, 15, 2, 11, 12, 8, 13, 21, 2, 7, 23, 12, 7, 17, 11, 27, 25, 26, 9, 7, 16, 7, 13, 15, 8, 26, 22, 15, 12, 6, 16, 17, 31, 23, 6, 24, 32, 31, 31, 9, 29, 4, 28, 30, 26, 3, 23, 15, 10, 24, 18, 18, 19, 28, 17, 25, 28, 2, 18, 22, 9, 14, 3, 29, 12, 12, 15, 26, 25, 8, 15, 26, 13, 2, 29, 28, 14, 29, 22, 0, 27, 21, 9, 1, 24, 28, 8, 21, 13, 10, 0, 6, 11, 14, 24, 28, 6, 5, 19, 13, 3, 20, 27, 14, 11, 11, 26, 12, 1, 31, 5, 23, 14, 25, 11, 12, 29, 13, 28, 32, 18, 29, 9, 20, 24, 16, 5, 13, 9, 5, 19, 23, 19, 25, 0, 8, 17, 16, 13, 29, 5, 21, 2, 25, 4, 19, 17, 3, 1, 14, 30, 6, 0, 0, 26, 31, 23, 14, 1, 21, 6, 6, 3, 31, 30, 24, 14, 17, 26, 8, 16, 24, 13, 12, 7, 7, 28, 22, 32, 25, 27, 15, 2, 23, 23, 14, 12, 16, 5, 26, 5, 27, 14, 7, 30, 7, 16, 16, 4, 19, 31, 30, 3, 7, 25, 13, 28, 22, 1, 23, 20, 10, 14, 6, 23, 28, 4, 27, 8, 25, 13, 23, 12, 5, 11, 19, 23, 27, 20, 4, 0, 26, 9, 5, 28, 1, 27, 13, 12, 4, 0, 24, 2, 5, 30, 7, 1, 12, 21, 9, 23, 21, 15, 12, 11, 10, 4, 32, 0, 10, 12, 9, 26, 1, 24, 25, 22, 21, 0, 20, 20, 29, 6, 19, 25, 6, 3, 28, 11, 16, 28, 21, 1, 15, 0, 1, 2, 4, 6, 32, 14, 4, 26, 23, 21, 25, 10, 2, 1, 31, 26, 14, 21, 20, 17, 27, 20, 17, 8, 22, 19, 3, 24, 23, 1, 26, 15, 26, 11, 32, 8, 4, 19, 14, 2, 11, 9, 28, 21, 11, 32, 10, 14, 18, 0, 3, 18, 8, 14, 3, 14, 19, 11, 8, 20, 13, 28, 24, 8, 4, 16, 25, 21, 6, 30, 10, 23, 8, 21, 22, 23, 3, 16, 21, 21, 29, 4, 18, 15, 29, 13, 29, 1, 9, 7, 27, 28, 26, 27, 23, 31, 8, 21, 27, 24, 31, 4, 28, 32, 23, 18, 8, 27, 11, 19, 14, 32, 26, 2, 1, 4, 23, 23, 1, 0, 7, 1, 1, 0, 14, 8, 18, 28, 10, 14, 14, 1, 5, 0, 20, 20, 32, 28, 6, 3, 10, 20, 14, 31, 31, 9, 8, 4, 19, 13, 30, 13, 22, 2, 25, 5, 21, 22, 25, 28, 12, 15, 5, 22, 20, 3, 30, 0, 30, 31, 24, 32, 25, 8, 9, 4, 26, 2, 30, 15, 10, 0, 17, 14, 12, 28, 6, 30, 29, 31, 28, 17, 32, 11, 16, 8, 5, 13, 23, 4, 16, 5, 11, 12, 22, 1, 13, 27, 23, 21, 9, 19, 11, 27, 26, 3, 22, 30, 17, 31, 21, 17, 28, 3, 14, 5, 11, 7, 29, 31, 7, 18, 3, 11, 29, 10, 17, 21, 7, 23, 1, 25, 10, 28, 7, 19, 30, 23, 20, 26, 3, 26, 10, 25, 25, 31, 30, 10, 16, 11, 0, 9, 26, 13, 10, 13, 24, 2, 18, 32, 31, 30, 20, 16, 20, 21, 6, 30, 12, 6, 31, 17, 9, 5, 9, 8, 2, 12, 14, 3, 1, 23, 15, 10, 9, 17, 24, 2, 25, 6, 25, 8, 25, 2, 4, 24, 7, 5, 30, 2, 14, 24, 26, 1, 24, 1, 31, 31, 6, 16, 32, 17, 27, 20, 22, 21, 26, 19, 27, 21, 12, 28, 16, 15, 16, 26, 32, 18, 14, 19, 28, 12, 10, 5, 31, 23, 15, 12, 10, 8, 11, 30, 17, 0, 10, 21, 20, 16, 32, 6, 27, 29, 23, 3, 8, 25, 19, 9, 14, 26, 4, 1, 30, 28, 31, 26, 11, 3, 5, 17, 25, 3, 20, 31, 26, 29, 12, 19, 20, 5, 2, 24, 20, 15, 24, 6, 3, 0, 22, 24, 32, 16, 12, 25, 23, 11, 20, 27, 7, 5, 12, 5, 23, 30, 5, 15, 9, 1, 0, 8, 30, 20, 14, 27, 13, 21, 32, 25, 0, 5, 18, 21, 20, 32, 5, 24, 4, 19, 23, 2, 32, 13, 9, 30, 31, 21, 15, 20, 7, 31, 31, 28, 31, 32, 0, 28, 21, 15, 0, 12, 32, 29, 26, 32, 28, 16, 26, 10, 3, 32, 17, 14, 22, 4, 32, 1, 23, 19, 11, 28, 23, 18, 23, 15, 19, 7, 26, 1, 15, 20, 18, 23, 3, 2, 15, 9, 11, 14, 23, 31, 9, 6, 14, 1, 31, 17, 20, 3, 15, 24, 14, 19, 20, 11, 28, 13, 13, 24, 1, 24, 26, 23, 15, 19, 30, 19, 19, 26, 31, 19, 31, 11, 3, 32, 18, 16, 19, 17, 5, 27, 19, 3, 30, 28, 18, 10, 10, 3, 6, 21, 9, 0, 9, 21, 21, 27, 13, 1, 0, 15, 19, 25, 1, 6, 5, 27, 12, 16, 31, 6, 22, 20, 24, 26, 6, 29, 1, 5, 29, 7, 1, 10, 16, 1, 31, 14, 24, 24, 26, 10, 17, 23, 9, 23, 14, 32, 23, 30, 1, 16, 19, 4, 12, 2, 8, 22, 1, 21, 11, 7, 17, 32, 1, 21]

Расшифрованный текст:

вотпримерстатьинатысячусимволов.этодостаточномаленькийтекст,оптимальноподходящийдлякарточектовароввинтернетилимагазинахилидлянебольшихинформационныхпубликаций.втакомтекстередкобываетболеедвухилитрёхабзацевиобычноодинподзаголовок.номожноибезнего.натысячусимволоврекомендованоиспользоватьодинилидваключаиоднукартину.текстнатысячусимволовэтосколькопримернослов.статистикапоказывает,чтотысячавключаетвсебястопятьдесятилидвестисловсреднейвеличины.но,еслизлоупотреблятьпредлогами,союзамиидругимичастямиречинаодинилидвасимвола,токоличествословнеизменновозрастает.вкопирайтерскойдеятельностипринятосчитатьтысячиспробеламиилибез.учетпробеловувеличиваетобъемтекстапримернонастоилидвестисимволовименностолькоразмыразделяемсловасвободнымпространством.считатьпробелызаказчикинелюбят,таккакэтопустоеместо.однаконекоторыефирмыибирживидятсправедливымставитьстоимостьзатысячусимволовспробелами,считаяпоследниеважнымэлементомкачественноговосприятия.согласитесь,читатьслитныйтекстбезединогопропуска,никтонебудет.нобольшинствунужнаценазатысячузнаковбезпробелов.

## Гаммирование ГОСТ 28147-89

При работе ГОСТ 28147-89 в режиме гаммирования описанным выше образом формируется криптографическая гамма, которая затем побитно складывается по модулю 2 с исходным открытым текстом для получения шифротекста. Шифрование в режиме гаммирования лишено недостатков, присущих режиму простой замены. Так, даже идентичные блоки исходного текста дают разный шифротекст, а для текстов с длиной, не кратной 64 бит, "лишние" биты гаммы отбрасываются. Кроме того, гамма может быть выработана заранее, что соответствует работе шифра в поточном режиме.

**Код программы:**

# -\*- coding:utf-8 -\*-

import sys

import numpy.random

import itertools

from base import alphabet, input\_for\_cipher\_short, input\_for\_cipher\_long, output\_from\_decrypted

import binascii

class GostCrypt(object):

    def \_\_init\_\_(self, key, sbox):

        self.\_key = None

        self.\_subkeys = None

        self.key = key

        self.sbox = sbox

    @staticmethod

    def \_bit\_length(value):

        return len(bin(value)[2:])

    @property

    def key(self):

        return self.\_key

    @key.setter

    def key(self, key):

        self.\_key = key

        self.\_subkeys = [(key >> (32 \* i)) & 0xFFFFFFFF for i in range(8)]

    def \_f(self, part, key):

        temp = part ^ key

        output = 0

        for i in range(8):

            output |= ((self.sbox[i][(temp >> (4 \* i)) & 0b1111]) << (4 \* i))

        return ((output >> 11) | (output << (32 - 11))) & 0xFFFFFFFF

    def \_decrypt\_round(self, left\_part, right\_part, round\_key):

        return left\_part, right\_part ^ self.\_f(left\_part, round\_key)

    def encrypt(self, plain\_msg):

        def \_encrypt\_round(left\_part, right\_part, round\_key):

            return right\_part, left\_part ^ self.\_f(right\_part, round\_key)

        left\_part = plain\_msg >> 32

        right\_part = plain\_msg & 0xFFFFFFFF

        for i in range(24):

            left\_part, right\_part = \_encrypt\_round(left\_part, right\_part, self.\_subkeys[i % 8])

        for i in range(8):

            left\_part, right\_part = \_encrypt\_round(left\_part, right\_part, self.\_subkeys[7 - i])

        return (left\_part << 32) | right\_part

    def decrypt(self, crypted\_msg):

        def \_decrypt\_round(left\_part, right\_part, round\_key):

            return right\_part ^ self.\_f(left\_part, round\_key), left\_part

        left\_part = crypted\_msg >> 32

        right\_part = crypted\_msg & 0xFFFFFFFF

        for i in range(8):

            left\_part, right\_part = \_decrypt\_round(left\_part, right\_part, self.\_subkeys[i])

        for i in range(24):

            left\_part, right\_part = \_decrypt\_round(left\_part, right\_part, self.\_subkeys[(7 - i) % 8])

        return (left\_part << 32) | right\_part

sbox = [numpy.random.permutation(l) for l in itertools.repeat(list(range(16)), 8)]

sbox = (

    (4, 10, 9, 2, 13, 8, 0, 14, 6, 11, 1, 12, 7, 15, 5, 3),

    (14, 11, 4, 12, 6, 13, 15, 10, 2, 3, 8, 1, 0, 7, 5, 9),

    (5, 8, 1, 13, 10, 3, 4, 2, 14, 15, 12, 7, 6, 0, 9, 11),

    (7, 13, 10, 1, 0, 8, 9, 15, 14, 4, 6, 12, 11, 2, 5, 3),

    (6, 12, 7, 1, 5, 15, 13, 8, 4, 10, 9, 14, 0, 3, 11, 2),

    (4, 11, 10, 0, 7, 2, 1, 13, 3, 6, 8, 5, 9, 12, 15, 14),

    (13, 11, 4, 1, 3, 15, 5, 9, 0, 10, 14, 7, 6, 8, 2, 12),

    (1, 15, 13, 0, 5, 7, 10, 4, 9, 2, 3, 14, 6, 11, 8, 12),

    )

key = 18318279387912387912789378912379821879387978238793278872378329832982398023031

text\_short = input\_for\_cipher\_short().encode().hex()

text\_short = int(text\_short, 16)

gost\_short = GostCrypt(key, sbox)

encode\_text\_short = gost\_short.encrypt(text\_short)

decode\_text\_short = gost\_short.decrypt(encode\_text\_short)

decode\_text\_short = bytes.fromhex(hex(decode\_text\_short)[2::]).decode('utf-8')

text\_long = input\_for\_cipher\_long().encode().hex()

text\_long = int(text\_long, 16)

gost\_long = GostCrypt(key, sbox)

encode\_text\_long = gost\_long.encrypt(text\_long)

decode\_text\_long = gost\_long.decrypt(encode\_text\_long)

decode\_text\_long = bytes.fromhex(hex(decode\_text\_long)[2::]).decode('utf-8')

print(f'''

Гост 28147-89:

КОРОТКИЙ ТЕКСТ:

Зашифрованный текст:

{encode\_text\_short}

Расшифрованный текст:

{output\_from\_decrypted(decode\_text\_short)}

ДЛИННЫЙ ТЕКСТ:

Зашифрованный текст:

{encode\_text\_long}

Расшифрованный текст:

{output\_from\_decrypted(decode\_text\_long)}

''')

**Тестирование:**

/bin/python3 /root/mospolytech-education-crypt-dev-2021-1/lab05\_14\_gost89.py

Гост 28147-89:

КОРОТКИЙ ТЕКСТ:

Зашифрованный текст:

56754026145183696286056690583114096463305996216823972540084957071450361516686534633542086281498134144445869433834001779406256238181699835567875396711529074587560012517759518637140963090682

Расшифрованный текст:

время,приливыиотливынеждутчеловека.

ДЛИННЫЙ ТЕКСТ:

Зашифрованный текст:

2660504179384763560870172300148736432061604479961145151290917749597470334059438903438971428697044508153936836050236848738880049098296804522950287185469780216499411778268801517322745336438645323207966302724507468373474684283063682350895571195422576014055891201234951917117574410777206443142305502703477508839984518986714583143400164257550139955951705291905260249349753577193635641327610319052903559952970011424379847993621475968548034595715017650775261104984599500375814441113013017875969917042030848511398372774550996207392191339181039778867511513245777658070846405184637307157764373019723472702707385419563060306821034731763551154503824714264812313316130706264074393244483028493575194280183695841632846694677441664704015987166764373865851781422124591521492824680316201781243695172538536956831291069312772130538659077026628200178989017827102455811284644197446147728227370823795695828448846841904902145096572681358073870106763470339835923246436026070889038200287505359250682058240850262033170450161709526251248889202783539456604116848059039836482794508570635524117505841700390724701975258248279406107572222767582208189240451665976152382603851822752929335645908960132784104671236312325136002226758406236192110306294400818405951215215170099869890409677082940113502073956343581542790367957821887506781091409598967321936432916690830836157099844286928789002776683458441918340765274704251161726938975327301299827924533461687715964034781601529646429568161764717204759432825169398902206756041104866390413896194971752446779789219834868520961790344530438359259087681089419095818947367924242461052662246177551436759868060964413164161379298518749743822796623991927620430286913534792524472501385616062283433788266136034182441382596852828678815464248898242655956549987693016474857317967045765805402303929238917825768453421398261020477598060009464910092710806727389232375505337101980972052186720533409389553482212142338388825165064316873200362417005979383125608949135305078202880494402590429918423534867498481325002725186208672068804910219444128725412313172857315960092225568479643477408050023889229147527678395405300014514788559927920823504740092277359994838446144638292221204810273531732178907255974872373943777174489278720360030245513206396286680923132665591932957479381688881713391447550197512421414080155618078837436716254239495738949352604395115116413190410600237037255778849942318131400487867993716612245815508666788386611535983488700840787806608061738649595754409762598103992609851236390147034058997120081926015140106802936589820873919900023139596870752250532835059504769730299189404268942453718027773910730364364478309783557301001910299876928643382267779773297344160509407488050435104053968601803987961053508330097865866726566439196742557135164782260908456730128332911775983685320723472682806496460888618031721287732914731095065151066593148360400366560924904292376661781984325986153706049185968639527031555339328097086138493583066405661508631851879253262157549893336342085850379421169485707846412390764082738059775423863152365222207358370420848642329032896143477728147462873088301241858500518352670460459172294780626495853313915274281775969597939390408588921013191052753639133045535705372177150216764101504465020904745448395363206321778018466643874865465360874268566642305436776611825466962196427195722777111044656000900826008555466570413212540474867009345483678501394846081885654740922361570249280181151517115435307624861660026083828186182751031813699667814905465529350145221133432739448140980614569357618476483550862855556317816602783334176925406116449091395553726830899141742583735706859991379762269375783746468934694732394565580644564403759572656563966408160663598247766009685660723020552419971761400822686542373391732314068592807190033601276697644910836577636698226330648236553701481899448499076055004435636081490697301414470015727863530449650719384883208173939823273509148707571491685195572982448679697231841618128665142056093605453472388015139879689507614636267932199613494818601175694495106823965735679852248453340323322884737319847422163663683329727707507865218141639579913178721468797011563675058848404432420337436492844035343087300285655784488990284687493681659591330992349220652565296410779661349740175153061305288645417558786186533328282877556855948224417771287356978869466410912431116815452381165518582443064263928368461040647310768425435788168185304791521567591202904460425598689779839465289865075045548803625879099049880457756634096216287517583379850827114094281279725186959719047716798991463990201696001595394956661030446430788409672457741058597704420444123772447654933964197877421550863353732634744480181841548898737461517584844294120903204988150464053418061288143949058682199344504008861383378392402642556140683240706965932049456398214008601555594844595489325508401028823872678584029686001711761573768314347772017072852998336991250377282698317207452140810115763062272712262299394159864510307804974680262082930993792882105434472845097445767818724620636250333319570116562701500864340123631182252024978565591020490325434070533337461974698559903223030221449025366488862781604023577829136213595700371095616666191295280158427548094843204106432381377975693276853993148974688913227746868482199358865760126313652003609854882178324694

Расшифрованный текст:

вотпримерстатьинатысячусимволов.этодостаточномаленькийтекст,оптимальноподходящийдлякарточектовароввинтернетилимагазинахилидлянебольшихинформационныхпубликаций.втакомтекстередкобываетболеедвухилитрёхабзацевиобычноодинподзаголовок.номожноибезнего.натысячусимволоврекомендованоиспользоватьодинилидваключаиоднукартину.текстнатысячусимволовэтосколькопримернослов.статистикапоказывает,чтотысячавключаетвсебястопятьдесятилидвестисловсреднейвеличины.но,еслизлоупотреблятьпредлогами,союзамиидругимичастямиречинаодинилидвасимвола,токоличествословнеизменновозрастает.вкопирайтерскойдеятельностипринятосчитатьтысячиспробеламиилибез.учетпробеловувеличиваетобъемтекстапримернонастоилидвестисимволовименностолькоразмыразделяемсловасвободнымпространством.считатьпробелызаказчикинелюбят,таккакэтопустоеместо.однаконекоторыефирмыибирживидятсправедливымставитьстоимостьзатысячусимволовспробелами,считаяпоследниеважнымэлементомкачественноговосприятия.согласитесь,читатьслитныйтекстбезединогопропуска,никтонебудет.нобольшинствунужнаценазатысячузнаковбезпробелов.