

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
Постоянный модуль.....	5
Блок А: ШИФРЫ ОДНОЗНАЧНОЙ ЗАМЕНЫ.....	6
1. Шифр простой замены АТБАШ.....	6
2. ШИФР ЦЕЗАРЯ.....	9
3. Квадрат Полибия.....	12
Блок В: ШИФРЫ МНОГОЗНАЧНОЙ ЗАМЕНЫ.....	16
4. Шифр Тритемия.....	16
5. Шифр Белазо.....	20
Блок С: ШИФРЫ БЛОЧНОЙ ЗАМЕНЫ.....	24
8. Матричный шифр.....	24
9. Шифр Плейфера.....	29
Д: ШИФРЫ ПЕРЕСТАНОВКИ.....	35
10. Шифр вертикальной перестановки.....	35
11. Решетка Кардано.....	39
Е: ШИФРЫ ГАММИРОВАНИЯ.....	44
13. Одноразовый блокнот К.Шеннона.....	44
Е: ПОТОЧНЫЕ ШИФРЫ.....	50
15. А5 /1.....	50
16. А5 /2.....	60
Блок G: КОМБИНАЦИОННЫЕ ШИФРЫ.....	70
17. МАГМА.....	70
БЛОК Н: АСИММЕТРИЧНЫЕ ШИФРЫ.....	78
21. RSA.....	78
Блок I: АЛГОРИТМЫ ЦИФРОВЫХ ПОДПИСЕЙ.....	83
24. RSA.....	83
25. El Gamal.....	87
Блок J: СТАНДАРТЫ ЦИФРОВЫХ ПОДПИСЕЙ.....	92
26. ГОСТ Р 34.10-94.....	92
27. ГОСТ Р 34.10-2012.....	97
Блок К: Обмен ключами.....	105
28. ОБМЕН КЛЮЧАМИ ПО ДИФФИ-ХЕЛЛМАНУ.....	105

## 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:

Главная

Программирование криптографических алгоритмов

Лаб. 1

Лаб. 2

Лаб. 3

Лаб. 4

Лаб. 5

Лаб. 6

Лаб. 7

Лаб. 8

Лаб. 9

Лаб. 10

Лаб. 11

Лабораторная работа 1. Шифры однозначной замены

1. Шифр АТБАШ

Перейти

2. Шифр Цезаря

Перейти

3. Квадрат Полибия

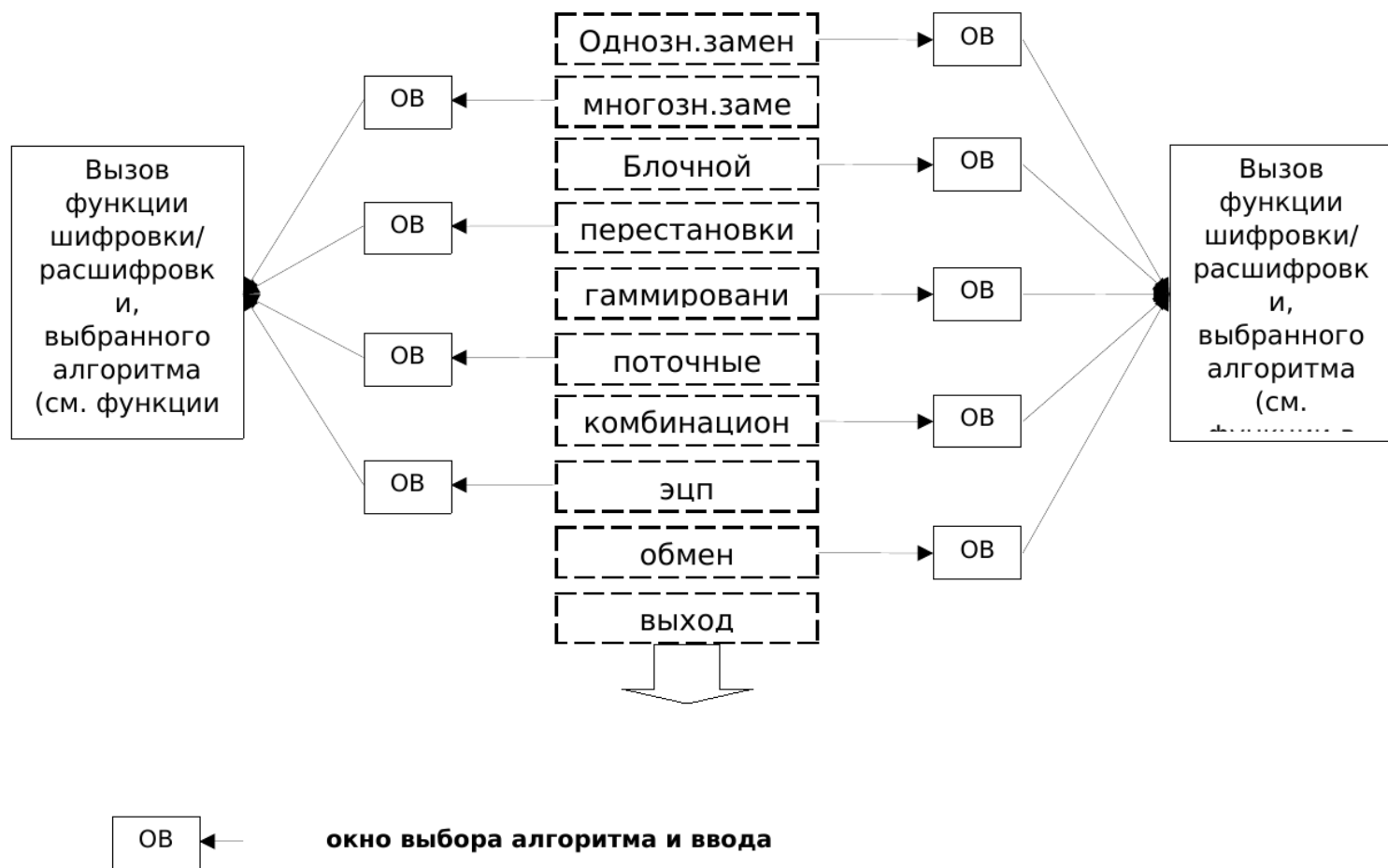
Перейти

Выполнил: Барышников С.С. 191-351

Интерфейс реализован с помощью фреймворка **Flask**. Программа имеет меню и разделение на страницы по блокам учебного курса.

Интерфейс можно протестировать онлайн <https://crypt.isenichl.ru/>.

**Блок-схема:**



**Код:**

```
from flask import Flask, render_template
app = Flask(__name__)
application = app

@app.route('/')
def index():
    return render_template('index.html')

from lab01 import bp as lab01_bp
app.register_blueprint(lab01_bp)

""" ... """

from lab11 import bp as lab11_bp
app.register_blueprint(lab11_bp)
```

## 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)

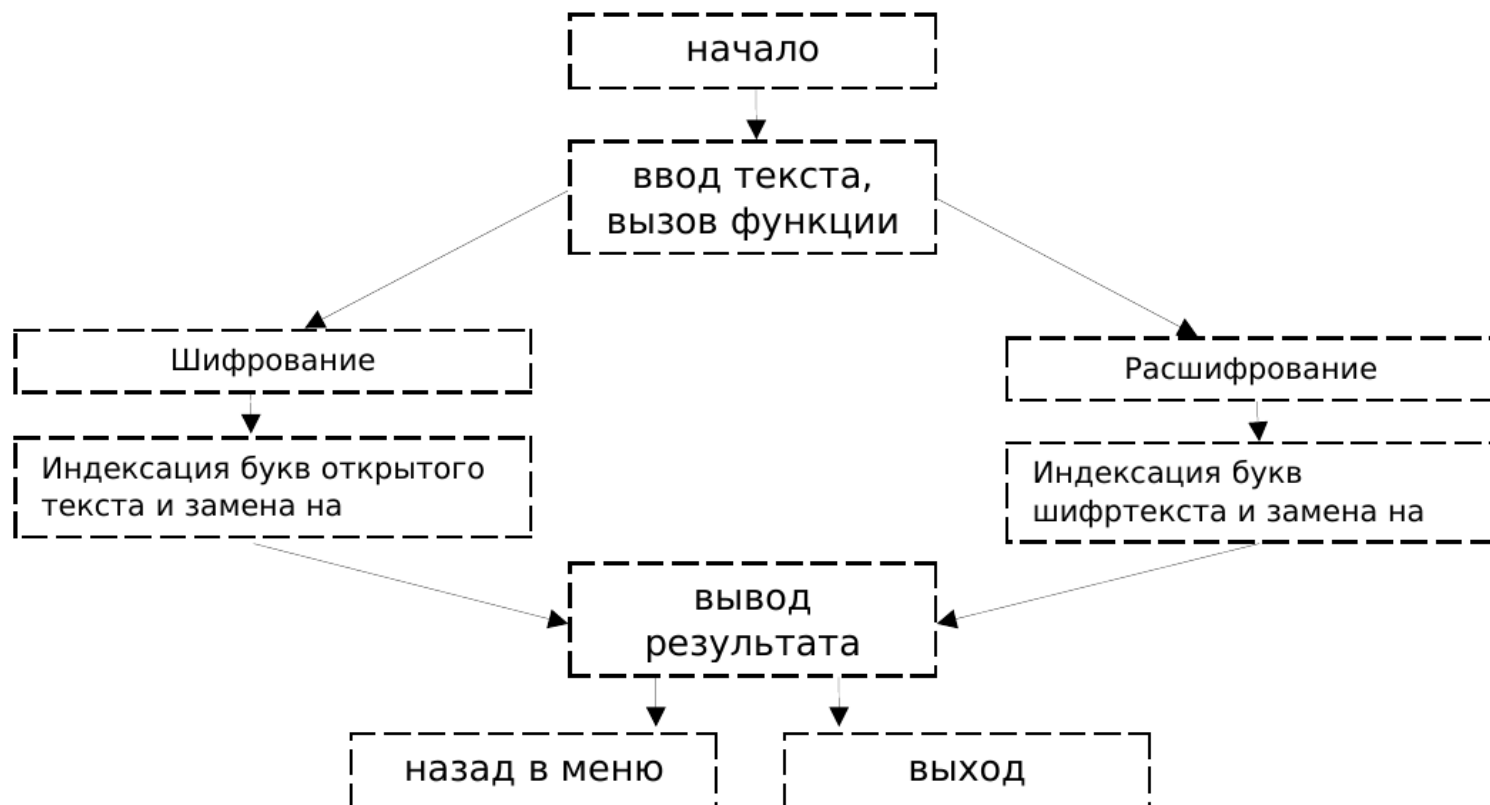
def output_from_decrypted(decrypted_text):
    return replace_all_from(decrypted_text, dict)
```

# Блок А: ШИФРЫ ОДНОЗНАЧНОЙ ЗАМЕНЫ

## 1. Шифр простой замены АТБАШ

Атбаш — простой шифр подстановки для алфавитного письма. Правило шифрования состоит в замене  $i$ -й буквы алфавита буквой с номером  $n-i+1$ , где  $n$  — число букв в алфавите.

Блок-схема:



Код программы:

```
# импорт компонентов, необходимых для работы программы
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted

# функция шифрования/расшифрования
def atbash(input):
    return input.translate(str.maketrans(
        alphabet + alphabet.upper(), alphabet[::-1] + alphabet.upper()
        [::-1]))

# вывод результатов работы программы
print(f'''
ШИФР АТБАШ:
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{atbash(input_for_cipher_short())}
Расшифрованный текст:
{output_from_decrypted(atbash(atbash(input_for_cipher_short())))}
ДЛИННЫЙ ТЕКСТ:
```

```
Зашифрованный текст:
{atbash(input_for_cipher_long())}
Расшифрованный текст:
{output_from_decrypted(atbash(atbash(input_for_cipher_long())))}
''' )
```

## Тестирование:

```
/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab01_1_atbash.py
```

ШИФР АТБАШ:

КОРОТКИЙ ТЕКСТ:

Зашифрованный текст:

эоътачпмпоцуцэдцрмуцэдсътшылмзъурэфямзф

Расшифрованный текст:

время, приливы и отливы не ждут человека.

ДЛИННЫЙ ТЕКСТ:

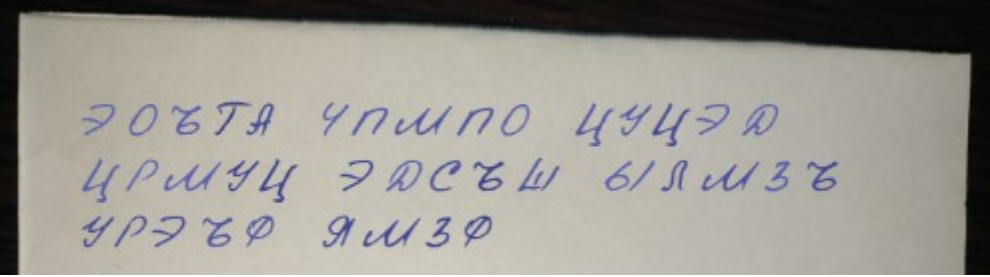
Зашифрованный текст:

эрмпотъонмямгцсямдназлнцтэрурэмзфвмрырнмямрзсртяуэстгфцхмъфнмчпмрпмцтяуг  
срприйрыаёцхуафяомрзъфмрэяорээцсмъосъмцуцтяъячсаяйцуцуаъсъругжцйцскротя  
ицрссдйплюцфяицхмзфэмяфртмъфнмъоъфрюдэяъмюруъыэлйцуцмощйяючяиъэцрюдзс  
ррыцспрычърурэрфмзфсртршсрцюъчсърмзфсямдназлнцтэрурэоъфртъсырэясрцнпру  
гчрэямгрыцсцуцызфубзяцрыслфяомцслмзфмъфнмсямдназлнцтэрурэвмрнфругфрпоцт  
ъосрнурэмзфнмямцнмцфяпрфячдэяъмчпмзмрмдназэяфубзяъмэнъюанмрпамгыънамцуцы  
эънмцнурэноъысътхъуцзцсдмзфсрчпмънуцчурлпрмоъюуамгпоъыуръятцчпмнрбчятццы  
ольцтцзянматцоъзцсярыцсцуцызянцтэруячпмрфруцъзмэрнурэсъцчтъсрэрчоянмя  
ъмзмзфэфрпцояхмъонфрхыъамъутсрнмцпоцсамрнзцяммгмдназцнпорюъуятццуцуюъчмзфл  
змпорюъурэлэуцзцэяъмрюетмъфнмяпоцтъосрсянмрцуцызънмцнцтэрурэцтъсрнмр  
угфроячтдоачыъуатнурэянэрюрысдтпорнмояснмэртмзфнзцяммгпорюъудчяфячзцфцс  
убюамчпммяффяфвмрплнмрътънмрмзфрысяфрсъфрмродъкцотдцюошцэцыамнпояэъуц  
эдтнмяэцмгнмрцтрнмгчямдназлнцтэрурэнпорюъуятцчпмнзцмяапрнуъысцъэашсдтвуй  
тъсмртфязънмэъсрърэрнпоамцамзфнръуянцмънгчпмзцяммгнцмсдхмъфнмюъчъыцср  
ърпорплнфячпмсцфмрсъюлыъмзмзфсрюругжцснмэлслшсяиъсячямдназлчсфрэюъчпорюъ  
урэмзф

Расшифрованный текст:

вот пример статьи на тысячу символов. это достаточно маленький текст, оптимально по  
ходящий для карточек товаров в интернет или магазина или для небольших информаци  
онных публикаций. в таком тексте редко бывает более двух или трёх абзацев и обычно один  
од заголовок. но можно и без него. на тысячу символов рекомендовано использовать оди  
ни или два ключа и одну картинку. текст на тысячу символов это сколько примерно слов. ста  
тистика показывает, что тысяча включает в себя столько десяти или двести слов средней  
величины. но, если злоупотреблять предложениями, союзами и другими частями речи, на оди  
ни или два символа, то количество слов не изменно возрастает. в копирайтерской деятел  
ьности принято считать тысячу пробелами или без. учёт пробелов увеличивает объём т  
екста примерно на столько, сколько символов именовано столько раз мы разделяем слова сво  
бодным пространством. считать пробелы заказчики не любят, так как это пустое место. од  
нако некоторые фирмы биржи видят справедливым ставить стоимость за тысячу символо  
в пробелами, считая последние важным элементом качественного восприятия. соглас  
итесь, читать слитный текст без единого пропуска, никто не будет. но большинству нуж  
на цена за тысячу знаков без пробелов.

Проверка:



Интерфейс:

Главная

Программирование криптографических алгоритмов

1. Шифр АТБАШ

Исходный текст

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 предложениями, 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

Зашифровать

Зашифрованный текст

эрмпоцтьонмягмцясмяднэлнцтэруэрэмфмрырн  
мямрэсртяуьсгфцмьфмнчмрлмцтяугспрпыйры  
аёцхуафяомрзьфмрзяорэзцсмьсьюмццтявяцц  
сйцццуюасьюргжцйскротяйцсдйглуюцфяи  
ццмзэмьяфртмьфнмьсфрюдзъмюръуьзэй  
цццмошйяючяиьэцрюдзсррыцспрычярурэрфмз  
фсрттршсрцюьсърмзфсмяднэлнцтэруэрэфрт  
ьсырзясрцпругрчрэмгырцццццфубзцрыслф  
яомцслэмзфмьфнмцясмяднэлнцтэруэрэмрфугр  
поцтьсрнурэмзфмямцмцфяпрфчдзъмчпмз  
мрмднзязфубзъмзъноанмрпамгьнамцццъз  
нмцнурэнзъсьхъуцццдмзсрчпмънуцурлпр  
мзюуамгпоъуурятцчпирбятцщыольццццям  
атцъзсъярцццццццянцтэруячпмрфццзмэ  
рнуръсцьцтьсрэрчочнямьмзфрпцояхмьонф  
рхьямъугсрнмцпоцсамрнццямгмднзцнпорю  
уьятцццуючмзфлзмпорюуэрэлъуццццямрю  
еътмьфмяпццтьсрсрянмрццццццмнцтэруэрэ  
тьсрнмруфрочтдолячъуаътнурзянэрюрцдтп  
орнмояснмэртмзфнццямгпорюудчяфчццфс

Расшифровать

Расшифрованный текст

вот пример статьи на тысячу символов. Это достато  
чно маленький текст, оптимально подходящий для кар  
точек товаров в интернет или магазинах или для не  
больших информационных публикаций. в таком тек  
сте редко бывает более двух или трёх абзацев и об  
ычно один подзаголовок. но можно и без него. на ты  
сячу символов рекомендовано использовать один или  
два ключа и одну картину. текст на тысячу символо  
в это сколько примерно слов. статистика показывает,  
что тысяча включает в себя сто пятьдесят или две  
сти слов средней величины. но, если злоупотребля  
ть предложениями, союзами и другими частями реч  
и на один или два символа, то количество слов неиз  
менно возрастает. в копирайтерской деятельности  
принято считать тысячи с пробелами или без. учет  
пробелов увеличивает объем текста примерно на ст  
о или двести символов именно столько раз мы

Очистить

Выполнил: Барышников С.С. 191-351

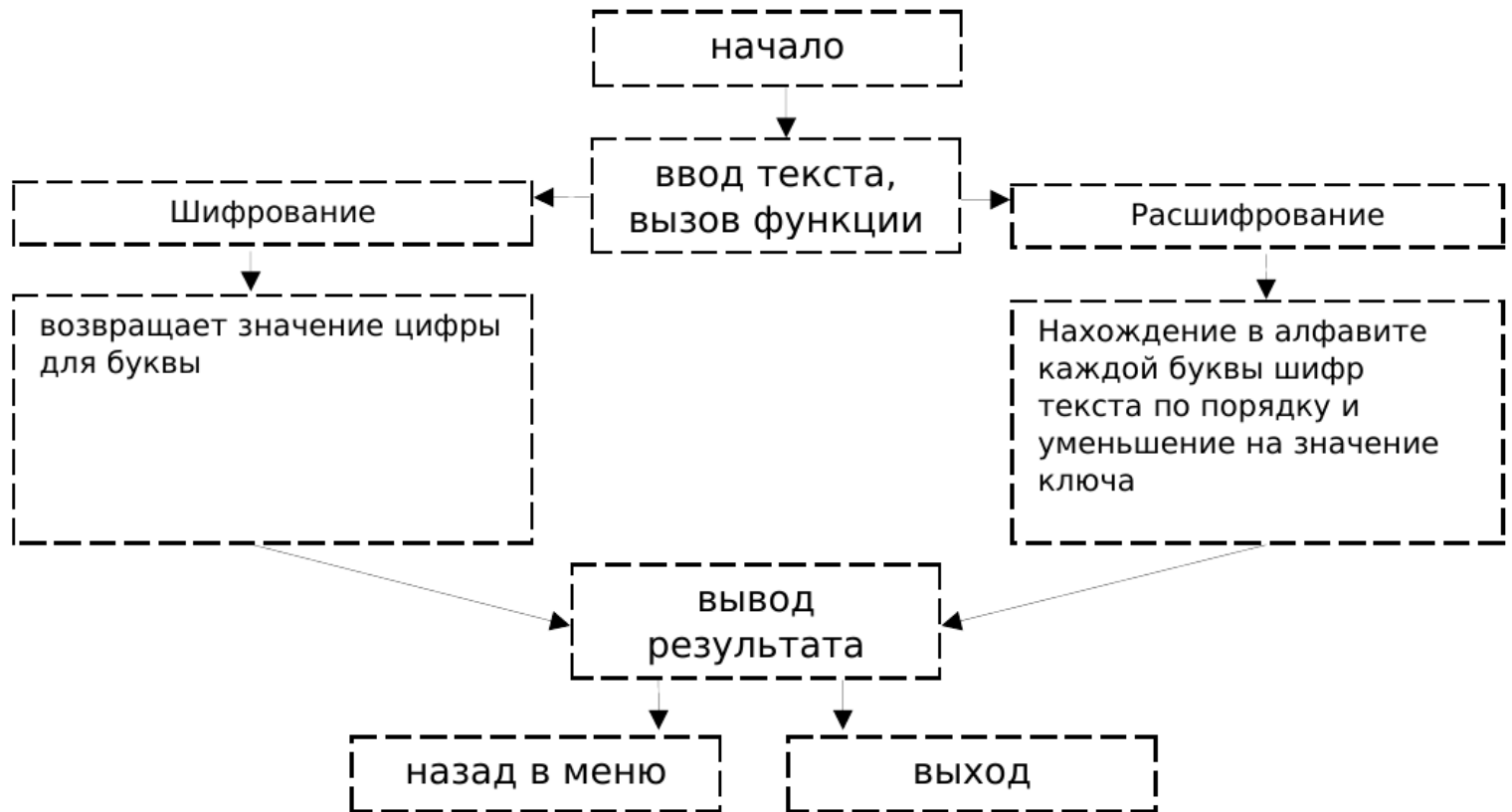


## 2. ШИФР ЦЕЗАРЯ

Шифр Цезаря, также известный как шифр сдвига, код Цезаря или сдвиг Цезаря — один из самых простых и наиболее широко известных методов шифрования.

Шифр Цезаря — это вид шифра подстановки, в котором каждый символ в открытом тексте заменяется символом, находящимся на некотором постоянном числе позиций левее или правее него в алфавите.

Блок-схема:



Код программы:

```
# импорт компонентов, необходимых для работы программы
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted
import random
# установка ключа
key = 5
# функция шифрования
def caesar_encode(input, step):
    return input.translate(
        str.maketrans(alphabet, alphabet[step:] + alphabet[:step]))
# функция расшифрования
def caesar_decode(input, step):
    return input.translate(
        str.maketrans(alphabet[step:] + alphabet[:step], alphabet))
# вывод результатов работы программы
print(f'''
```

```

ШИФР ЦЕЗАРЯ:
Ключ: {key}
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{caesar_encode(input_for_cipher_short(), key)}
Расшифрованный текст:
{output_from_decrypted(caesar_decode(caesar_encode(
    input_for_cipher_short(), key), key))}
ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
{caesar_encode(input_for_cipher_long(), key)}
Расшифрованный текст:
{output_from_decrypted(caesar_decode(caesar_encode(
    input_for_cipher_long(), key), key))}
'''

```

## Тестирование:

```

/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab01_2_caesar.py
ШИФР ЦЕЗАРЯ:
Ключ: 5
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
жхйсдмфчфхнржанучрнжатйлишчъйружйпечъп

Расшифрованный текст:
время, приливы и отливы не ждут человека.

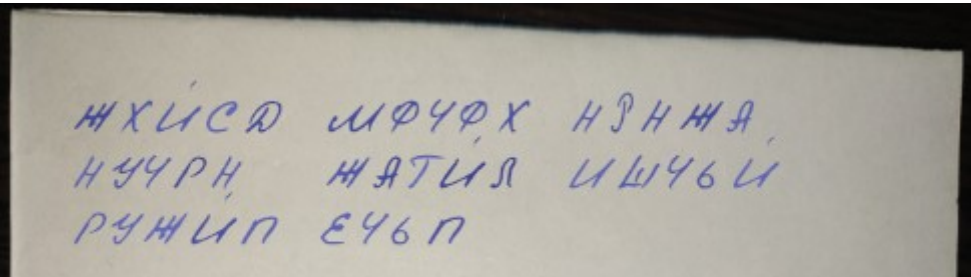
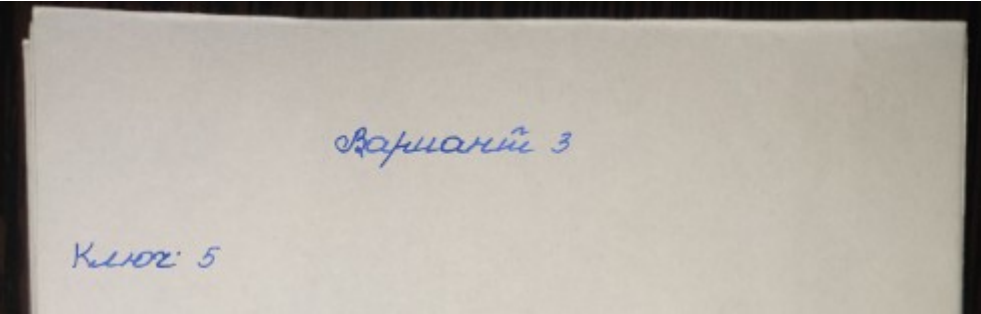
ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
жучфхнсийхцчечбнтечацдъшцнсжуружчъпвчуиуцчечуътусерйтбпночйппчмфчуфчнсерб
туфуиъуидюноирдпехчуъйпчужехужнтчйхтйчнрсеземнтеърнирдтйёурбэнънтщуксе
ынуттаъфшёрнпемыночьпжчепусчйпцчйхйипуёажейчёурййишънрнчхкьеёмеййжнуёаьту
унтффуимезуружупчъптусултунёймтйзучьптечацдъшцнсжуружхйпусйтиужетунцфурбм
ужеббуинтнрнижепргъенуитшпехчнтшчъпчйпцчтечацдъшцнсжуружвчуцпурбпуфхнсийх
туцружчъпцчечнцнпепфупемажейчмфчъчучацдъежпргъейчжцйёдцчуфдчбийцдчнрнижйц
чнцружцхйитйожирнънтачъптумфчйцнрмрушфучхйёрдчбфхйирузеснмфчцугмесннихшзн
снъецдснхйънтеуинтнрнижецнсжурумфччупурньйцжужцружтйнмситтужумхецчейчъ
пжпуфнхеочйхцпуойдчйрбтуцнфхнтдчуцънчечбчацдънцфхуёйресннрнёймчъпшъйчфх
уёйружшжирнънжейчуёяйсчйпцчэфхнсийхтутецчунрнижйцнцнсжуружнсийттуцчурбпuxe
мсахемийрдйсцружецжуёуитасфхуцхетцжусчъпцънчечбфхуёйрамепемьппнтйргёдчм
фччеппепвчуфшцчуйсйцчучъпуитепутйпучухайшнхсанёнхлнжнидцфжежйирнжасцчежн
чбцчунсуцчбмечацдъшцнсжуружцфхуёйреснмфчцънчедфуцрйитнйжелтасврйсийтчуспъ
йцчжйтттузужуцфхндчндчъпцузрецнйцбмфчънчечбцрнчтаочйпцчёймйинтузуфхуфшцпе
мфчтнпчутйёшийчъптуёурбэнтцжштшлтеийтемечацдъшмтепужёймфхуёйружчъп

Расшифрованный текст:
вот пример статьи на тысячу символов. это достаточно маленький текст, оптимально по
дходящий для карточек товаров в интернет или магазина или для небольших информацио
нных публикаций. в таком тексте редко бывает более двух или трёх абзацев и обычно один п
одзаголовок. но можно и без него. на тысячу символов рекомендовано использовать оди
ни или два ключа и одну картинку. текст на тысячу символов это сколько примерно слов. ста

```

статистика показывает, что тысяча включает в себя сто пятьдесят и две слова средней величины. Но, если злоупотреблять предлогами, союзами и другими частями речи, а одна или два символа, то количество слов неизменно возрастает. В копирайтерской деятельности принято считать тысячу пробелами или без. Учет пробелов увеличивает объем текста примерно на сто и две слова, а количество символов неизменно столько, сколько раз мы разделяем слова свободным пространством. Считать пробелы заказчики не любят, так как это пустое место. Однако некоторые фирмы биржи видят справедливым ставить стоимость за тысячу символов с пробелами, считая последние важным элементом качественного восприятия. Согласитесь, читать слитный текст без единого пропуса, никто не будет. Но большинству нужно наценить тысячу знаков без пробелов.

Проверка:



Интерфейс:

2. Шифр Цезаря

Ключ

5

Исходный текст

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 картинку. 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 тысячу с пробелами или без. Учет пробелов увеличивает объем текста примерно на сто или двести символов именно столько раз мы

Зашифрованный текст

жучфхнсьцщчбнчбтчащдщщснжуржчфвчуиуц чечуьтусерйтбпнчйцщмфучфчнсербтуфуиуид юноирдпехуьпчужехужжнтхйтхчнрсеземнте ънрнирдтйёурбэнънтщусеынутаъфшёрпееыно чьлжчепусйщпщхйипуважейчёруййижшьнрч ххсёёмейжнубаьтуинтфимезурупчъптусулт унеймтйзучьптецащдщснжуржуйситиужету нцйфурбмужебуйнтрнижепръненитшпехчнтщ ьпчйццтецащдщснжуржувчуцлурбпуфхнйхту цружчъццеченццнпфупемажеуцмфчуччащдье жпргъейжщйщдццуфдбйццнрнижйццнцружц хйитйожйрнътачпумфйццнрмушфучхйёрдчб фхйирузеснмфццгмесннхшзснъецдснхйнте уинтрнижеццжуремфццпурныйццжуржуйтн мсйтжумхецчейчьпжпуфнеоцйццпуйдчйр бтуццнфндццуннчбчащдщцфуйёреснрнёй мчльшъцфхуйёружжйрнънейчуйёйсйцццеф хнсьхтуеццнрнижйццнснжуржнсьтущурбп ухемсахемйрдысружеццуйёитасфущцхеццжу счъпцнчбфхуйёрапемъннтйгдчмфчеп

Расшифрованный текст

вот пример статьи на тысячу символов. это достато чно 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 картинку. 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 тысячу с пробелами или без. учет пробелов увеличивает объем текста примерно на сто и две слова, а количество символов неизменно столько, сколько раз мы разделяем слова свободным пространством. считать пробелы заказчики не любят, так как это пустое место. однако некоторые фирмы биржи видят справедливым ставить сто

Зашифровать

Расшифровать

Очистить

Выполнил: Барышников С.С. 191-351

11

### 3. Квадрат Полибия

Квадрат Полибия – метод шифрования текстовых данных с помощью замены символов, впервые предложен греческим историком и полководцем Полибием.

К каждому языку отдельно составляется таблица шифрования с одинаковым (не обязательно) количеством пронумерованных строк и столбцов, параметры которой зависят от его мощности (количества букв в алфавите). Берутся два целых числа, произведение которых ближе всего к количеству букв в языке — получаем нужное число строк и столбцов. Затем вписываем в таблицу все буквы алфавита подряд — по одной на каждую клетку. При нехватке клеток можно вписать в одну две буквы (редко употребляющиеся или схожие по употреблению).

Блок-схема:



Код программы:

```
# импорт компонентов, необходимых для работы программы
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted
# объявление алфавита
hard_dictionary = {"a": "11", "б": "12", "в": "13",
```

```

        "Г": "14", "Д": "15", "Е": "16", "Ё": "21", "Ж": "22",
"З": "23", "И": "24", "Й": "25", "К": "26", "Л": "31", "М": "32", "Н":
"33", "О": "34", "П": "35", "Р": "36", "С": "41", "Т": "42", "У": "43",
"Ф": "44", "Х": "45", "Ц": "46", "Ч": "51", "Ш": "52", "Щ": "53", "Ъ":
"54", "Ы": "55", "Ь": "56", "Э": "61", "Ю": "62", "Я": "63"}

# функция шифрования
def square_encode(input):
    new_txt = ""
    for x in input:
        if x in hard_dictionary:
            new_txt += hard_dictionary.get(x)
        else:
            new_txt += (x + x)
    return new_txt

# функция расшифрования
def square_decode(input):
    new_txt = ""
    list_fraze = []
    step = 2
    for i in range(0, len(input), 2):
        list_fraze.append(input[i:step])
        step += 2
    key_hard_dictionary_list = list(hard_dictionary.keys())
    val_hard_dictionary_list = list(hard_dictionary.values())

    for x in list_fraze:
        if x in val_hard_dictionary_list:
            i = val_hard_dictionary_list.index(x)
            new_txt += key_hard_dictionary_list[i]
        else:
            new_txt += x[0:1]
    return new_txt

# вывод результатов работы программы
print(f'''
КВАДРАТ ПОЛИБИЯ:
Ключ: {hard_dictionary}
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{square_encode(input_for_cipher_short())}
Расшифрованный текст:
{output_from_decrypted(square_decode(square_encode(
    input_for_cipher_short())))}
ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
{square_encode(input_for_cipher_long())}
Расшифрованный текст:
{output_from_decrypted(square_decode(square_encode(
    input_for_cipher_long())))}
''')

```

## Тестирование:

```
/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab01_3_square.py
КВАДРАТ ПОЛИБИЯ:
Ключ: {'а': '11', 'б': '12', 'в': '13', 'г': '14', 'д': '15', 'е': '16',
'ё': '21', 'ж': '22', 'з': '23', 'и': '24', 'й': '25', 'к': '26', 'л':
'31', 'м': '32', 'н': '33', 'о': '34', 'п': '35', 'р': '36', 'с': '41',
'т': '42', 'у': '43', 'ф': '44', 'х': '45', 'ц': '46', 'ч': '51', 'ш':
'52', 'щ': '53', 'ъ': '54', 'ы': '55', 'ь': '56', 'э': '61', 'ю': '62',
'я': '63'}
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
133616326323354235362431241355243442312413553316221543425116313413162611
425126
Расшифрованный текст:
время, приливы и отливы не ждут человека.

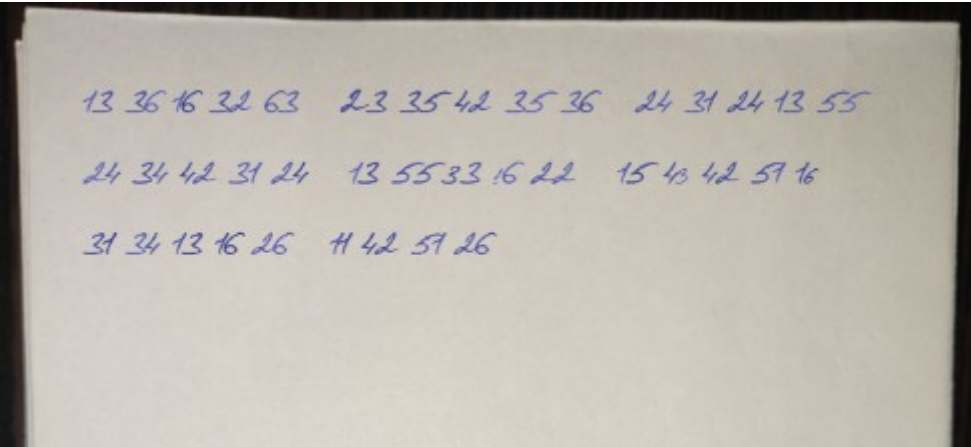
ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
133442353624321636414211425624331142554163514341243213343134134251266142
341534414211423451333432113116335626242542162641422335423435422432113156
333435341545341563532425153163261136423451162642341311363413132433421636
331642243124321114112324331145243124153163331612343156522445243344343632
114624343333554535431231242611462425425126134211263432421626414216361615
263412551311164212343116161513434524312442362145111223114616132434125551
333434152433353415231114343134133426425126333432342233342412162333161434
425126331142554163514341243213343134133616263432163315341311333424413534
315623341311425634152433243124151311263162511124341533432611364224334342
512642162641423311425541635143412432133431341361423441263431562634353624
321636333441313413425126414211422441422426113534261123551311164223354251
423442554163511113263162511116421341161263414234356342561516416342243124
151316414224413134134136161533162513163124512433554251263334233542164131
242331344335344236161231634256353616153134141132242335424134622311322424
153643142432245111414263322436165124331134152433243124151311412432133431
112335424234263431245116414213344131341333162423321633333413342336114142
111642425126132634352436112542163641263425151663421631563334414224353624
336342344151244211425642554163512441353634121631113224243124121623425126
435116423536341216313413431316312451241311164234125416324216264142113536
243216363334331141423424312415131641422441243213343134132432163333344142
343156263436112332553611231516316316324131341311411334123415335532353634
414236113341421334324251264151244211425635363412163155231126112351242624
331631621263422335424211262611266142343543414234163216414234425126341533
112634331626344234365516442436325524122436222413241563424135361113161531
241355324142111324425641423424323441425623114255416351434124321334313413
413536341216311132242335424151244211633534413116153324161311223355326131
163216334234322611511641421316333334143413344135362463422463425126413414
311141244216415623354251244211425641312442335525421626414212162316152433
341434353634354341261123354233242642343316124315164242512633341234315652
243341421343334322331146163311231142554163514323331126341312162335363412
16313413425126
```

Расшифрованный текст:



вот пример статьи на тысячу символов. это достаточно маленький текст, оптимально подходящий для карточек товаров в интернет или магазина или для небольших информационных публикаций. в таком тексте редко бывает более двух или трёх абзацев и обычно один подзаголовок. можно и без него. тысячу символов рекомендовано использовать или два ключа и одну картинку. 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 предложениями, союзами и другими частями речи, она или два символа, то количество слов неизменно возрастает. в копирайтерской деятельности принято считать тысячу 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. одна из некоторых фирм биржи видит справедливым поставить стоимость за тысячу символов с пробелами, считая последние важным элементом качественного восприятия. согласитесь, читать слитный текст без единого пропуска, никто не будет. но большинству нужно наценить за тысячу знаков без пробелов.

Проверка:



Интерфейс:

Главная

Программирование криптографических алгоритмов

3. Квадрат Полибия

Ключ

{'a': '11', 'b': '12', 'c': '13', 'd': '14', 'e': '15', 'f': '16', 'g': '21', 'h': '22', 'i': '23', 'j': '24', 'k': '25', 'l': '26', 'm': '31', 'n': '32', 'o': '33', 'p': '34', 'q': '35', 'r': '36', 's': '41', 't': '42', 'u': '43', 'v': '44', 'x': '45', 'y': '46', 'z': '51', 'w': '52', 'ш': '53', 'ъ': '54', 'ы': '55', 'ь': '56', 'э': '61', 'ю': '62', 'я': '63'}

Исходный текст

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 картинку. 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 предложениями, союзами и другими частями речи на один или два символа, то количество слов неизменно возрастает. В копирайтерской деятельности принято считать тысячу с пробелами или без. Учет пробелов

Зашифрованный текст

133442353624321636414211425624331142554163  
514341243213343134134251266142341534414211  
423451333432113116335626242542162641422335  
423435422432113156333435341545341563532425  
153163261136423451162642341311363413132433  
421636331642243124321114112324331145243124  
153163331612343156522445243344343632114624  
343333554535431231242611462425425126134211  
263432421626414216361615263412551311164212  
343116161513434524312442362145111223114616  
132434125551333434152433353415231114343134  
133426425126333432342233342412162333161434  
425126331142554163514341243213343134133616  
263432163315341311333424413534315623341311  
425634152433243124151311263162511124341533  
432611364224334342512642162641423311425541  
635143412432133431341361423441263431562634  
353624321636333441313413425126414211422441

Расшифрованный текст

вот пример статьи на тысячу символов. это достаточно маленький текст, оптимально подходящий для карточек товаров в интернет или магазинов или для больших информационных публикаций. в таком тексте редко бывает более двух или трёх абзацев и обычно один подзаголовок. можно и без него. на тысячу символов рекомендовано использовать один или два ключа и одну картинку. 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 предложениями, союзами и другими частями речи, она или два символа, то количество слов неизменно возрастает. в копирайтерской деятельности принято считать тысячу 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 заказчик

Зашифровать

Расшифровать

Очистить

Выполнил: Барышников С.С. 191-351

15

## Блок В: ШИФРЫ МНОГОЗНАЧНОЙ ЗАМЕНЫ

### 4. Шифр Тритемия

Шифр Тритемия предполагал использование алфавитной таблицы. Он использовал эту таблицу для многоалфавитного зашифрования самым простым из возможных способов: первая буква текста шифруется первым алфавитом, вторая буква — вторым и т. д. В этой таблице не было отдельного алфавита открытого текста, для этой цели служил алфавит первой строки. Таким образом, открытый текст, начинающийся со слов HUNC CAVETO VIRUM ..., приобретал вид HXPF GFBMCZ FUEIB ... .

Преимущество этого метода шифрования по сравнению с методом Альберти состоит в том, что с каждой буквой задействуется новый алфавит. Альберти менял алфавиты лишь после трех или четырех слов. Поэтому его шифртекст состоял из отрезков, каждый из которых обладал закономерностями открытого текста, которые помогали вскрыть криптограмму. Побуквенное зашифрование не дает такого преимущества. Шифр Тритемия является также первым нетривиальным примером периодического шифра. Так называется многоалфавитный шифр, правило зашифрования которого состоит в использовании периодически повторяющейся последовательности простых замен.

Блок-схема:



Код программы:

```
# импорт компонентов, необходимых для работы программы
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted

# функция расшифрования
def trithemius_decode(input):
```



```

decode: str = ""
k = 0
for position, symbol in enumerate(input):
    index = (alphabet.find(symbol) + k) % len(alphabet)
    decode += alphabet[index]
    k -= 1
return decode

# функция шифрования
def trithemius_encode(input):
    encode = ""
    k = 0
    for position, symbol in enumerate(input):
        index = (alphabet.find(symbol) + k) % len(alphabet)
        encode += alphabet[index]
        k += 1
    return encode

# вывод результатов работы программы
print(f'''
Шифр Тритемия:
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{trithemius_encode(input_for_cipher_short())}

Расшифрованный текст:
{output_from_decrypted(trithemius_decode(trithemius_encode(
    input_for_cipher_short())))}

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
{trithemius_encode(input_for_cipher_long())}

Расшифрованный текст:
{output_from_decrypted(trithemius_decode(trithemius_encode(
    input_for_cipher_long())))}
''')

```

## Тестирование:

```

/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab02_4_trithemius.py

Шифр Тритемия:
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
всжпгмхщцщтцфоиичюгэыхпгыкьюммтбимбелвхып

Расшифрованный текст:
время, приливы и отливы не ждут человека.

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:

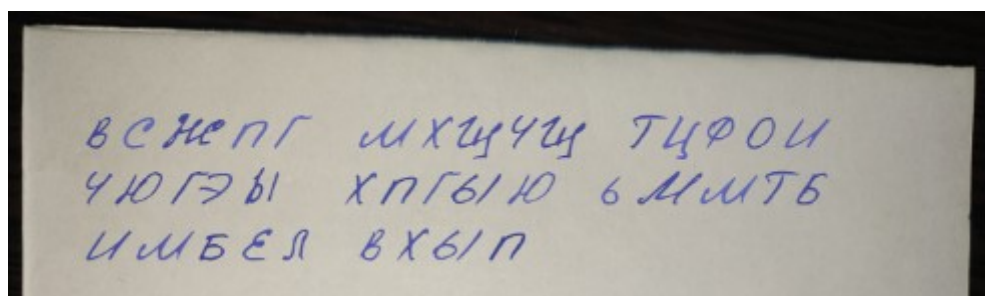
```

впффтнтлшъъкюицъпгмдтлизаеиижкярцкюфсзучщзышвщъъоыхяоюэяиъгкмгглпмотйогпб  
 ухччнаърмзшъхютяхжйжряёолаярпдемтшлщцоэфшцыпусъвъхладвюжыкгаомюымъофъчъчл  
 гцээкмзгзцвагшрдёпхйвувнтсшлтъпъсщюсмфущзчёдояяорузлйфуйъёзпиапнхъпкзб  
 вшюджджэвыялйнвпмхыпукфчршъхоучюцхвжмбешлхмыфсриндспузчмушчръсэсрябъёеегфб  
 иэъпъндйплпнйизухигмцэуеуеуезяеллсёовиртовхяцеюыгътчныщэсндбеядвугзйлейгпн  
 пуотжешъиуэяцщпааэуършчэлвкофрнтъувыезсужбкряпафсрдёгежйфэямыпжжкнедзхчощ  
 учлъъссъфъучяэмяеулсёйлёотёуоммсхышшэъсоъогнвдщвёйщжддмррояйгрнокшъмушке  
 обгряъеънаёицияекжжкнедгхнтицфйтയാэъяэыкшжищжкёйюкгззнжрсужхпщйъюмтбвзъфюг  
 еязшгмамоинийежвцйсгхчыэфъючэпдбелюичкхмцнъхзсртсейсжфстцфнтцвзъёкщзёзжъяж  
 кясбемкъмжёлъчкерщауъвдтгеюгъижиъорнимкжёйчщрътчныщэсщючбвдзйжецёкнюжмът  
 гликтжнцъчыпобтаувшгсдзйимонюмсудсрчэсщатэляйюаятчяпэцвшбсджёлддлягвцмжщйи  
 фхлчкбюаедёгъгъйлинузмгнбмссйрхъчъёрбцяълъуъеъяшбэщннмоявёёомжбущйймфяйав  
 яъявёзтяшлхмыфсриндспузшщшлршнышшбеелюиччинимзввмиёдгктбуефочръаэъысйшёзш  
 вёжиюкямрпсквцнещяуыщошнашгцдпъеиоблъншойзтоэмцйршъйотррцъюуавдгейигщвкок  
 жйтппзешйлъыбхщюэымйбёзшкёмбиъугмаивхяцеюырчкцыппфшбгвхъвёълсё

### Расшифрованный текст:

вот пример статьи на тысячу символов. это достаточно маленький текст, оптимально по  
 дходящий для карточек товаров в интернет или магазина или для небольших информационных  
 публикаций. в таком тексте редко бывает более двух или трёх абзацев и обычно один  
 из заголовков. можно и без него. на тысячу символов рекомендовано использовать  
 или два ключа и одну картинку. текст на тысячу символов это сколько примерно слова ста  
 тистика показывает, что тысяча включает в себя сто пятьдесят или двести слов средней  
 величины. но, если злоупотреблять предложениями, союзами и другими частями речи наоди  
 ни два символа, то количество слов не изменно возрастает. в копирайтерской деятель  
 ности принято считать тысячу пробелами или без. учёт пробелов увеличивает объём  
 текста примерно на сто или двести символов именно столько раз мы разделяем слова сво  
 бодным пространством. считать пробелы заказчики не любят, так как это пустое место. од  
 на некоторые фирмы биржи видят справедливым ставить стоимость за тысячу символо  
 в пробелах, считая последние важным элементом качественного восприятия. соглас  
 итесь, читать слитный текст без единого пропуска, никто не будет. но большинству нуж  
 на цена за тысячу знаков без пробелов.

### Проверка:



4. Шифр Третьеия

Исходный текст

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 предложениями, 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

Зашифровать

Зашифрованный текст

внрмджюиихжпярвляялюйсифртефшкърла  
йллшйемввясьфйчфхнтшмхурртзэзчззёкг  
чскщцщлцксъюкпцсдбнряеимноиезыяюрт  
оххкякспйргржвокьхергжмёжвфйаюэйгялхст  
жъннхшлвсюэкзмногийешцярйпмрэкчулкиёцч  
йлзрнвръачмъцбтлжъыпучщкнежспнипуё  
ззеёэггъущзурьювфцзбцбгъхтилалзыйъевеэцб  
цапъхэзъшфдождкопдзлэзюшвлуынафэвш  
мъэтцъцэпцмттипопчмшриявцеюатмьющхх  
тхххмпжснпикэмргиихжыдщъозыцксдйжкчлф  
штрншнъсщъевулцдтболоэзшщвиифбуиужб  
юалбивдцегфэуорцтарфбчнппорнжбдбей  
егёгцсынаиыомухирлйртрмыдъжвбжичъяз  
паалшуыняпуетжжюкжбюялвещебфывдгющышф  
воъйфцпсдоезёйбиззёёноёфеесхвбчнхшрчё  
цэтткнзббъмюдаддёбъбшьмъбъсбъжбцюацз  
пммйюндчгнозяткбечавоыччфйиоргубиетгы  
шйдхнчюгфщяялфсфлцттрвтнжияёыйяжазф  
аьюрцщрщчщщфрансжшмывзэгхъбеюатпап  
ымщцнгухотффсампюйжлпзмэётдямянрцэрзс

Расшифровать

Расшифрованный текст

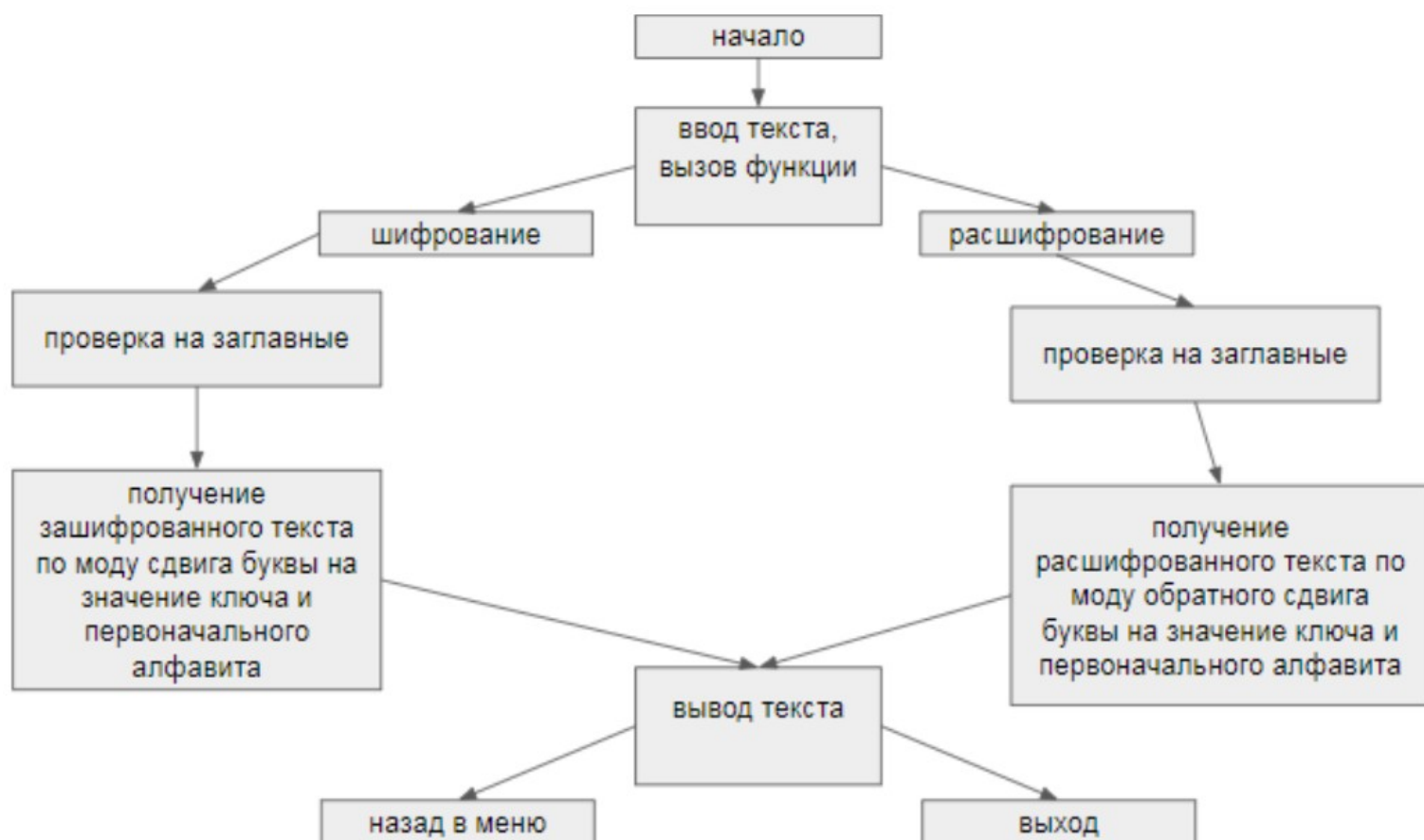
вотпримерстатънатысчусимволов.этодостаточ  
номаленькийтекст,оптимальноподходящийдляка  
рточектоваровинтернетилимагазинахилидляне  
большоихинформационныхпубликаций.втакомтек  
стередкобываетболеедвухилитрёхабзацевиобыч  
ноодинподзаголовок.номожноибезнего.натысч  
усимволоврекомендованоиспользоватьодинили  
дваключаоднукартину.текстнатысчусимволовэ  
тосколькопримернослов.статистикапоказывает,ч  
тотысчавключаетвсебясто пятьдесят или двести  
словсреднейвеличины.но,еслизлоупотреблятьпред  
логами,союзамиидругимичастямиречинаодинил  
идвасимвола,токоличествословнеизменновозра  
стает.вкопирайтерскойдеятельностипринятосчит  
атьтысчиспробеламиилибез.учетпробеловувел  
ичиваетобъёмтекстапримернонастоилидвестиси  
мволовименностолькоразмыразделяемсловавко  
боднымпространством.считатьпробелызаказчик  
инелюбят,таккакэтопустоеместо.однаконекотор  
ыефирмыибирживидятсправедливымставитьсто

Очистить

## 5. Шифр Белазо

В 1553 Джованни Баттиста Белазо предложил использовать для многоалфавитного шифра буквенный, легко запоминаемый ключ, который он назвал паролем. Паролем могло служить слово или фраза. Пароль периодически записывался над открытым текстом. Буква пароля, расположенная над буквой текста, указывала на алфавит таблицы, который использовался для зашифрования этой буквы. Например, это мог быть алфавит из таблицы Тритемия, первой буквой которого являлась буква пароля. Однако Белазо, как и Тритемий, использовал в качестве алфавитов шифра обычные алфавиты.

Блок-схема:



Код программы:

```
# импорт компонентов, необходимых для работы программы
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted

# установка ключа
key = 'ключ'

# функция расшифрования
def bellaso_decode(input, key):
    decrypted = ''
    offset = 0
    for ix in range(len(input)):
        if input[ix] not in alphabet:
            output = input[ix]
```

```

        offset += -1
        elif (alphabet.find(input[ix])) > (len(alphabet) -
(alphabet.find(key[((ix + offset) % len(key))])) - 1):
            output = alphabet[(alphabet.find(
                input[ix]) - (alphabet.find(key[((ix + offset) %
len(key))])) % 33]
            else:
                output = alphabet[alphabet.find(
                    input[ix]) - (alphabet.find(key[((ix + offset) %
len(key))]))]
            decrypted += output
    return decrypted

# функция шифрования
def bellaso_encode(input, key):
    encoded = ''
    offset = 0
    for ix in range(len(input)):
        if input[ix] not in alphabet:
            output = input[ix]
            offset += -1
            elif (alphabet.find(input[ix])) > (len(alphabet) -
(alphabet.find(key[((ix + offset) % len(key))])) - 1):
                output = alphabet[(alphabet.find(
                    input[ix]) + (alphabet.find(key[((ix + offset) %
len(key))])) % 33]
                else:
                    output = alphabet[alphabet.find(
                        input[ix]) + (alphabet.find(key[((ix + offset) %
len(key))]))]
                encoded += output
    return encoded

#вывод результатов работы программы
print(f'''
Шифр Белазо:
Ключ: {key}
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{bellaso_encode(input_for_cipher_short(), key)}

Расшифрованный текст:
{output_from_decrypted(bellaso_decode(bellaso_encode(
    input_for_cipher_short(), key), key))}

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
{bellaso_encode(input_for_cipher_long(), key)}

Расшифрованный текст:
{output_from_decrypted(bellaso_decode(bellaso_encode(
    input_for_cipher_long(), key), key))}
''')

```

## Тестирование:

```
/bin/python3  
/root/mospolytech-education-crypt-dev-2021-1/lab02_5_bellaso.py
```

Шифр Белазо:

Ключ: ключ

КОРОТКИЙ ТЕКСТ:

Зашифрованный текст:

мьгдйунйъжгунщащюямжльспсйврйёмричэги

Расшифрованный текст:

время, приливы и отливы не ждут человека.

ДЛИННЫЙ ТЕКСТ:

Зашифрованный текст:

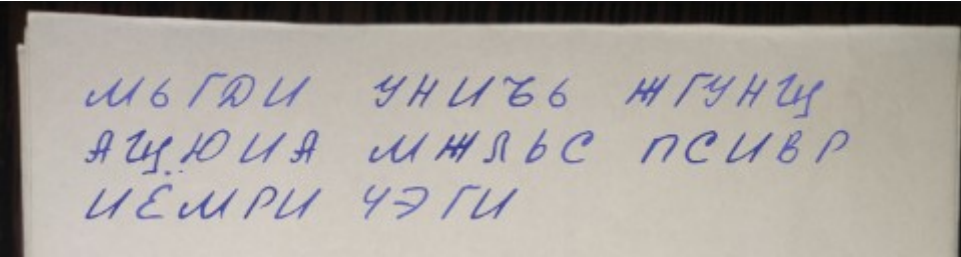
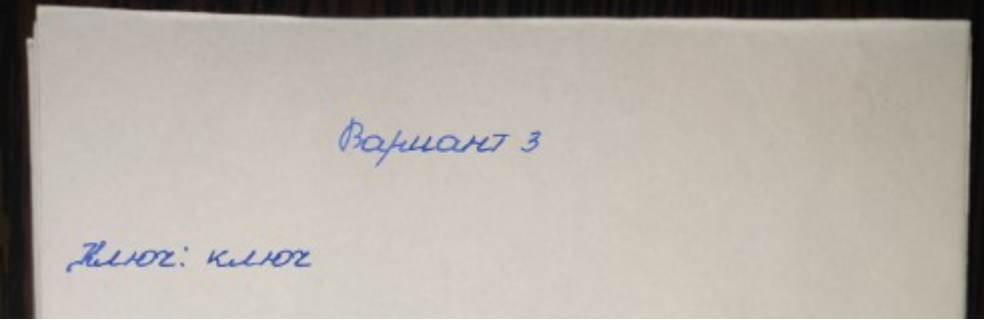
мържыфкыэрчэзжекущийгсиушаёцъайвцийщпмиэлрёвшмдкчгежцжбэрииэунйщырачлйуш  
ънёобмйежбчэзвкрёврийщнюзснаашюгзшррацфкчнлёашлуацфвгйщгшщчъпубжеяъодкв  
жёшщмъаягуцюнухрхнрчхъкйпцпйпъгыхъятмлгйлъйъппакафйаэъдмкмёчбраашмщошъм  
ыущнёоуъщчмщщцрохшмдштлёмгяшрбёэгиекущийгсиушаёцъазпцмдпщвёмллёуэнёцзёё  
млрущпжеучжымлигигюащплкхлойущсйвцрххэрекущийгсиушаёцъафёъпвщчъвщюачроещ  
эйёмюхвъюйуэрахлнёхлётмлгйтыроэъртъкхчмцйхвлгймэгшйэрёькруорпцэфйаонгиэф  
пгщнпзпплъфнггуггжеёюхвшъёжэрпгууйёюымйрыггйюъжырвгщюдуунйъъьякшжаовъсъушж  
окэрцчфовъфлчщпжеучжымлпачнмгкунйэъиёцфхъюаёъчмщржячрлещнмяылпйкррйвцав  
щыжзкхръыэиёфпгцэрьушъпйуыоашкрёьгжйкюъйёээоуэнзщмггкшжацфяътюхвюггйъьмшп  
чмщюнггуггжщкррёлёгдэрииэлнзущгзшълчъюмацфвщпэраьфкщчмщушгешъпйщчъвщюячж  
очтпггйркицъачънмшщплтчыоёъюочшэрщщшрохэхаэлруъьмшпчщякцюявфиашрйхлкряъор  
чхцювзюмжюэрёпшгиэърохъвекцмепцмйщъщъяфодёфяытжшупэйъыочмрвгунщдьюющуюи  
эъждшэрутлртькхкьфкщчмщъюёлрйччфёжээхаэлэжщэйъощжъмлееёшыгпшгееъквкгггиз  
нгешъбёмъпжыфёйукрохэмъцлпаэрпутыроуюйжэйаэщбэрииэмгяппжещомжыънкъцюяъю  
лажумепмсыпюрохшмшщчъпушпймялкщюнпщюякющийгсяшлиёммгяъьмшпчмщэги

Расшифрованный текст:

вот пример статьи на тысячу символов. это достаточно маленький текст, оптимально по  
дходящий для карточек товаров в интернет или магазина или для небольших информаци  
онных публикаций. в таком тексте редко бывает более двух или трех абзацев и обычно один п  
одзаголовок. но можно и без него. на тысячу символов рекомендовано использовать оди  
ни или два ключа и одну картинку. текст на тысячу символов это сколько примерно слов. ста  
тистика показывает, что тысяча включает в себя сто пятьдесят или двести слов средней  
величины. но, если злоупотреблять предлогами, союзами и другими частями речи, наоди  
ни или два символа, то количество слов не изменно и возрастает. в копирайтерской деятел  
ьности принято считать тысячу пробелами или без. учет пробелов увеличивает объем т  
екста примерно на сто или двести символов именно столько раз мы разделяем слова сво  
бодным пространством. считать пробелы заказчики не любят, так как это пустое место. од  
нако некоторые фирмы биржи видят справедливым поставить стоимость за тысячу симво  
лов в пробелах, считая последние важным элементом качественного восприятия. соглас  
итесь, читать слитный текст без единого пропуска, никто не будет. но большинство нуж  
на цена за тысячу знаков без пробелов.



Проверка:



Интерфейс:

Главная

Программирование криптографических алгоритмов

5. Шифр Белазо

Ключ

ключ

Исходный текст

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

Зашифровать

Зашифрованный текст

мържыфькьэрчэзжекоющийгсиушаёцъайвщйщп  
мизлрвщмдкчгекжцкбэризуищйрачлйушънё  
бмйежбчэвкьрврийцнозшнаашюгзшррацфк  
чнлвэшуацфегйгшщчпубжеяодквжэшщми  
ьяягуцонухрохнрчкьйцпйпгыхъятмлгйлъп  
пакафйазьдмкмёчбраащмошмьюшнёоующч  
мшщцрохшмдштлёмюяшрбёггекующийгсиушаё  
цъазпцмдпшвёмллёуэнёцзёёмлрушпжеучжмми  
гигоащплххлойущйвщрхэрекующийгсиушаёцъа  
фэъпвщчъвщьюаочроещэйёмнохвьююуэрахлнх  
лётмлйтырозьръкхчмцйхвлгймэгшйэрёкруорп  
цзфйаонгизфгпгнпзплпфнггугжёйохвшьёжэрп  
гууёюымйрягйюъжырегцоудунйъъякшжао  
ьсушжохэзрцфьфвщлщпжеучжмлпачнмгкун  
ййёйёцфьююёёмшщржячрлещнмьялпйкррйв  
цавщьюжэохрынёйфгпгзрйушпйуоашкрёгжй  
кюйёёзоэнзнгтгкшжафьафьтохвоггйёмшпчм  
щонггугжщкррёлгдзриэлнзшгшъльчюмацфв  
щпэраьфкшщмшущушешлпшчъщьюжчочтпгг  
йркицъачьнмшщплтчыёьюоочзрщшщроххазл

Расшифровать

Расшифрованный текст

вот пример статьи на тысячу символов. это достаточ  
но маленький текст, оптимально подходящий для кар  
точек товаров в интернет или магазинах или для не  
больших информационных публикаций. в таком тек  
сте редко бывает более двух или трёх абзацев и об  
ычно один подзаголовок. но можно и без него. на тыся  
чу символов рекомендовано использовать один или  
два ключа и одну картину. текст на тысячу символов э  
то сколько примерно слов. статистика показывает, ч  
тотысяча включает в себя сто пятьдесят или двести  
слов средней величины. но, если злоупотреблять пред  
логами, 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 увели  
чивает объем текста примерно на сто или двести сим  
болов именно столько раз мы разделяем слова в  
бодном пространстве. считать пробелы заказчик  
и не любит, так как это пустое место. однако некоторы  
е фирмы биржи видят справедливым ставить сто

Очистить

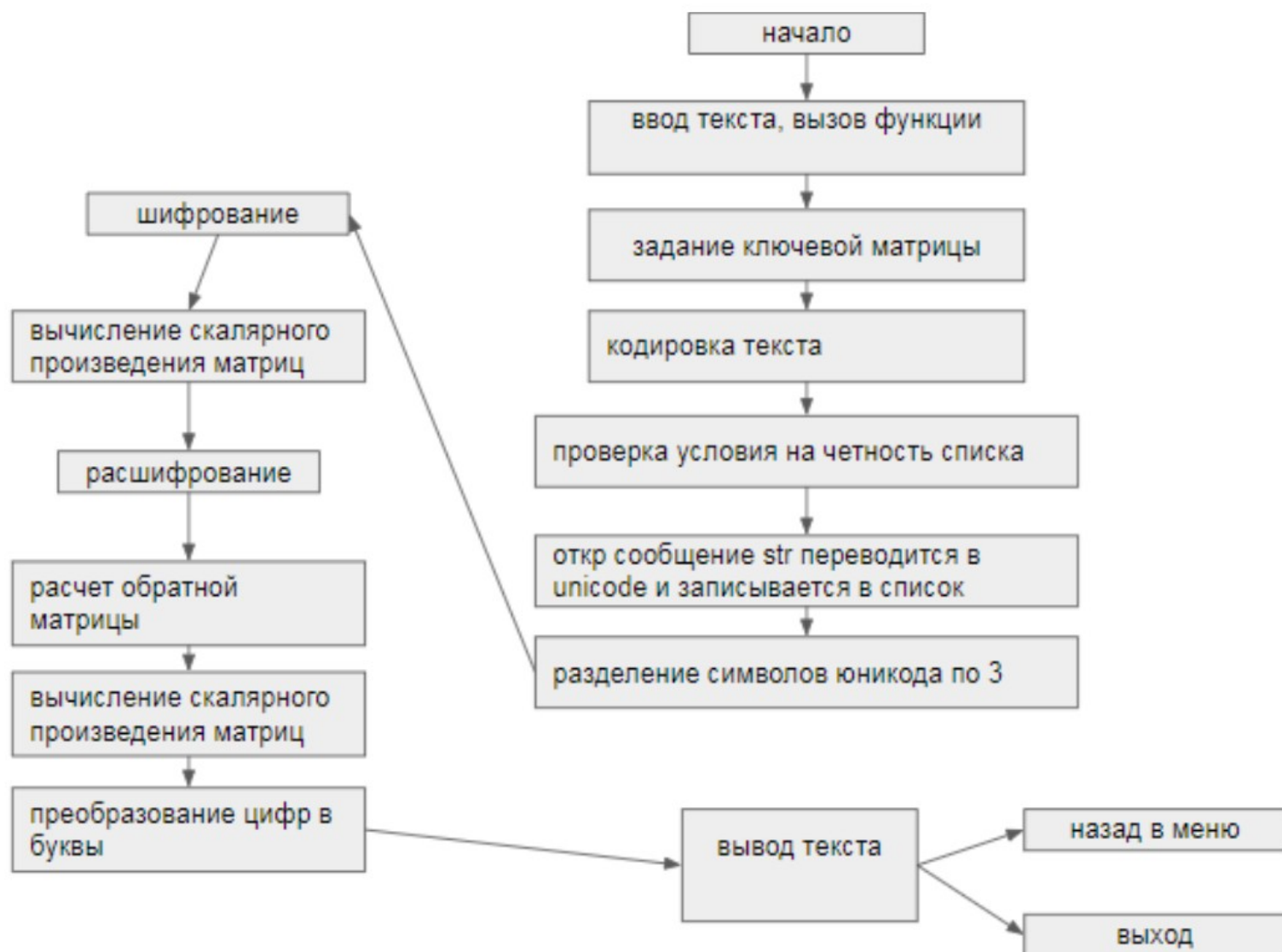
Выполнил: Барышников С.С. 191-351

## Блок С: ШИФРЫ БЛОЧНОЙ ЗАМЕНЫ

### 8. Матричный шифр

Шифр Хилла — полиграммный шифр подстановки, основанный на линейной алгебре и модульной арифметике. Изобретён американским математиком Лестером Хиллом в 1929 году. Это был первый шифр, который позволил на практике (хотя и с трудом) одновременно оперировать более чем с тремя символами. Шифр Хилла не нашёл практического применения в криптографии из-за слабой устойчивости ко взлому и отсутствия описания алгоритмов генерации прямых и обратных матриц большого размера.

Блок-схема:



Код программы:

```
# импорт компонентов, необходимых для работы программы
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted
import numpy as np
from egcd import egcd
```



```

# установка ключа
key = '3 10 20 20 19 17 23 78 17'
inp = key.split(' ')

key = np.matrix([[int(inp[0]), int(inp[1]), int(inp[2])], [int(inp[3]),
int(
inp[4]), int(inp[5])], [int(inp[6]), int(inp[7]), int(inp[8])]])

letter_to_index = dict(zip(alphabet, range(len(alphabet))))
index_to_letter = dict(zip(range(len(alphabet)), alphabet))

# функция вычисления обратной матрицы
def matrix_mod_inv(matrix, modulus):
    det = int(np.round(np.linalg.det(matrix)))
    det_inv = egcd(det, modulus)[1] % modulus
    matrix_modulus_inv = (
        det_inv * np.round(det * np.linalg.inv(matrix)).astype(int) %
modulus
    )

    return matrix_modulus_inv

# функция шифрования
def matrix_encode(message, K):
    # проверка на определитель равный 0
    if np.linalg.det(K) == 0:
        raise ValueError('Определитель матрицы равен 0! Дальнейшая
работа программы невозможна!')

    encrypted = ""
    message_in_numbers = []
    for letter in message:
        message_in_numbers.append(letter_to_index[letter])

    split_P = [
        message_in_numbers[i: i + int(K.shape[0])]
        for i in range(0, len(message_in_numbers), int(K.shape[0]))
    ]

    for P in split_P:
        P = np.transpose(np.asarray(P))[:, np.newaxis]

        while P.shape[0] != K.shape[0]:
            P = np.append(P, letter_to_index[" "])[:, np.newaxis]

        numbers = np.dot(K, P) % len(alphabet)
        n = numbers.shape[0]

        for idx in range(n):
            number = int(numbers[idx, 0])
            encrypted += index_to_letter[number]
    return encrypted

```

```

# функция расшифрования
def matrix_decode(cipher, Kinv):
    decrypted = ""
    cipher_in_numbers = []
    for letter in cipher:
        cipher_in_numbers.append(letter_to_index[letter])

    split_C = [
        cipher_in_numbers[i: i + int(Kinv.shape[0])]
        for i in range(0, len(cipher_in_numbers), int(Kinv.shape[0]))
    ]

    for C in split_C:
        C = np.transpose(np.asarray(C))[:, np.newaxis]
        numbers = np.dot(Kinv, C) % len(alphabet)
        n = numbers.shape[0]

        for idx in range(n):
            number = int(numbers[idx, 0])
            decrypted += index_to_letter[number]
    return decrypted

# вывод результатов работы программы
print(f'''
Матричный шифр:
Ключ: {key}
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{matrix_encode(input_for_cipher_short(), key).replace(' ', '')}

Расшифрованный текст:
{output_from_decrypted(matrix_decode(matrix_encode(
    input_for_cipher_short(), key), matrix_mod_inv(key,
len(alphabet))))}.replace(' ', '')}

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
{matrix_encode(input_for_cipher_long(), key).replace(' ', '')}

Расшифрованный текст:
{output_from_decrypted(matrix_decode(matrix_encode(
    input_for_cipher_long(), key), matrix_mod_inv(key,
len(alphabet))))}.replace(' ', '')}
''')

```

## Тестирование:

```

/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab03_8_matrix.py
Матричный шифр:
Ключ: 3 10 20 20 19 17 23 78 17
КОРОТКИЙ ТЕКСТ:

```

дѣйсѣнбнжбеѣнцмѣаэгщсѣттлюцгнхосцгфжгн

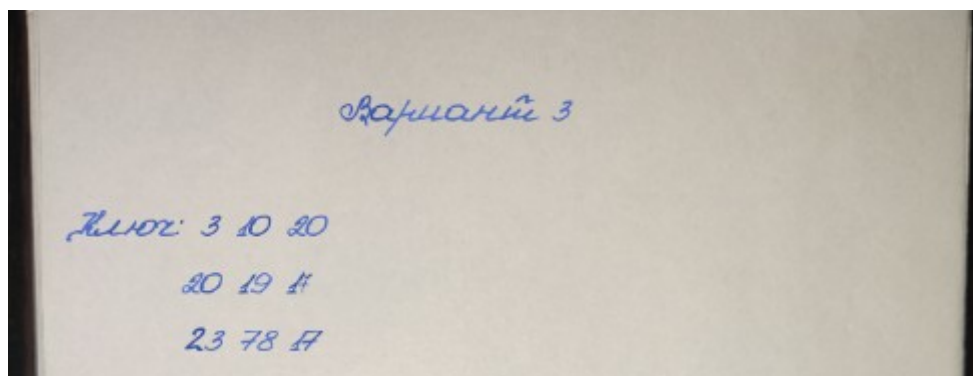
время, приливы и отливы не ждут человека.

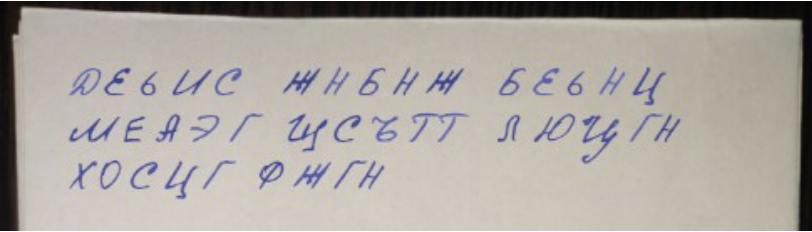
### Зашифрованный текст:

щвичнкфящёжщрээншусзуйдътюцбёэьяшцктыёжщжтийдмпжбзярюмсигфохжртичаужбвфэо  
вкквкыкзчвзяжтиьудюяшгмювжаепыэсофрдюейхьёзпфрдчвзйсоюбёштифчыхвлицижилшн  
дкыбокттжжгнщъябллфпыдчикъллвусфвпъбёёетстрэуижлиатйчлихчрозюпдмзмлёмзпжи  
цбёэввижгнйдемфшжтлйбншедщжгнншусзуйдътюцбёдёмбллющрюяшядмдбмйиخواшбээмлфр  
дяпйшцпнфшшвкаушмдуйюьсшигфъарыгхзчэдкхтщтяшцтиовьфчлчнкфящзфэхосжгнёжщюж  
ръчасождцповусштдбнившусзйдзшцпвфшьйотрцачаиъшцъйёгфрдеъпчияъжпшеъёттскиен  
аеэёгсюъвуефъацжшофабюязърцпяшлнкчтшюистщоъшхгчкгёядъчтйёъчаешъёяъьявшзм  
лфрдяпйэдкхтщщплгйяоъъжхлжщцэоаяшъбъоийфэцъаптъхйжгнзыгвядгобомийъжеёжкр  
мзпчичнкыфмшэомюцмэуэсзлеэзнкфмедщкфрдимежгнщцгмфвтэяхосймфыгщмнцъвпёдццэ  
обзчожтвнокдызпмвщвгщзыюжрътюцбёпчръщръмяубёмллаъпьяогнобрчгюяшажиьгдмжзн  
кёчичыдгъяьиммсшмюцфммрэнюхюёжюпплкьящбяъьнохжиъожфцшщъщъачижччачижгншвкв  
хшёсошязцезрэлцъсшдыоёмъсимзецарцётънцфжуэфгъмммвщебжюммвфудпулккктиэхосйб  
мтэячтфтщогземюцлиъвъжщцкбвжяьбнфжччхлтмъбёючжъяётгтьеобзобеежржгншпгхвдкн  
ёыъчдбнмюцъммэуилэчфпыщчишюззмлбасзнкэучцпозбнжяимещкштжвпуипозаяйщерджбйо  
зюьшштщрэжюусзчйдчшрозжншрэнъенэъягнх

во пример статьи на тысячу символов. это достаточно маленький текст, оптимально подходящий для карточек товаров в интернет-магазине или для небольших информационных публикаций. в таком тексте редко бывает более двух или трёх абзацев и обычно один-два заголовка. можно и без него. тысячу символов рекомендовано использовать или два ключа и одну картинку. 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, можно и два символа, то количество слов неизменно возрастает. в копирайтерской деятельности принято считать тысячу пробелами или без. учёт пробелов увеличивает объём 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 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 нужно наценить тысячу знаков без пробелов.

### Проверка:





Интерфейс:

8. Матричный шифр

Ключ

3 10 20 20 19 17 23 78 17

Исходный текст

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 предложениями, 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

Зашифровать

Зашифрованный текст

щвичнкфящёжщрээнш усзуйдътюцбёзя  
щщтыёжщжтийдмпжбзярюмсигожрти  
чаужбёфэовкккыкзчвз  
яжтиудюяшгнюваепызсофрдоейх  
ьёзпфрдчвэйсоюбёьштифч  
ыхвлицилщцдыбокттжжгнщьяблфпдычкь  
ллуусфелъбёётстрзуйжлятичлихчрзюпдзмл  
ёъзпжюцбёввижгнйдемфшжтлйбншедщжгнш  
усзуйдътюцбёдёьблпошряядмбмийихояшбэ  
змлфрдяпйшщпнфшшваошмдуйюьсщ игфа  
рыгхзчэдкштщтщтишьфчлчнкфящзфэхосжгнж  
щюжръчасожщповушт  
дбнившусэйдзщлпфшйшотрцаиашщтьё гфрдэ  
ьпчияьжпшеёттскиенаеэёгсювьефацжщофаб  
юязьрцпяшлнкштшюистщюьшхгчкёядьчтьёчеа  
жщъёьёявш  
змлфрдяпйэдкштщпгпгйяоььжжлщцзюеяшь  
бьюиыфэцаптьхй жгнзыгва дгобюмиьхеёж  
крмзпчичнкыфмшзоомцмзэусзлеэзнкфмедщкфр  
димежгнщцгмфвтэяхосймфыгшмнцвп

Расшифровать

Расшифрованный текст

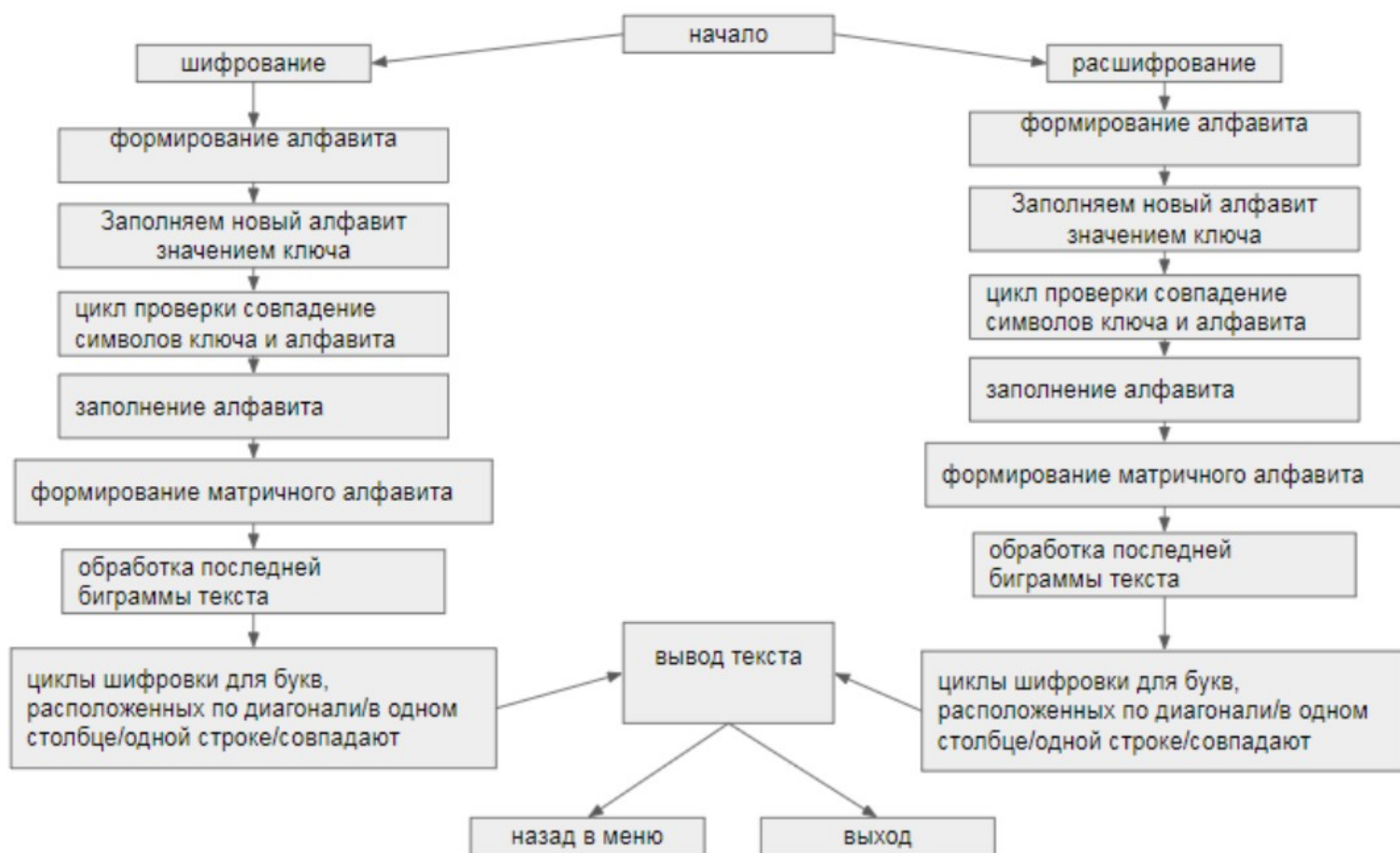
вот пример статьи на тысячу символов. это достаточ  
но маленький текст, оптимально подходящий для кар  
точек товаров в интернет или магазинах или для не  
больших информационных публикаций. в таком тек  
сте редко бывает более двух или трёх абзацев и об  
ычно один подзаголовок. но можно и без него. на ты  
сячу символов рекомендовано использовать один или  
два ключа и одну картину. текст на тысячу символо  
в это сколько примерно слов. статистика показывает,  
что тысяча включает в себя сто пятьдесят или две  
сти слов средней величины. но, если злоупотребля  
ть предложениями, союзами и другими частями речи  
на один или два символа, то количество слов неиз  
менно возрастает. в копирайтерской деятельности  
принято считать тысячи с пробелами или без. учет  
пробелов увеличивает объем текста примерно на ст  
о или двести символов именно столько раз мы

Очистить

## 9. Шифр Плейфера

Шифр Плейфера или квадрат Плейфера — ручная симметричная техника шифрования, в которой впервые использована замена биграмм. Изобретена в 1854 году английским физиком Чарльзом Уитстоном, но названа именем лорда Лайона Плейфера, который внёс большой вклад в продвижение использования данной системы шифрования в государственной службе. Шифр предусматривает шифрование пар символов (биграмм) вместо одиночных символов, как в шифре подстановки и в более сложных системах шифрования Виженера. Таким образом, шифр Плейфера более устойчив к взлому по сравнению с шифром простой замены, так как усложняется его частотный анализ. Он может быть проведён, но не для символов, а для биграмм. Так как возможных биграмм больше, чем символов, анализ значительно более трудоёмок и требует большего объёма зашифрованного текста.

Блок-схема:



Код программы:

```
# импорт компонентов, необходимых для работы программы
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted
alphabet = alphabet.replace(' ', '') + 'abc'

# установка ключа
key = 'ключ'

# функция шифрования
def playfair_encode(clearText, key):
```

```

text = clearText
new_alphabet = []
for i in range(len(key)):
    new_alphabet.append(key[i])
for i in range(len(alphabet)):
    bool_buff = False
    for j in range(len(key)):
        if alphabet[i] == key[j]:
            bool_buff = True
            break
    if bool_buff == False:
        new_alphabet.append(alphabet[i])
mtx_abt_j = []
counter = 0
for j in range(6):
    mtx_abt_i = []
    for i in range(6):
        mtx_abt_i.append(new_alphabet[counter])
        counter = counter + 1
    mtx_abt_j.append(mtx_abt_i)
# проверка на одинаковые биграммы
for i in range(len(text) - 1):
    if text[i] == text[i + 1]:
        if text[i] != 'я':
            text = text[:i + 1] + 'я' + text[i + 1:]
        else:
            text = text[:i + 1] + 'ю' + text[i + 1:]
# проверка на четную длину текста
if len(text) % 2 == 1:
    text = text + "я"
enc_text = ""
for t in range(0, len(text), 2):
    flag = True
    for j_1 in range(6):
        if flag == False:
            break
    for i_1 in range(6):
        if flag == False:
            break
    if mtx_abt_j[j_1][i_1] == text[t]:
        for j_2 in range(6):
            if flag == False:
                break
        for i_2 in range(6):
            if mtx_abt_j[j_2][i_2] == text[t+1]:
                if j_1 != j_2 and i_1 != i_2:
                    enc_text = enc_text + \
                        mtx_abt_j[j_1][i_2] + \
                        mtx_abt_j[j_2][i_1]
                elif j_1 == j_2 and i_1 != i_2:
                    enc_text = enc_text + \
                        mtx_abt_j[j_1][(i_1+1) % 6] + \
                        mtx_abt_j[j_2][(i_2+1) % 6]

```

```

        elif j_1 != j_2 and i_1 == i_2:
            enc_text = enc_text + \
                mtx_abt_j[(j_1+1) % 5][i_1] + \
                mtx_abt_j[(j_2+1) % 5][i_2]
        elif j_1 == j_2 and i_1 == i_2:
            enc_text = enc_text + \
                mtx_abt_j[j_1][i_1] + \
                mtx_abt_j[j_1][i_1]
        flag = False
        break

    return enc_text

# функция расшифрования
def playfair_decode(clearText, key):
    text = clearText
    new_alphabet = []
    for i in range(len(key)):
        new_alphabet.append(key[i])
    for i in range(len(alphabet)):
        bool_buff = False
        for j in range(len(key)):
            if alphabet[i] == key[j]:
                bool_buff = True
                break
        if bool_buff == False:
            new_alphabet.append(alphabet[i])
    mtx_abt_j = []
    counter = 0
    for j in range(6):
        mtx_abt_i = []
        for i in range(6):
            mtx_abt_i.append(new_alphabet[counter])
            counter = counter + 1
        mtx_abt_j.append(mtx_abt_i)
    if len(text) % 2 == 1:
        text = text + "я"
    enc_text = ""
    for t in range(0, len(text), 2):
        flag = True
        for j_1 in range(6):
            if flag == False:
                break
            for i_1 in range(6):
                if flag == False:
                    break
                if mtx_abt_j[j_1][i_1] == text[t]:
                    for j_2 in range(6):
                        if flag == False:
                            break
                        for i_2 in range(6):
                            if mtx_abt_j[j_2][i_2] == text[t+1]:
                                if j_1 != j_2 and i_1 != i_2:
                                    enc_text = enc_text + \

```



```

        mtx_abt_j[j_1][i_2] + \
        mtx_abt_j[j_2][i_1]
    elif j_1 == j_2 and i_1 != i_2:
        enc_text = enc_text + \
            mtx_abt_j[j_1][(i_1-1) % 6] + \
            mtx_abt_j[j_2][(i_2-1) % 6]
    elif j_1 != j_2 and i_1 == i_2:
        enc_text = enc_text + \
            mtx_abt_j[(j_1-1) % 5][i_1] + \
            mtx_abt_j[(j_2-1) % 5][i_2]
    elif j_1 == j_2 and i_1 == i_2:
        enc_text = enc_text + \
            mtx_abt_j[j_1][i_1] + \
            mtx_abt_j[j_1][i_1]
    flag = False
    break

return enc_text

#вывод результатов работы программы
print(f'''
Шифр Плейфера:
Ключ: {key}
Таблица : {table}
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{playfair_encode(input_for_cipher_short(), key)}

Расшифрованный текст:
{output_from_decrypted(playfair_decode(playfair_encode(
    input_for_cipher_short(), key), key))}

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
{playfair_encode(input_for_cipher_long(), key)}

Расшифрованный текст:
{output_from_decrypted(playfair_decode(playfair_encode(
    input_for_cipher_long(), key), key))}
''')
```

## Тестирование:

```

/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab03_9_playfair.py

Шифр Плейфера:
Ключ: ключ
Таблица:
['к', 'л', 'ю', 'ч', 'а', 'б'],
['в', 'г', 'д', 'е', 'ё', 'ж'],
['з', 'и', 'й', 'м', 'н', 'о'],
['п', 'р', 'с', 'т', 'у', 'ф'],
['х', 'ц', 'ш', 'щ', 'ъ', 'ы'],
```



[ 'в', 'э', 'я', 'а', 'b', 'с' ]

КОРОТКИЙ ТЕКСТ:

Зашифрованный текст:

гпмтьйрурсргзгцомфгржмёвефуембигёлбщеюь

Расшифрованный текст:

время, приливыиотливынеждутчеловека.я

ДЛИННЫЙ ТЕКСТ:

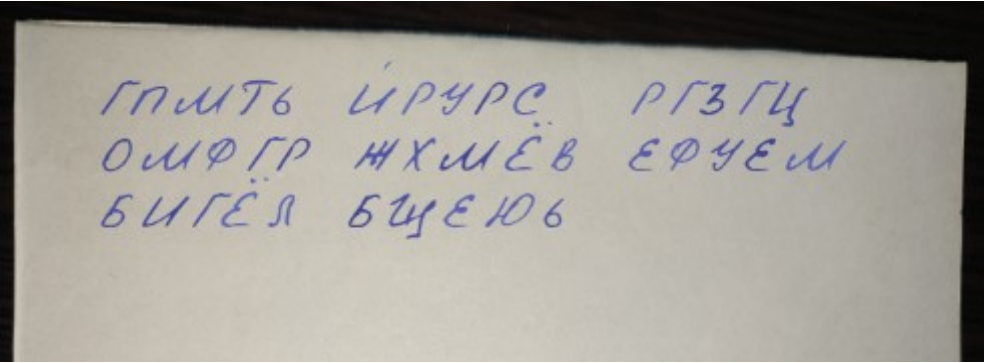
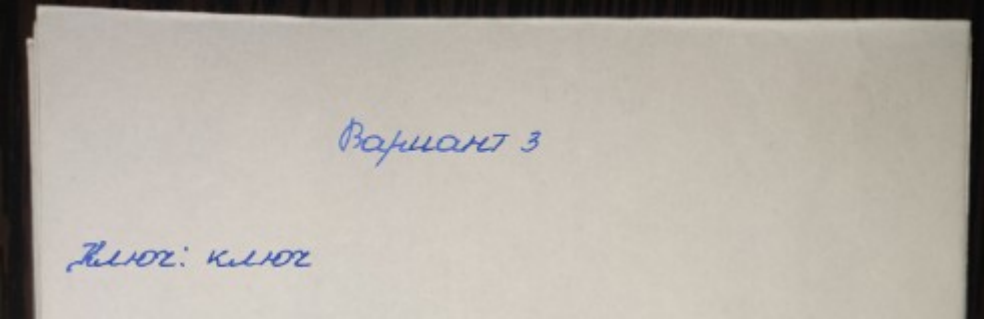
Зашифрованный текст:

жзурцртмстучпайочушфаюфтинжзбиепаларйжйфучфмамзньбюёмвзймщмюппмрузфрмнчкэо  
эфзвшйжашймгюьюлуфмемчпзжлузжгзмугтмёрмгрнчёлийуёцзгргюьйжчибяхзцийофтиль  
йзннхцрфкюзлльймщевзучбзтщвчтутгтёебзжбёкмщжфчгтёеёпцзгрусвъбкнкшггзфжщбозй  
жйофзвйлёибзжзбщеаззнфоозолвммёжищеазчушфаюфтинжзбигпвчзнёмжйёкозйрфзкэиз  
ёкпайжйоргйгёкключалнйжуълбсуйофуалщмюпумчушфаюфтинжзбигъфмпюибвзфцртмуий  
фбиепалтучуйррмлбфзлбожёмкщпхщэфмфщшдабзвючабмщдпжчдшфмсъпаеёшдрмгрегдтрм  
рюзжтсёемёздгчмлийошфалозпхщмрюйибифрмфггкюасвхтггюижчнйируфйкйчниигсрёмн  
люусанйтггмуйёйжйоргйгёкрйзеибкнруфмбзгремтузрюзжмёйитмннзжзиултучётталзв  
зфрцнщмстбзсйдащмкэозтузрцрйбфмтюрчуапшфаюйррсфжгччниигрчжмпалтамщрсфжг  
чзжпёгчмлзгчёфмаымтщмюпучрсйнгтозуётузйгрегдтрмрйзеибзжйнёмозтуибввифкнощ  
ульвгчадйтбиёкпдфжйжозтфитуульупезнщеплмучпарсфжгчхоблкнлмлзмёючюспмруу  
чккбларзффтфмттдтфмщбзёйблзовчмфифщжротицолофгзгйгаструлгёгюзгщотукёмряп  
фмйнийфпанкфщшдатрйзеибзжтрфижюбнйпхутлмучьсйфчгёймгёкёощоггмтёмфмзчбадтп  
еёмозжижзтрцрасйёщеюпижюбрийщмпяпхщемрчуяпгрумшошмюпфчвмёёиоижзффирфпюкнру  
ойчпзожчсёмщчазфжибяхйотуёпужёольёмкнчушфаюпнуёбзжквмрсфжгчзжщеюь

Расшифрованный текст:

вот пример статьи на тысячу символов. это достаточно маленввий текст, оптимально по  
дходящий для карточек товаров в интернет или магазина или для небольших информацио  
нных публикаций. в таком тексте редко бывает более двух или трех абзацев и обычно один  
од заголовок. но можно и без него. на тысячу символов рекомендовано использовать оди  
ни или два ключа и одну картинку. текст на тысячу символов это около 300 слов. ста  
тистика показывает, что тысяча включает в себя 500 слов и 500 слов средней  
величины. но, если злоупотреблять предложениями, союзами и другими частями речи на оди  
ни или два символа, то количество слов неизменно возрастает. в копирайтерской деятел  
ьности принято считать тысячу пробелами или без. учет пробелов увеличивает объем  
текста примерно на 50% и увеличивает количество слов в 2 раза. мы разделяем слова сво  
бодным пространством. считать пробелы заказчики не любят, так как это пустое место. од  
на некоторые фирмы биржи видят справедливым ставить стоимость за тысячу символо  
в пробелами, считая последние важными элементами качественного восприятия. соглас  
итесь, читать слитный текст без единого пропуска, никто не будет. но большинству нуж  
на цена за тысячу знаков без пробелов.я

Проверка:



Интерфейс:

Главная

Программирование криптографических алгоритмов

9. Шифр Плэйфера – шифр биграммной замены

Ключ

ключ

Исходный текст

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 предложениями, 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

Зашифровать

Зашифрованный текст

жзурцртмстцпайочушфаюфтинжзбиепаларйжй  
фучфмазнбюёмвйммцнопмрзфрмнчкзозфзв  
шйжашймгююлфмемчпзжлужждйощмуимшрг  
йнлэкнйокъргйнозмёжкэццзоиинчлрзобйхц  
рфоюзлльймщевзучбэтщвчтуттёебзжбёкмшжфг  
адегтьргмругькольёгйзжбамйсийжйфзвйлвйбз  
жзбжеазнфоозолвммёжищеазчшфаюфтинжзб  
игпвчзнёмжйёкозйрфкэизэкпайжйоргйгёкюча  
лнйжжълбсуйофуалцмюпумчушфаюфтинжзбигь  
фмпюибввзфцртмуййфбиепалтучуррмблфзлбо  
хёкмшлщцефмфшдабзвючабмшдпжчдшфмсьп  
аеёшдрмгрегдтрмрюзжтсёемёздчмлюощфалозп  
хщмрюибфмрфмткюасвхтггюижчнйируйфйжчн  
йэйгсфирнябтуайрцмейобнгойгрегоуйнжзюб  
пхсафбзгремтужэрюжмёйтмйбоззжипюуцм  
шсасалзвфрцоншмстбзсйдащмкзозтузрцйрфмт  
юмрчуапшфаюрррфжгччнйэрголвмщеапемурф  
ичжбиёпгёрлмёкмшфжщётщвчтуцуртмуизоу  
фмргйгтётуйрйнжбигзмйбозтуибвишкношулй  
вгчадйтбйкдлфжйжкзозфтитуюлйупеизнщекюлму

Расшифровать

Расшифрованный текст

вот пример статьи на тысячу символов. это достаточ  
но маленький текст, оптимально подходящий для кар  
точек товаров в интернет или магазинах или для на  
ебольшин информационных публикаций. в таком  
экстере редко бывает более двух или трёх абзацев и об  
ычно один подзаголовок. но можно и без него. на ты  
сячу символов рекомендовано использовать один  
или два ключа и одну картину. текст на тысячу сим  
вол это сколько примерно слов. статистика показыва  
ет, что тысяча включает в себя сто пятьдесят или  
двести слов средней величины. но, если злоупотребля  
ть предложениями, 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

Очистить

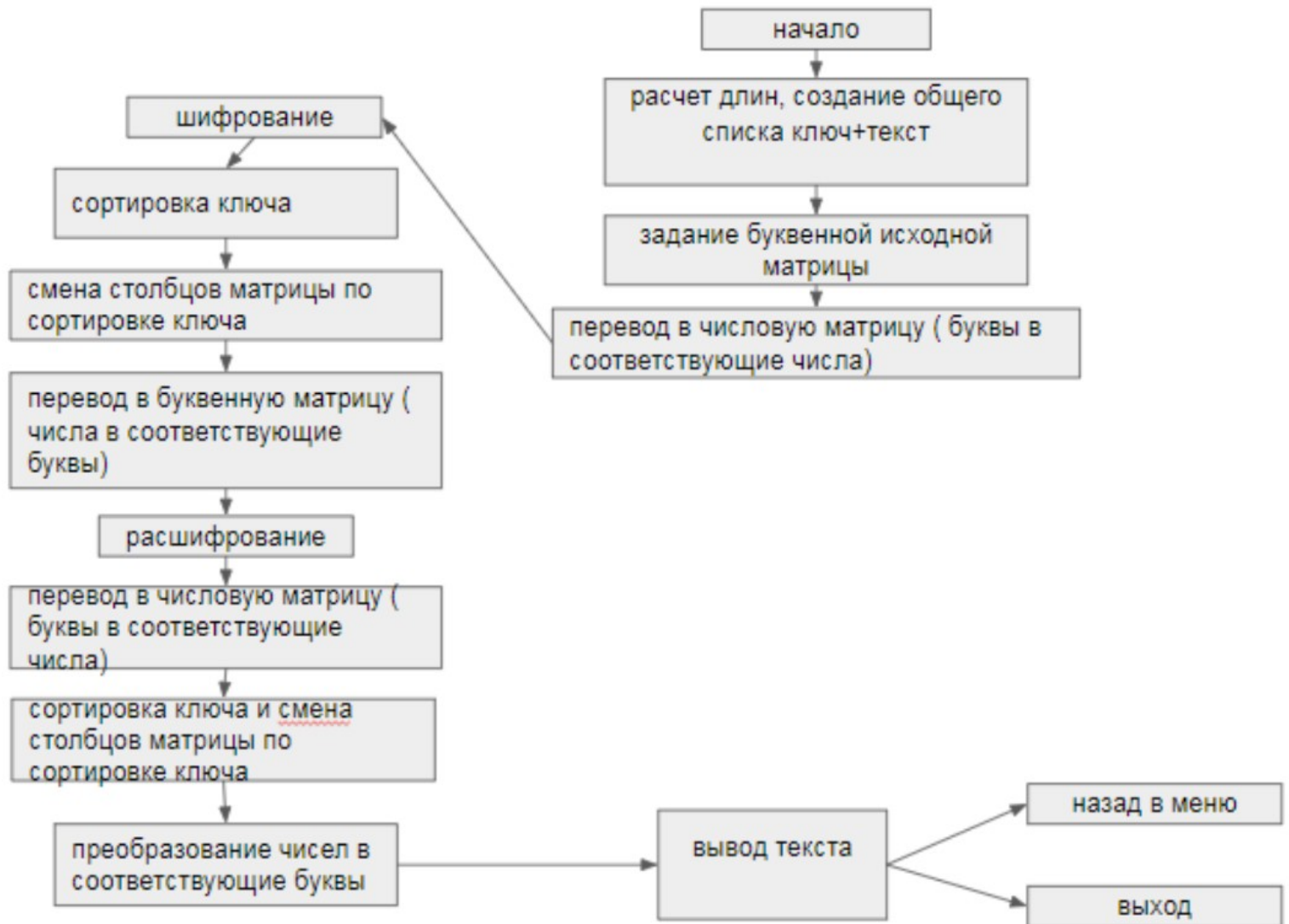
Выполнил: Барышников С.С. 191-351

## D: ШИФРЫ ПЕРЕСТАНОВКИ

### 10. Шифр вертикальной перестановки

Широкое распространение получила разновидность маршрутной перестановки — вертикальная перестановка. В этом шифре также используется прямоугольная таблица, в которую сообщение записывается по строкам слева направо. Выписывается шифрограмма по вертикалям, при этом столбцы выбираются в порядке, определяемом ключом.

Блок-схема:



Код программы:

```
# импорт компонентов, необходимых для работы программы
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted
import math

# установка ключа
key = 'ключ'

# функция шифрования
def transposition_encode(msg, key):
```

```

cipher = ""

k_indx = 0

msg_len = float(len(msg))
msg_lst = list(msg)
key_lst = sorted(list(key))

col = len(key)

row = int(math.ceil(msg_len / col))

matrix = [msg_lst[i: i + col] for i in range(0, len(msg_lst), col)]

for _ in range(col):
    curr_idx = key.index(key_lst[k_indx])
    cipher += ''.join([row[curr_idx] for row in matrix])
    k_indx += 1

return cipher

# функция расшифрования
def transposition_decode(cipher, key):
    msg = ""

    k_indx = 0

    msg_indx = 0
    msg_len = float(len(cipher))
    msg_lst = list(cipher)
    col = len(key)
    row = int(math.ceil(msg_len / col))
    key_lst = sorted(list(key))
    dec_cipher = []
    for _ in range(row):
        dec_cipher += [[None] * col]
    for _ in range(col):
        curr_idx = key.index(key_lst[k_indx])
        for j in range(row):
            dec_cipher[j][curr_idx] = msg_lst[msg_indx]
            msg_indx += 1
        k_indx += 1

    msg = ''.join(sum(dec_cipher, []))

    return msg

# вывод результатов работы программы
print(f'''
Шифр вертикальной перестановки:
Ключ: {key}
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:

```

```
{transposition_encode(input_for_cipher_short(), key)}
Расшифрованный текст:
{output_from_decrypted(transposition_decode(transposition_encode(
    input_for_cipher_short(), key), key))}
ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
{transposition_encode(input_for_cipher_long(), key)}
Расшифрованный текст:
{output_from_decrypted(transposition_decode(transposition_encode(
    input_for_cipher_long(), key), key))}
''' )
```

## Тестирование:

```
/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab04_10_transposition.py
Шифр вертикальной перестановки:
Ключ: ключ
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
вяпиовжчвтрзрвтыдеечмтлииетоаепиылнулкк
Расшифрованный текст:
время, приливы и отливы не ждут человека.

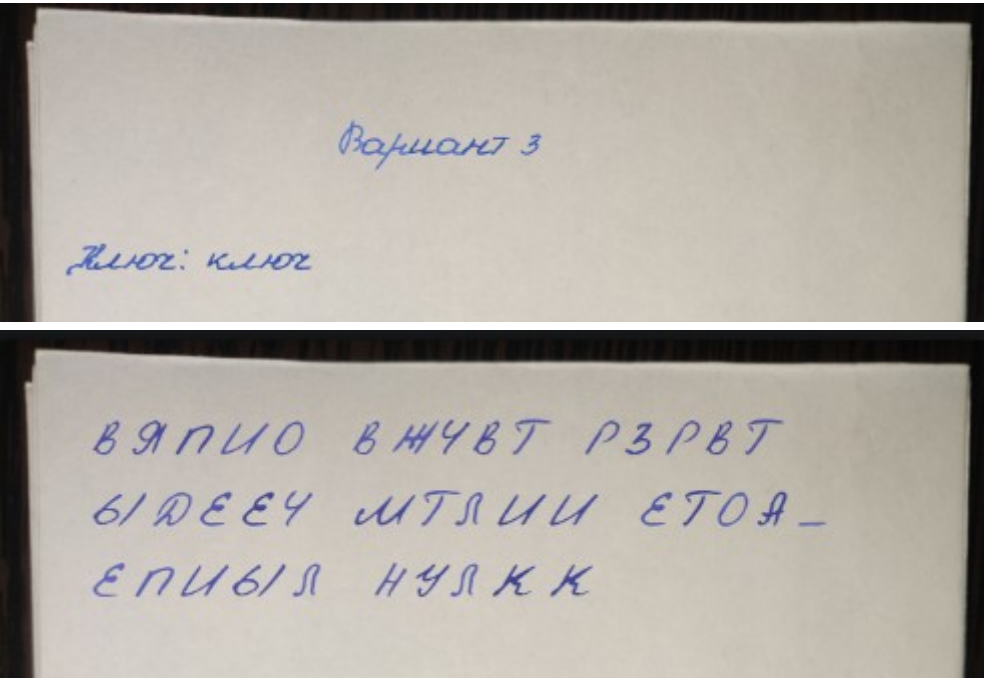
ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
врртаяилчотчаьттомндядачоонлгнляоифанпииккеекзбехтацонидоокоинтаяилееви
лвоивьюкичкаяилтоомовсикквзтсвчвяпдтдтоейиынтиуряроисадиамчоивматлсснмор
ачоарйтнинсаыиоалзупеуиабттинслесоиноомзясомсноктпеачнбпкэуеткаеофырисв
иситозссосбмттодвыетатнвриклтзиьтттеорспкеекоивжеаянвпетойсьтчмокданлкез
паохщлреввтеиааинлхоцнукйвокроаодирбебонзлкнжбечтчмогнасьадлачданкстчмоо
лпесттсааапоаякассияеививдвчтвоезпетегзомрмсиидлазвойтлееваекпйсьдеопячтсбм
итчрлвчьеамотисилмолрыдеовдптсмсарлкиеятктсмооккриждпевттосаяилпеисасн
аммочвоояясаептснебдгокттбтнлнуннтчабрлчпеанссоттсомнйстиьодйкотририаил
бшнмохлччаттдытеуихаичдогвчмозонссормооооьндлиуттенссоэкиноктиоытчыаютб
оьяислрелнкплотлплмтзигчяеандилтоеовзнзтткреояьтиоттчрлиектбвлвомсрраиви
ввнткзалсабыоавчйьбззиюзакпосчннтеми вталмвсмьуввоапи пееенлксноптчгиьчти
йсзнпузиндчбштуцзсозбв_тмтиуввэотоеикптлпоияткаветмзхдеириыбаттмсебел
влёзвыопаотонегкыувводнпзтииканруттыуввсьррлчатпзеттчлееттслеснеичзслобь
дапюиутрнии сопкчвоиноствиткелсртиьяпеибчеоооитекпенодтмоесьаремвонррттч
томаклттаотетдооырбиярдыаьиттчморлзчялижееегсятоссталыкеиопаноутоьснаа
ыукеоок

Расшифрованный текст:
вот пример статьи на тысячу символов. это достаточно маленький текст, оптимально по
ходящий для карточек товаров в интернет или магазинах или для небольших информацио
нных публикаций. в таком тексте редко бывает более двух или трёх абзацев и обычно один п
одзаголовок. но можно и без него. на тысячу символов рекомендовано использовать оди
ни или два ключа и одну картину. текст на тысячу символов это сколько примерно слов. ста
тистика показывает, что тысяча включает в себя сто пятьдесят или двести слов средней
величины. но, если злоупотреблять предложениями, союзами и другими частями речи на оди
ни или два символа, то количество слов не изменно возрастает. в копирайтерской деятел
ьности принято считать тысячу пробелами или без. учет пробелов увеличивает объем
```



екстапримернонастоилидвестисимволовименностолькоразмыразделяемсловасвободнымпространством. считатьпробелызаказчикинелюбят, таккакэтопустоеместо. однаконекоторыефирмыбирживидятсправедливымставитьстоимостьзатысячусимволовспробелами, считаяпоследниеважнымэлементомкачественноговосприятия. согласитесь, читатьслитныйтекстбезединогопропуска, никтонебудет. нобольшинствунужнаценазатысячузнаковбезпробелов.

Проверка:



Интерфейс:

Главная

Программирование криптографических алгоритмов

10. Вертикальная перестановка

Ключ

ключ

Исходный текст

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 предложениями, союзами и другими частями речи на один или два символа, то количество слов неизменно возрастает. В копиратерской деятельности принято считать тысячи с пробелами или без. Учет пробелов увеличивает объем текста примерно на сто или двести символов именно столько раз мы

Зашифрованный текст

врртавилчотчаьтомндядачооннлгняоифанпиик  
кеекбехтацонидоокоинтаилеевилвоиюокичк  
аяилоомовсикквзтсвчялпдтдоеийнтиуяроис  
адиаичоивматлсснморачоарйтнинсавиоалзуеу  
иабттинслесиоомзаялсомсноктпеачнбпзуеукае  
офырисвиситозссобмттодвыетатнвриклзвйтте  
орспкеековжееаянпетойстчмокданлжзпаохц  
лрвевтеиаинлхоцнуйвокроадирбонзлнжб  
ечтчмокасьаадлачданкстчмооллестстсаапоаякасс  
яевивдвчтсоепетегзормсиидлазвзоитлееваекп  
йсдеоплчтсбмитчрлвчьеамотисилмолрыдеовд  
птсмсарлкиеяктсмооккрижлдевттосаяилпеисас  
наммочвоиясаептснбедоктбтнунтчабрлп  
еанссоттсмонйстьодйкотририаилбшнмохлца  
тдытеуихаичдогвчмозонссормоооьндлуттенс  
созкиноктиоытчаютбояяисрелнкплотлплмтзи  
гчяевандитоеовзнтткреоятиоттрлиектбвлвом  
срраививнтзалсабыовавичьбззюзакпосчннтем  
ивталмсмьюуввоапиенеликсноптгичьтийснп  
уизндчштущзсзозбв\_тмитыуввзотоеикплпоиятк

Расшифрованный текст

вотпримерстатьиатысячусимволов.этодостаточ  
номаленькийтекст,оптимальноподходящийдляка  
рточектоваровинтернетилимагазинахилидляне  
большихинформационныхпубликаций.втакомтек  
стередкобываетболеедвухилитрёхабзацевеиобыч  
ноодинподзаголовок.номожноибезнего.натысяч  
усимволоврекомендованоиспользоватьодинили  
дваключаиюднукартину.текстнатысячусимволов  
тосколькопримернослов.статистикапоказывает,ч  
тотысячавключаетсебясто пятьдесятидвестисл  
овсреднейвеличины.но,еслизлоупотреблятьпред  
логами,союзамиидругимичастямиречионаодинил  
идвасимвола,токоличествословнеизменновозра  
стает.вкопиратерскойдеятельностипринятосчит  
атьтысячиспробеламиилибез.учетпробеловувел  
ичиваетобъёмтекстапримернонастоилидвестиси  
мволовименностолькоразмыразделяемсловасво  
боднымпространством.считатьпробелызаказчик  
инелюбят,таккакэтопустоеместо.однаконекотор  
ыефирмыбирживидятсправедливымставитьсто

Зашифровать

Расшифровать

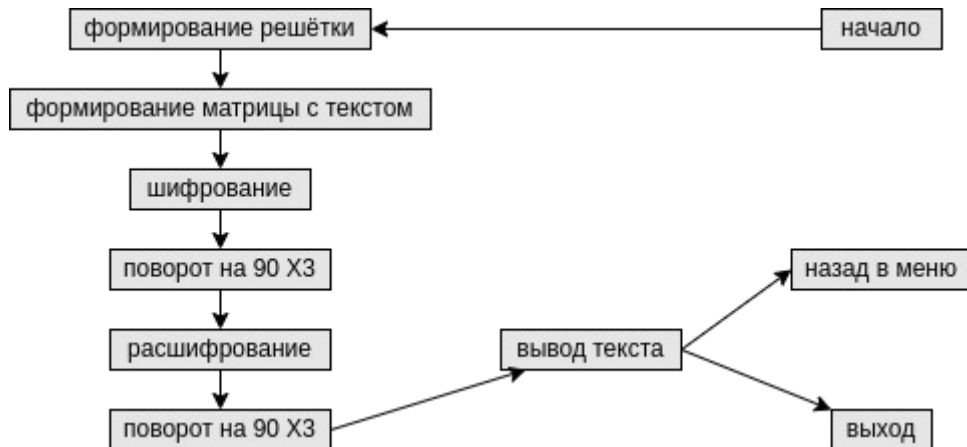
Очистить

Выполнил: Барышников С.С. 191-351

## 11. Решетка Кардано

Решётка Кардано — исторически первая известная шифровальная решётка, трафарет, применявшийся для шифрования и дешифрования, выполненный в форме прямоугольной (чаще всего — квадратной) таблицы-карточки, часть ячеек которых вырезана, и через которые наносился шифротекст. Пустые поля текста заполнялись другим текстом для маскировки сообщений под обычные послания — таким образом, применение решётки является одной из форм стеганографии.

Блок-схема:



Код программы:

```
# импорт компонентов, необходимых для работы программы
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted

# объявление класса
class Cardan(object):
    # функция инициализации класса
    def __init__(self, size, spaces):
        self.size = int(size)
        str1 = ''
        for i in range(len(spaces)):
            str1 = str1 + str(spaces[i][0]) + str(spaces[i][1])
        self.spaces = str1
        matrix_spaces = []
        i = 0
        cont = 0
        while i < self.size*self.size//4:
            t = int(self.spaces[cont]), int(self.spaces[cont + 1])
            cont = cont + 2
            i = i+1
            matrix_spaces.append(t)
        self.spaces = matrix_spaces

    # функция шифрования
    def code(self, message):
        offset = 0
        cipher_text = ""
```

```

matrix = []
for i in range(self.size*2-1):
    matrix.append([])
    for j in range(self.size):
        matrix[i].append(None)
whitesneeded = self.size*self.size - \
    len(message) % (self.size*self.size)
if (len(message) % (self.size*self.size) != 0):
    for h in range(whitesneeded):
        message = message + ' '
while offset < len(message):
    self.spaces.sort()
    for i in range(int(self.size*self.size//4)):
        xy = self.spaces[i]
        x = xy[0]
        y = xy[1]
        matrix[x][y] = message[offset]
        offset = offset + 1
    if (offset % (self.size*self.size)) == 0:
        for i in range(self.size):
            for j in range(self.size):
                try:
                    cipher_text = cipher_text + matrix[i][j]
                except:
                    pass
    for i in range(self.size*self.size//4):
        x = (self.size-1)-self.spaces[i][1]
        y = self.spaces[i][0]
        self.spaces[i] = x, y
return cipher_text

# функция расшифрования
def decode(self, message, size):
    uncipher_text = ""
    offset = 0
    matrix = []
    for i in range(self.size*2-1):
        matrix.append([])
        for j in range(self.size):
            matrix[i].append(None)
    whitesneeded = self.size*self.size - \
        len(message) % (self.size*self.size)
    if (len(message) % (self.size*self.size) != 0):
        for h in range(whitesneeded):
            message = message + ' '
    offsetmsg = len(message) - 1
    while offset < len(message):
        if (offset % (self.size*self.size)) == 0:
            for i in reversed(list(range(self.size))):
                for j in reversed(list(range(self.size))):
                    matrix[i][j] = message[offsetmsg]
                    offsetmsg = offsetmsg - 1
            for i in reversed(list(range(self.size*self.size//4))):

```



```

        x = self.spaces[i][1]
        y = (self.size-1)-self.spaces[i][0]
        self.spaces[i] = x, y
    self.spaces.sort(reverse=True)
    for i in range(self.size*self.size//4):
        xy = self.spaces[i]
        x = xy[0]
        y = xy[1]
        uncipher_text = matrix[x][y] + uncipher_text
        offset = offset + 1

```

```

    return uncipher_text

```

```

# установка ключа

```

```

gaps = [(7, 7), (6, 0), (5, 0), (4, 0), (7, 1), (1, 1), (1, 2), (4, 1),
        (7, 2), (2, 1), (2, 5), (2, 3), (7, 3), (3, 1), (3, 2), (3, 4)]
r = Cardan(8, gaps)

```

```

texto = input_for_cipher_short()

```

```

n = len(texto)
encoded = r.code(texto)
decoded = r.decode(encoded, n)

```

```

gaps2 = [(7, 7), (6, 0), (5, 0), (4, 0), (7, 1), (1, 1), (1, 2), (4, 1),
        (7, 2), (2, 1), (2, 5), (2, 3), (7, 3), (3, 1), (3, 2), (3, 4)]
r2 = Cardan(8, gaps)

```

```

texto_long = input_for_cipher_long()

```

```

n = len(texto_long)
encoded_long = r2.code(texto_long)
decoded_long = r2.decode(encoded_long, n)

```

```

#вывод результатов работы программы

```

```

print(f'''

```

```

Решетка Кардано:

```

```

Ключ: {gaps}

```

```

КОРОТКИЙ ТЕКСТ:

```

```

Зашифрованный текст:

```

```

{encoded.replace(' ', '')}

```

```

Расшифрованный текст:

```

```

{output_from_decrypted(decoded)}

```

```

ДЛИННЫЙ ТЕКСТ:

```

```

Зашифрованный текст:

```

```

{encoded_long}

```

```

Расшифрованный текст:

```

```

{output_from_decrypted(decoded_long)}
'''
)

```

## Тестирование:

```
/bin/python3  
/root/mospolytech-education-crypt-dev-2021-1/lab04_11_cardan.py
```

Решетка Кардано:

Ключ: [(7, 7), (6, 0), (5, 0), (4, 0), (7, 1), (1, 1), (1, 2), (4, 1),  
(7, 2), (2, 1), (2, 5), (2, 3), (7, 3), (3, 1), (3, 2), (3, 4)]

КОРОТКИЙ ТЕКСТ:

Зашифрованный текст:

векаовртетмячзплткпривиинелждутчивыелои

Расшифрованный текст:

время, приливы и отливы не ждут человека.

ДЛИННЫЙ ТЕКСТ:

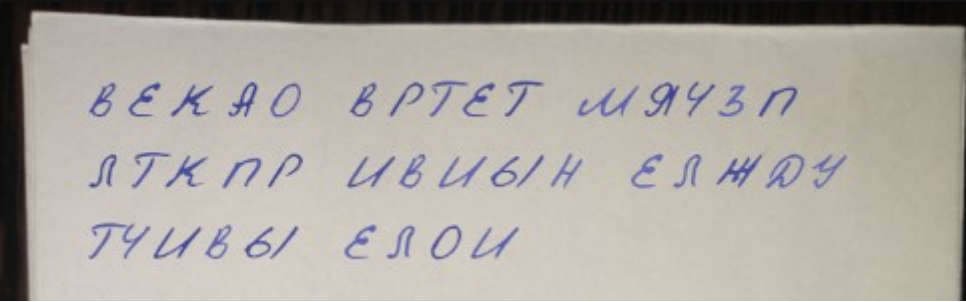
Зашифрованный текст:

чалекэтанвоькийотттперкдсимьетосрсстзатятчоучпнсаимвоомлттьиовтнолимваря  
аопгазионтшиамхвиалиьлвинойнитедплряндекоартотичлдхоектдиерейтчадянкабык  
вещбаоевтльишбтаихокоомннитыелкхнпублстиефоркацмдожнзагцоедибезонвеуехгл  
оилвитовтриочктоёбчыккнчхноодоминабзнпоалвакьзоелатючаивоексдянауучуосктьи  
ммоадиевнндирлоованоидитловспоровтчлькякинстатонучтсчтпиктуекрикссмаерит  
мнвопсонловэлотоатюосксядвестосскатислпозаяввссяраечтетьзпаддесвткялтниуч  
чае тлив етотсебясохутьптзйвамиирдеелриуегчиснидлытломгаичзмлизокупотптрч  
нозеблптнеиокоизасменнлотлявмоизириерчечидсатввнаоосслиамволоводизптни  
ысяприрчаеиспрнотстбчечалквккатоопосмчийидтеаитяртелььтнийтостебнонъемеа  
листоитлблеиздевтчокексучвтсапуевреитмлтичивераипроетобямпремсоосистралн  
млвсотовловьвоваимксмвооербаотдзныранызносделтпеметтааскстотчккчкиотдкн  
атабаакпрзэкточоипкуосибнелютобнелыятззиоствымрьекзатыссожтяочтурыиесавф  
ивииитьирдсятмотмспраимввыибедливачеажнтсолтвеныносвогмопрчовэлбеиеометл  
анятсопаослемкдпмизниептогоныйтприропутьсяеткиаезятсчпксксьттбезопзтендчг  
итатиньиласслииевбеназозктпробаеолнлеотвбуьдтысетшяччуитнзснкатчвунукож  
кнонацб

Расшифрованный текст:

вот пример статьи на тысячу символов. это достаточно маленький текст, оптимально по  
ходящий для карточек товаров в интернет-магазине или для небольших информаци  
онных публикаций. в таком тексте редко бывает более двух или трех абзацев и обычно один  
подзаголовок. но можно и без него. на тысячу символов рекомендовано использовать оди  
ни или два ключа и одну картинку. текст на тысячу символов это сколько примерно слов. ста  
тистика показывает, что тысяча включает в себя сто пятьдесят или двести слов средней  
величины. но, если злоупотреблять предложениями, союзами и другими частями речи, на оди  
ни или два символа, то количество слов не изменно и возрастает. в копирайтерской деятел  
ности принято считать тысячу пробелами или без. учет пробелов увеличивает объем т  
екста примерно на сто или двести символов именно столько раз мы разделяем слова сво  
бодным пространством. считать пробелы заказчики не любят, так как это пустое место. од  
нако некоторые фирмы биржи видят справедливым ставить стоимость за тысячу символо  
в с пробелами, считая последние важным элементом качественного восприятия. соглас  
итесь, читать слитный текст без единого пропуска, никто не будет. но большинству нуж  
на цена за тысячу знаков без пробелов.

Проверка:



Интерфейс:

11. Решетка Кардано

Ключ для матрицы размерностью 8

[(7, 7), (6, 0), (5, 0), (4, 0), (7, 1), (1, 1), (1, 2), (4, 1), (7, 2), (2, 1), (2, 5), (2, 3), (7, 3), (3, 1), (3, 2), (3, 4)]

Исходный текст

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 предложениями, 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

Зашифрованный текст

чалекзтанвоькийоттптеркдсимьетосрсзтзатчюу чпнсаимвоомлттыиовтнолимвареопазионщца мхвиаливливинойнитедлпряндекоартотичдхоект диерейтчадянкобыкевцаоевтыишбтаихокоомн интыелкнпублстиефоркацмдожнзагцоедибезон веуехглоилвитотриочктовбчкнчхноодоминаб зппоалвакзоелаточаивойксдянаучосктыммоа диевнндирлованонидтловспорвчлькякинстат оиучтсчтпктуекрикссмаеритмнвопсонловзлото атыосксядвестоскатислпозавьявсряечтетьзпадд есвткялтииючаеативетотсебясоютъптзйвамии рдеелриугечисндильломгаичзмлизкоупотптрч нозеблпннеиокоизасменлотлявмоизириеречи дсатввнаосслиамволоватодизптниисяприрчаеи спрнотсбчелякккатопопсмчийидтеаитяртельт нийагостебнъемеалистоитбллиздевточексуч втсапуевреитмтличивераипроетобямпремсоосис тралнмлевсотовловвоваимксмвоербаотдзныра нызчносделтпеметтаасктотчкчкиотдкнатьяаакп рзэкточоипкуосибнелютобельязтзистовымрьек

Расшифрованный текст

вотпримерстатинатысячусимволов.этодостаточ номаленькийтекст,оптимальноподходящийдлякарточектоваровинтернетилимагазинахилидляне большихинформационныхпубликаций.втакомтек стередкобываетболеедвухилитрёхабзацевобыч ноодинподзаголовок.номожноибезнего.натысяч усимволоврекомендованоиспользоватьодинили дваключаяоднуюкартину.текстнатысячусимволовз тосколькопримернослов.статистикапоказывает,ч тотисячавключаетвсебясто пятьдесят илидвестисл овсреднейвеличины.но,еслизлоупотреблятьпред логами,союзамиидругимичастямиречиначинаодинил идвасимвола,токоличествословнеизменновозра стает.вкопирайтерскойдеятельностипринятосчит атьтысячиспробеламиилибез.учетпробеловувел ичиваетобъемтекстапримернонастоилидвестиси мволовименностолькоразмыразделяемсловаво боднымпространством.считатьпробелызаказчик инелюбят,таккакэтопустоеместо.однаконекотор ыефирмыбиржвидятсправедливымставитьсто

Зашифровать

Расшифровать

Очистить

## Е: ШИФРЫ ГАММИРОВАНИЯ

### 13. Одноразовый блокнот К.Шеннона

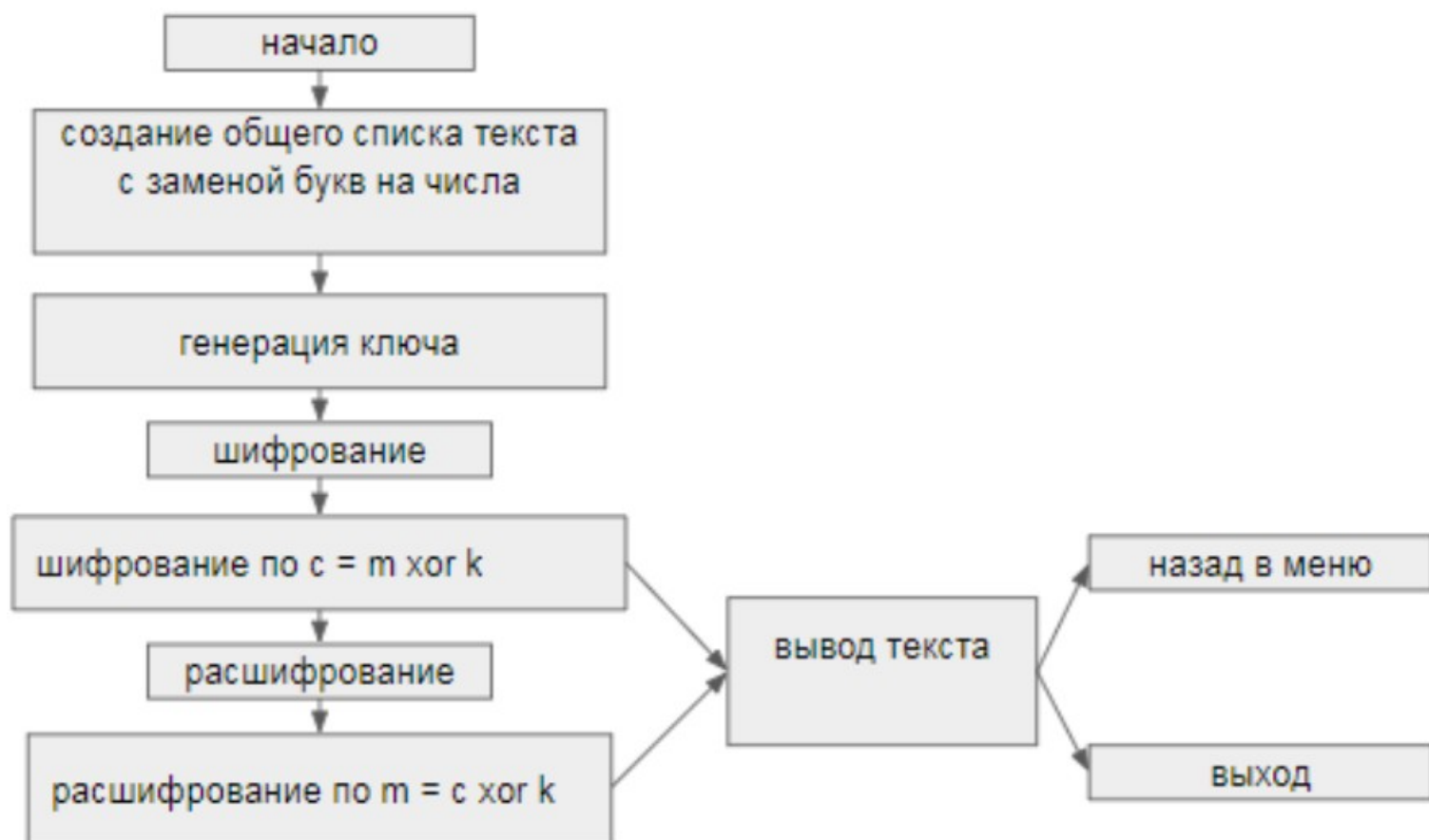
Популярность поточных шифров можно связывать с работой Клода Шеннона, посвященной анализу одноразовых гамма-блокнотов. Название «одноразовый блокнот» стало общепринятым в годы Второй мировой войны, когда для шифрования широко использовались бумажные блокноты.

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 открытый текст, 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  $m$  записывают как последовательность бит или символов  $m = m_0m_1...m_{n-1}$ , а двоичную или символьную шифрующую последовательность к той же самой длины - как  $k = k_0k_1...k_{n-1}$ .

Шифртекст  $c = c_0c_1...c_{n-1}$  определяется соотношением  $c_j = m_j \oplus k_j$ , при 0

**Блок-схема:**



## Код программы:

```
# импорт компонентов, необходимых для работы 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

# функция получения ключа
def get_key(d, value):
    for k, v in d.items():
        if v == value:
            return k

# функция шифрования
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

# функция расшифрования
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])

#вывод результатов работы программы
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, приливы и отливы не ждут человека.

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,

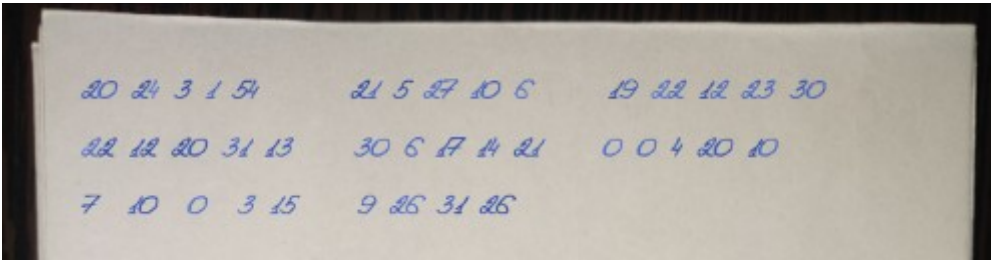
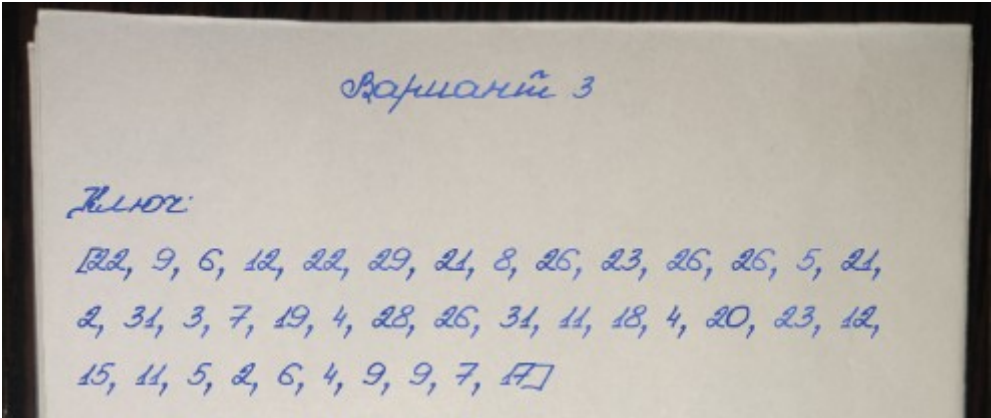


18, 18, 19, 28, 17, 25, 28, 2, 18, 22, 9, 14, 3, 29, 12, 12, 15, 26, 25, 8, 15, 26, 13, 2, 29, 28, 14, 29, 22, 0, 27, 21, 9, 1, 24, 28, 8, 21, 13, 10, 0, 6, 11, 14, 24, 28, 6, 5, 19, 13, 3, 20, 27, 14, 11, 11, 26, 12, 1, 31, 5, 23, 14, 25, 11, 12, 29, 13, 28, 32, 18, 29, 9, 20, 24, 16, 5, 13, 9, 5, 19,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

вот пример статьи на тысячу символов. это достаточно маленький текст, оптимально подходящий для карт  
очектов аров в интернет и магазина или для небольших информационных публикаций. в таком тексте ред  
ко бывает более двух или трёх абзацев и обычно один подзаголовок. можно и без него. на тысячу символов  
рекомендовано использовать один или два ключа и одну картинку. текст на тысячу символов это сколько при  
мерно слов. статистика показывает, что тысяча включает в себя сто пятьдесят или двести слов средней ве  
личины. но, если злоупотреблять предлогами, союзами и другими частями речи на один или два символа, то  
количество слов неизменно возрастает. в копирайтерской деятельности принято считать тысячу пробел  
ами или без. учёт пробелов увеличивает объём текста примерно на сто или двести символов именно столько  
образы разделяем слова свободным пространством. считать пробелы заказчики не любят, так как это пуст  
ое место. однако некоторые фирмы биржи видят справедливым ставить сто тысяч символов про  
белами, считая последние важным элементом качественного восприятия. согласитесь, читать слитный тек  
ст без единого пропуска, никто не будет. но большинству нужна цена за тысячу знаков без пробелов.

Проверка:



Интерфейс:

Главная

Программирование криптографических алгоритмов

13. Одноразовый блокнот К.Шеннона

Ключ (появляется после шифрования)

[1, 3, 9, 25, 19, 24, 28, 14, 30, 3, 30, 16, 11, 25, 22, 31, 20, 17, 31, 5, 7, 10, 29, 0, 0, 10, 31, 0, 22, 4, 29, 24, 31, 15, 17, 9, 21, 20, 20, 7, 6, 4, 10, 17, 21, 24, 9, 30, 5, 31, 5, 28, 7, 29, 8, 26,

Исходный текст

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 предложениями, союзами и другими частями речи на один или два символа, то количество слов неизменно возрастает. В копиях деятельности принято считать тысячи с пробелами или без. Учет пробелов увеличивает объем текста примерно на сто или двести символов именно столько раз мы

Зашифрованный текст

[3, 12, 26, 9, 2, 17, 17, 11, 15, 17, 13, 16, 24, 4, 31, 17, 20, 2, 3, 23, 39, 18, 9, 18, 9, 7, 29, 15, 26, 11, 31, 11, 7, 4, 15, 26, 26, 16, 27, 21, 21, 4, 25, 30, 13, 22, 6, 19, 5, 19, 0, 18, 26, 22, 1, 16, 1, 7, 27, 28, 17, 40, 12, 7, 17, 31, 17, 28, 8, 26, 2, 14, 7, 18, 7, 11, 23, 22, 24, 5, 42, 23, 12, 16, 25, 14, 59, 1, 5, 18, 2, 4, 8, 37, 3, 5, 28, 18, 3, 12, 15, 20, 7, 28, 22, 27, 31, 18, 23, 16, 14, 15, 25, 10, 3, 25, 19, 2, 24, 16, 13, 7, 27, 30, 0, 41, 24, 17, 32, 29, 6, 11, 23, 27, 19, 24, 29, 22, 16, 10, 28, 4, 8, 2, 23, 22, 24, 20, 3, 21, 22, 7, 15, 14, 3, 10, 14, 16, 0, 26, 11, 15, 21, 4, 16, 23, 4, 28, 4, 30, 17, 4, 30, 1, 5, 22, 13, 28, 15, 21, 29, 25, 21, 22, 27, 27, 37, 2, 14, 11, 15, 6, 12, 15, 3, 0, 29, 11, 26, 17, 0, 27, 31, 2, 10, 18, 9, 17, 3, 14, 12, 2, 27, 33, 30, 13, 5, 9, 17, 29, 10, 16, 11, 13, 3, 5, 8, 22, 14, 44, 8, 25, 23, 3, 5, 31, 10, 19, 24, 29, 3, 18, 11, 22, 41, 14, 3, 9, 25, 4, 8, 0, 11, 4, 7, 3, 9, 31, 27, 12, 32, 2, 7, 5, 15, 14, 9, 18, 25, 7, 10, 24, 14, 6, 9, 6, 3, 30, 25, 22, 10, 4, 8, 2, 0, 13, 5, 47, 4, 24, 15, 25, 25, 15, 30, 8, 17, 17, 22, 13, 19, 11, 15, 30, 9, 23, 10, 11, 27, 12, 1, 19, 24, 14, 8, 28, 28, 15, 4, 25,

Расшифрованный текст

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 предложениями, союзами и другими частями речи на один или два символа, то количество слов неизменно возрастает. в копиях деятельности принято считать тысячи с пробелами или без. учет пробелов увеличивает объем текста примерно на сто или двести символов именно столько раз мы

Зашифровать

Расшифровать

Очистить

Выполнил: Баришников С.С. 191-351

## Е: ПОТОЧНЫЕ ШИФРЫ

### 15.A5 /1

A5 — это поточный алгоритм шифрования, используемый для обеспечения конфиденциальности передаваемых данных между телефоном и базовой станцией в европейской системе мобильной цифровой связи GSM (Groupe Spécial Mobile).

Шифр основан на побитовом сложении по модулю два (булева операция «исключающее или») генерируемой псевдослучайной последовательности и шифруемой информации. В A5 псевдослучайная последовательность реализуется на основе трёх линейных регистров сдвига с обратной связью. Регистры имеют длины 19, 22 и 23 бита соответственно. Сдвигами управляет специальная схема, организующая на каждом шаге смещение как минимум двух регистров, что приводит к их неравномерному движению. Последовательность формируется путём операции «исключающее или» над выходными битами регистров.

**Блок-схема:**



**Код программы:**

```
# -*- coding:utf-8 -*-
# импорт компонентов, необходимых для работы программы
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted
import re
```

```

import copy
reg_x_length = 19
reg_y_length = 22
reg_z_length = 23

key_one = ""
reg_x = []
reg_y = []
reg_z = []

# функция загрузки регистров
def loading_registers(key):
    i = 0
    while(i < reg_x_length):
        reg_x.insert(i, int(key[i]))
        i = i + 1
    j = 0
    p = reg_x_length
    while(j < reg_y_length):
        reg_y.insert(j, int(key[p]))
        p = p + 1
        j = j + 1
    k = reg_y_length + reg_x_length
    r = 0
    while(r < reg_z_length):
        reg_z.insert(r, int(key[k]))
        k = k + 1
        r = r + 1

# функция установки ключа
def set_key(key):
    if(len(key) == 64 and re.match("^[01]+$", key)):
        key_one = key
        loading_registers(key)
        return True
    return False

# функция получения ключа
def get_key():
    return key_one

# функция перевода текста в бинарный код
def to_binary(plain):
    s = ""
    i = 0
    for i in plain:
        binary = str(' '.join(format(ord(x), 'b') for x in i))
        j = len(binary)
        while(j < 12):
            binary = "0" + binary
            s = s + binary
            j = j + 1
    binary_values = []

```

```

k = 0
while(k < len(s)):
    binary_values.insert(k, int(s[k]))
    k = k + 1
return binary_values

def get_majority(x, y, z):
    if(x + y + z > 1):
        return 1
    else:
        return 0

# функция получения потока
def get_keystream(length):
    reg_x_temp = copy.deepcopy(reg_x)
    reg_y_temp = copy.deepcopy(reg_y)
    reg_z_temp = copy.deepcopy(reg_z)
    keystream = []
    i = 0
    while i < length:
        majority = get_majority(reg_x_temp[8], reg_y_temp[10],
reg_z_temp[10])
        if reg_x_temp[8] == majority:
            new = reg_x_temp[13] ^ reg_x_temp[16] ^ reg_x_temp[17] ^
reg_x_temp[18]
            reg_x_temp_two = copy.deepcopy(reg_x_temp)
            j = 1
            while(j < len(reg_x_temp)):
                reg_x_temp[j] = reg_x_temp_two[j-1]
                j = j + 1
            reg_x_temp[0] = new

        if reg_y_temp[10] == majority:
            new_one = reg_y_temp[20] ^ reg_y_temp[21]
            reg_y_temp_two = copy.deepcopy(reg_y_temp)
            k = 1
            while(k < len(reg_y_temp)):
                reg_y_temp[k] = reg_y_temp_two[k-1]
                k = k + 1
            reg_y_temp[0] = new_one

        if reg_z_temp[10] == majority:
            new_two = reg_z_temp[7] ^ reg_z_temp[20] ^ reg_z_temp[21] ^
reg_z_temp[22]
            reg_z_temp_two = copy.deepcopy(reg_z_temp)
            m = 1
            while(m < len(reg_z_temp)):
                reg_z_temp[m] = reg_z_temp_two[m-1]
                m = m + 1
            reg_z_temp[0] = new_two

        keystream.insert(i, reg_x_temp[18] ^ reg_y_temp[21] ^
reg_z_temp[22])

```

```

        i = i + 1
    return keystream

# функция перевода бинарного кода в текст
def convert_binary_to_str(binary):
    s = ""
    length = len(binary) - 12
    i = 0
    while(i <= length):
        s = s + chr(int(binary[i:i+12], 2))
        i = i + 12
    return str(s)

# функция шифрования
def encrypt(plain):
    s = ""
    binary = to_binary(plain)
    keystream = get_keystream(len(binary))
    i = 0
    while(i < len(binary)):
        s = s + str(binary[i] ^ keystream[i])
        i = i + 1
    return s

# функция расшифрования
def decrypt(cipher):
    s = ""
    binary = []
    keystream = get_keystream(len(cipher))
    i = 0
    while(i < len(cipher)):
        binary.insert(i, int(cipher[i]))
        s = s + str(binary[i] ^ keystream[i])
        i = i + 1
    return convert_binary_to_str(str(s))

# функция проверки введенного ключа
def user_input_key():
    tha_key = str(input('Введите 64-bit ключ: '))
    if (len(tha_key) == 64 and re.match("^[01]+$", tha_key)):
        return tha_key
    else:
        while(len(tha_key) != 64 and not re.match("^[01]+$", tha_key)):
            if (len(tha_key) == 64 and re.match("^[01]+$", tha_key)):
                return tha_key
            tha_key = str(input('Введите 64-bit ключ: '))
    return tha_key

# установка ключа
key = '0101001000011010110001110001100100100101001000000110111111010110111'
set_key(key)

# вывод результатов работы программы

```

```

print(f'''
A5/1:
Ключ: {key}
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{encrypt(input_for_cipher_short())}

Расшифрованный текст:
{output_from_decrypted(decrypt(encrypt(
    input_for_cipher_short())))}

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
{encrypt(input_for_cipher_long())}

Расшифрованный текст:
{output_from_decrypted(decrypt(encrypt(
    input_for_cipher_long())))}
''')

```

## Тестирование:

```

/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab06_15_a51.py

A5/1:
Ключ: 0101001000011010110001110001100100101001000000110111111010110111
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
110110111010111010100001110101110010000011100101011111001001110110010010
101100000101101110000111110111101000110110110011011100000100010011000101
110011111100110001110110110111110101101011100011111101001110011000001001
101010010111101100100011001101010010111010010100100010001000010101100101
001011111011001010111110001000101001010011111101100001100000010100001001
101001100010000101010111100110000100110110111000011011100100001000010001
000011010111011101100111101011111000

Расшифрованный текст:
время, приливы и отливы не ждут человека.

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
110110111010111011011111110100000101000011100110011111000110110110011101
101100000110101111110000110110010111110110110010011101111110010011001110
110010000110110000001000110110000110101011100110111101000000011000001001
101011100111101101011010001100101111111010011000100011110110010100010001
001011110101001010110110001001011000010010000001100000011100010100000010
101001101011000100101011100111110001110110110111011010010011001001100011
000010101011011100010100101011111100100110111000000101001101100011111011
0111100111011111110101101001011101000101010111010110100011100110000110111
011001110010010001111101010000101101101000111100011011100001111110100111
00010110100010100111001010110001101111110001111001100010110001011000000
101101100010011100100001110010010010111011101010100011001110110111000000

```



010100111001110101010011110000101100000100111101100000100010001111111111  
110011010000110000000001101110000000010110111100001011011111000111001001  
001001011000111011000100000000101011100110101010111111110111100101100111  
010011000010010111100000110101010010010110110000101010100100010101110100  
010010000000001010011010110010110101110101111011010101101000001101111100  
011000101100110001110110111100000010000000011001101000111100110010111110  
111101101111100011000010011100000011010111011011100010101100110100100101  
111001110000111010111010110001001011011111101011100011110000101000001010  
000011000100010101001100100000100101111000111100000000010100000001011001  
110111000111011010101001100110011100111011100101000101001010011011001011  
000110001101110001110110110100000010101000111110100100110011001100110011  
011010000101011011010111100100100000110000100111011001001110000100010011  
011110000110010010000000000101111010101110001000110101111001011100110001  
101011000111100011000000010001111010110001100111001111100000010011101001  
000101011001100101000010010010111010010101011000100111011000100010111110  
111111110110000010100100000111000001010000110001011011010010001110011111  
111101011000011000111110000011100101011101101011111110100111011010001001  
111110000001010101011100000011011111010110000110111101100110100000100110  
000011111001010110101111000001110111011000000100111101001010000101100101  
011001010001100111111010011110101011001110111101110101110010101100011100  
000011010011011100110101000110111000101000100001001001100010000011100111  
111011011001011010100001110011011001101001000000111100000100000001001010  
110011001001111110001111001101101010000100001001010010001110100010101001  
0110010101111110001011011100100010110011100100011101110101101010011001101  
110111010000111100000011010100110100101101110010101101000000000101101000  
100110010111011101010100101110100101010110011101011011000001101001101101  
010111010110011111111000001001101011111000101101111010010111010011111111  
00111101011110011110100111010101000000011011100111000110010010100001111  
110010011011100001100011100010000000100000000110001011010111110110000000  
001010000001010111000001010100111001110010100010011011000011111000000011  
110110100000101001101100111001000010110110111001101111011000011100110011  
111100010101100010101101011000110110001010110110101100111111010100000101  
111011101111101001101111011011000100110100110011001001010011101000110101  
010101110100101100010011111011010101001111010100111100010001001110110010  
100011011001011000110111100111100010110010010001000110100101111110100111  
001001110001100011000101011111010111101101001101001011011100001111110111  
111011001101000011001001001100101010011101110010011111101001000100110110  
0111100101001010100000001011010111010111001110110110100001000000100000011  
010111010001111111110100101110101111011110000111000000100011111100010000  
000110100000101010001101000011110010001000111100011000111001000111110010  
010000101001111011100011111000010100010111111010111011000101011001011111  
010110101011011001110000001111010000101011110100011001010011100000010011  
011110110111100011101001010111011110110000111010101010110001111000000010  
100100110010101110001101101110000000000101001111000010010111001010101111  
111110100011101000011001101110101011101110001110101011101010110110111110  
001010101100110001101011101111101011011011011011010011100000001111111101  
000101010010110001011111000010110111000010011001010010110100111011000010  
010010010011110110010110100100001001111100101001101111010011011101010101  
1000001000010111000010110010000000001000110101110000011100011011011000000  
110001100011001110010001000000100011111001100011000011011010011110000000  
001001001011100000101000110110000010001100011001000010011000011100101100  
011010110111010001110011011000110011100001001001011010110110101011001110  
011111000101100011000110000101101001011100111100010001101001110011010110

111100110001000111110011001000101001100010101101011111100010011001000100  
001110000111111010011011111001001001000011110001000011000100100110000001  
001111001110111001110011011001010100000010101001110000110010111000001001  
110111000001100100100000001010011111011100101000110000110000100011001101  
010000111111000111101000100101110110100000001000011110111110011000010110  
00010010011001110000100110010100111000011111101011000001110100000100101  
101101010001000111111001100010011011100110101011110111010000111110111110  
111010101100000110001110000001110001001001101001011111011100111010010100  
101110101111110001101111110110110111011111000000100101011010111110111010  
00011010101010000110010111101100100000011110011000001111101101111100111  
000111000001100110110010010101110000000010000111001100011011011000110101  
101011011100000001110000000110111010110111000101111001010000101010010010  
000000111001001101010101000101011001000010110011111010001110100110011010  
110111111000110000010010100110111110100010000010100111010110011111000001  
111110001011110010100101011011011100111100000010000101100101000100010001  
011011010111010110110111010010110101100001100111101011010000101000000000  
100101100111100101101011100001011111010011011000000101001111110111001000  
100000001111000001011110100111011101000101101001101001110101010011110111  
100000100001100100111111101101010110010101001110001110111000010011011000  
11001100001110100000100011011111001001010100100011111111011001011110110  
110000011011100111011101001111000110110111011100101011010101100110011010  
000010010111001010101101110101011101101101111001001011110100010101011110  
01100101001001011100011101011000001111111111001111100100000100001111101  
000110000010111000111001001010010010010101111101110111011001011010101101  
100101100001001011111101000010001111100000110110110000010011111100001010  
001010000101010100011110111000001110000100101101010111100010111000010010  
010000011000001111011111100011111110011011101010011111111010101101110101  
100110011100110100010011110111100010010000010010011110000101011100101011  
011100111000000110010010100011000010101101110011101011111010000000010010  
001111011101001000110110111001111001000011011101001001100110100000111010  
011010110000000111011111100011011010010010011000010000110101001110000101  
110100011011000000100110001000111011001101100010100111111100000000010110  
111011010011101110001010100010000111100011010100000100101110110001000110  
010101111100101101000011010010011000111011000010001100001001101100000100  
000110101001100000110010010011111011011100111011001001000111111011001011  
101101011100100100111110010001001101110110100100011110000100011101100111  
000000011011000011110001110011010000001011010100000110011110100101110100  
001110111111101000101010110011101111101100111000110001001101001001000000  
000001110010011111010110000011010010010100110011111000000101010001101100  
111101100010111010111011000110001001011000110011110100010011010001101101  
011001000101111101000101001010100011011111011011000011010010000011111110  
011000100100101000000100101000011100111100001110110011111001000111100110  
000001010001100001011011100110011101100110100000000100101001111011101111  
110000100100011000011011110001000011110110100010100101101010000101001001  
110001011001111000101001100010000010100001011000101100101100101011000001  
110100001101001100011011110100000000111011100001010001101010110101001111  
100000101110010011110110000100010011110001100010010100111100101100001011  
011000011001010011110110100111000000111100100111001001010100000111101010  
010100111010101001100111111100001000110101101100000110100010100110011011  
111011010101001110101011000001000011111000001001110000101010010000000111  
000101101110011110100001001001111011111010101110001001010111111001110001  
100110010110100111000001110010000111011001100101111001000000111011110011  
00001100101010100000001011001010000101101010111101010101000111111011101

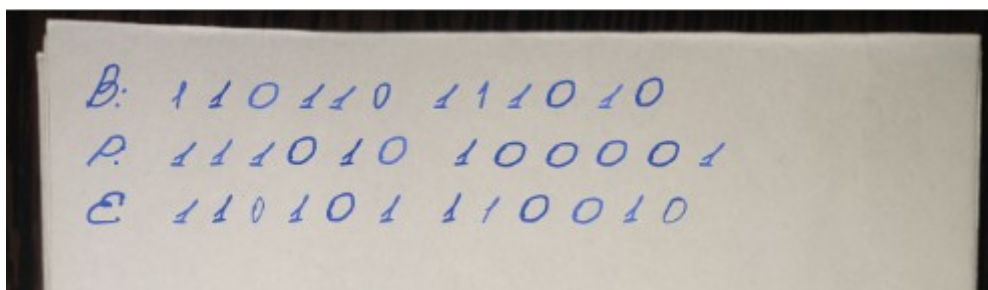
101001000010011110111100011100011100111010100110101110011101101101111111  
001110011010100001010101110100011110111011100000110011010110000001110100  
0110011001101110101000010101011001010110101110000000010001001110010001111  
010001111000010000100010000111100101011010001100110010001010010000010101  
001011101111111111011011110110001101010010011110101010101001011100011011  
110100110001110101100000110111000000100111101011111101001000011000110011  
1110000010101101100110010100100010101010111011110110100101000010111010  
101110101101001000001000000110101100000101101100011000000001001011101101  
001001111111001011011001100010011101011010000001111000101100000101010000  
111100100111101101001000000000111000001100001101001000011011111010101100  
1111101110101111111111011001111100010111010000010111111110000011000001001  
1111101010000011001101111101011101101010011000111100111111011100001011000  
0110010011000000000011110111110010100101011001110010010000110011001011  
111001110011011111000000011000001000001001000010001111101011100011100100  
11010110111000100010010110010011000000000100011111110000111001100111111  
111010010110110010100000010101000100000101010001000010011100100110110111  
000010011010110110110100001000011110111000110001000011101000101100000000  
011001001111111101001111100100001110001010000010110101111010100010000111  
111010011010101110100001001010100110010100000001100011100001110011001101  
011010000000010100010100010000101111011011101101000110100101110010010110  
111011010010001110110001111100100010011111110000000110000011000010111001  
000100111001000100000011011101011110101101100110100110111011110000111110  
01000110000010000101010101110011100111101011111010010110100011011101111  
111001000110001000100111001100111010111100011001101001101101101101101001  
001101001111011111100000101001001110000100110101110011010001111110110000  
011110101010101000011101011111100110000000001010101011010110011100100111  
001001101100010101110101110100000010010001100100111000111011100001111000  
100111111001110000001000010000111001110001010001010000111001110110111000  
000100110101100111001110011110101011101001101101000001010101000011100010  
010010100000011011011111010101101100101100001010011110000110011000111111  
010000100100111000111101100110101010010100000111000010100101001110101010  
000011011100110000010110100010011000101000110001111110100101111010000100  
100010001110010110101000111011111111000011011100100111111110101010111100  
000010110000111011100001010011010101000000010011001101101101010001101100  
010111010001111001000000100010000110101110111111100100100011010001100111  
110110101110011000001110110111011110100010110110011111000110111100100111  
011011100000001000011000010000011001001111010001111001001011011111111001  
011010010101111100010000101010100010000001010000101101110111111111101101  
11001100000000001011100000111100111111111111110101110111010010001111000110  
010101010001101110001011011111001010011001101110011001111000110011001011  
0010001000010110110111111000000001000110111100001100010011111111000001000  
1001011100101000000000001010100000010111101100101111011010001010110011100  
100010010000011111100100010001111011001001000011110100001000001011001010  
111101100011011110010100011100100111110000110010001110110000001101101011  
111110101111101011111110011001011011101100111011100110111100001000001010  
101000110100000111101111001011000100011111001000111001000000110011101101  
010100111010100110001001101001010000010111111000111000010100010101010111  
0100101100000101001100100111101111101010100011010010110010110110010100110  
010100000000010111001110101010111101100110101001001000010101100010010111  
010110110100110100000011010100000111100100000100001001011001101100011100  
01001101100101111101100000010111111111100010010110011001001111110101111  
010000110010010011000111010011001100111111110010101111110110000011000100  
001000100110100011110010101101010111000001001010001110000111001000011110

111000111010110011110100001010001010101100010010001111011110010001011001  
001011001101100001100000010001110001011001101001100110000111101101000101  
001100010010111111100010010100000110110100111100011000111001011101010010  
101110001010110011111111101000110111000011101010010110001101110011110111  
000001001001001101010011111101001000111100010000010110111110111001000101  
000001100011110100101010100111001001000101111000100101100011001011101111  
00111010010101000000100000010001010110101101110100101100000011111010111  
000100101110001110110000101011001000010101100110000001101111001100111100  
000100110011111110011011100100011011110011011010110010011101010001101100  
111000101111010100000111001000100101101111000110001001101101000100110001  
000001101001110111010111011010111011101001001100110000110010011111101010  
101011011010

#### Расшифрованный текст:

вот пример статьи на тысячу символов. это достаточно маленький текст, оптимально подходящий для карточек товаров в интернет или магазина или для небольших информационных публикаций. в таком тексте редко бывает более двух или трёх абзацев и обычно один или два заголовка. но можно и без него. на тысячу символов рекомендовано использовать один или два ключа и одну картинку. 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, можно и двести символов, то количество слов неизменно возрастает. в копирайтерской деятельности принято считать тысячу пробелами или без. учёт пробелов увеличивает объём 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 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 нужно наценить тысячу знаков без пробелов.

#### Проверка:



### Ключ

### Исходный текст

Зашифровать

Расшифровать

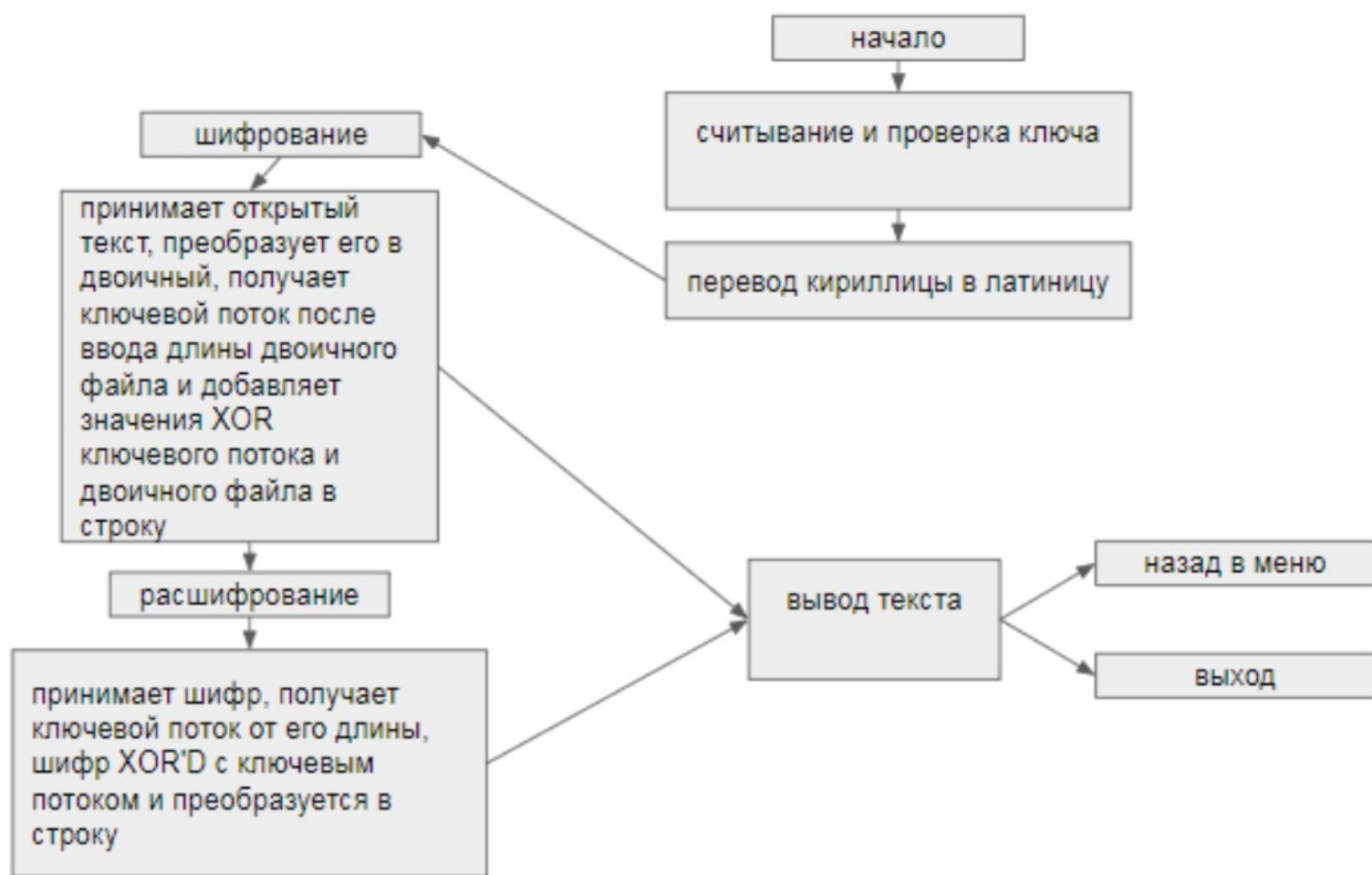
ОЧИСТИТЬ

## 16.A5 /2

A5 — это поточный алгоритм шифрования, используемый для обеспечения конфиденциальности передаваемых данных между телефоном и базовой станцией в европейской системе мобильной цифровой связи GSM (Groupe Spécial Mobile).

Шифр основан на побитовом сложении по модулю два (булева операция «исключающее или») генерируемой псевдослучайной последовательности и шифруемой информации. В A5 псевдослучайная последовательность реализуется на основе трёх линейных регистров сдвига с обратной связью. Регистры имеют длины 19, 22 и 23 бита соответственно. Сдвигами управляет специальная схема, организующая на каждом шаге смещение как минимум двух регистров, что приводит к их неравномерному движению. Последовательность формируется путём операции «исключающее или» над выходными битами регистров.

**Блок-схема:**



**Код программы:**

```
# -*- coding:utf-8 -*-
# импорт компонентов, необходимых для работы программы
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted
import sys
import copy
import re
```

```

reg_x_length = 19
reg_y_length = 22
reg_z_length = 23
reg_e_length = 17

key_one = ""
reg_x = []
reg_y = []
reg_z = []
reg_e = []

# функция загрузки регистров
def loading_registers(key):
    i = 0
    while(i < reg_x_length):
        reg_x.insert(i, int(key[i]))
        i = i + 1
    j = 0
    p = reg_x_length
    while(j < reg_y_length):
        reg_y.insert(j, int(key[p]))
        p = p + 1
        j = j + 1
    k = reg_y_length + reg_x_length
    r = 0
    while(r < reg_z_length):
        reg_z.insert(r, int(key[k]))
        k = k + 1
        r = r + 1
    i = 0
    while(i < reg_e_length):
        reg_e.insert(i, int(key[i]))
        i = i + 1

# функция установки ключа
def set_key(key):
    if(len(key) == 64 and re.match("^[01]+$", key)):
        key_one = key
        loading_registers(key)
        return True
    return False

def get_key():
    return key_one

# функция перевода текста в бинарный код
def to_binary(plain):
    s = ""
    i = 0
    for i in plain:
        binary = str(' '.join(format(ord(x), 'b') for x in i))
        j = len(binary)
        while(j < 12):

```



```

        binary = "0" + binary
        s = s + binary
        j = j + 1
binary_values = []
k = 0
while(k < len(s)):
    binary_values.insert(k, int(s[k]))
    k = k + 1
return binary_values

def get_majority(x, y, z):
    if(x + y + z > 1):
        return 1
    else:
        return 0

# функция получения потока
def get_keystream(length):
    reg_x_temp = copy.deepcopy(reg_x)
    reg_y_temp = copy.deepcopy(reg_y)
    reg_z_temp = copy.deepcopy(reg_z)
    reg_e_temp = copy.deepcopy(reg_e)
    keystream = []
    i = 0
    while i < length:
        majority = get_majority(reg_e_temp[3], reg_e_temp[7],
reg_e_temp[10])
        if get_majority(reg_x_temp[12], reg_x_temp[14], reg_x_temp[15])
== majority:
            new = reg_x_temp[13] ^ reg_x_temp[16] ^ reg_x_temp[17] ^
reg_x_temp[18]
            reg_x_temp_two = copy.deepcopy(reg_x_temp)
            j = 1
            while(j < len(reg_x_temp)):
                reg_x_temp[j] = reg_x_temp_two[j-1]
                j = j + 1
            reg_x_temp[0] = new

        if get_majority(reg_y_temp[9], reg_y_temp[13], reg_y_temp[16]) ==
majority:
            new_one = reg_y_temp[20] ^ reg_y_temp[21]
            reg_y_temp_two = copy.deepcopy(reg_y_temp)
            k = 1
            while(k < len(reg_y_temp)):
                reg_y_temp[k] = reg_y_temp_two[k-1]
                k = k + 1
            reg_y_temp[0] = new_one

        if get_majority(reg_z_temp[13], reg_z_temp[16], reg_z_temp[18])
== majority:
            new_two = reg_z_temp[7] ^ reg_z_temp[20] ^ reg_z_temp[21] ^
reg_z_temp[22]
            reg_z_temp_two = copy.deepcopy(reg_z_temp)

```

```

        m = 1
        while(m < len(reg_z_temp)):
            reg_z_temp[m] = reg_z_temp_two[m-1]
            m = m + 1
        reg_z_temp[0] = new_two

        keystream.insert(i, reg_x_temp[18] ^ reg_y_temp[21] ^
reg_z_temp[22])
        i = i + 1
    return keystream

# функция перевода бинарного кода в текст
def convert_binary_to_str(binary):
    s = ""
    length = len(binary) - 12
    i = 0
    while(i <= length):
        s = s + chr(int(binary[i:i+12], 2))
        i = i + 12
    return str(s)

# функция шифрования
def encrypt(plain):
    s = ""
    binary = to_binary(plain)
    keystream = get_keystream(len(binary))
    i = 0
    while(i < len(binary)):
        s = s + str(binary[i] ^ keystream[i])
        i = i + 1
    return s

# функция расшифрования
def decrypt(cipher):
    s = ""
    binary = []
    keystream = get_keystream(len(cipher))
    i = 0
    while(i < len(cipher)):
        binary.insert(i, int(cipher[i]))
        s = s + str(binary[i] ^ keystream[i])
        i = i + 1
    return convert_binary_to_str(str(s))

# функция проверки введенного ключа
def user_input_key():
    tha_key = str(input('Введите 64-bit ключ: '))
    if (len(tha_key) == 64 and re.match("^[01]+$", tha_key)):
        return tha_key
    else:
        while(len(tha_key) != 64 and not re.match("^[01]+$", tha_key)):
            if (len(tha_key) == 64 and re.match("^[01]+$", tha_key)):
                return tha_key

```

```

        tha_key = str(input('Введите 64-bit ключ: '))
    return tha_key

# установка ключа
key = '0101001000011010110001110001100100101001000000110111111010110111'
set_key(key)

# вывод результатов работы программы
print(f'''
A5/2:
Ключ: {key}
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{encrypt(input_for_cipher_short())}

Расшифрованный текст:
{output_from_decrypted(decrypt(encrypt(
    input_for_cipher_short())))}

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
{encrypt(input_for_cipher_long())}

Расшифрованный текст:
{output_from_decrypted(decrypt(encrypt(
    input_for_cipher_long())))}
''')

```

Тестирование:

```

/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab06_16_a52.py

A5/2:
Ключ: 0101001000011010110001110001100100101001000000110111111010110111
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
010000110010010001000000010000110101010000111100010001001111010000110111
010000111111010001000010010000111111010001000000010000111000010000111011
010000111000010000110010010001001011010000111000010000111110010001000010
010000111011010000111000010000110010010001001011010000111101010000110101
010000110110010000110100010001000011010001000010010001000111010000110101
010000111011010000111110010000110010010000110101010000111010010000110000
010001000010010001000111010000111010

Расшифрованный текст:
время, приливы и отливы не ждут человека.

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
010000110010010000111110010001000010010000111111010001000000010000111000
0100001111100010000110101010001000000010001000001010001000010010000110000
010001000010010001001100010000111000010000111101010000110000010001000010
010001001011010001000001010001001111010001000111010001000011010001000001

```

010000111000010000111100010000110010010000111110010000111011010000111110  
010000110010010001000010010001000111010000111010010001001101010001000010  
010000111110010000110100010000111110010001000001010001000010010000110000  
010001000010010000111110010001000111010000111101010000111110010000111100  
010000110000010000111011010000110101010000111101010001001100010000111010  
010000111000010000111001010001000010010000110101010000111010010001000001  
010001000010010000110111010000111111010001000010010000111110010000111111  
010001000010010000111000010000111100010000110000010000111011010001001100  
010000111101010000111110010000111111010000111110010000110100010001000101  
010000111110010000110100010001001111010001001001010000111000010000111001  
010000110100010000111011010001001111010000111010010000110000010001000000  
010001000010010000111110010001000111010000110101010000111010010001000010  
010000111110010000110010010000110000010001000000010000111110010000110010  
0100001100100100001110000100001111010100001000010010000110101010001000000  
010000111101010000110101010001000010010000111000010000111011010000111000  
010000111100010000110000010000110011010000110000010000110111010000111000  
010000111101010000110000010001000101010000111000010000111011010000111000  
010000110100010000111011010001001111010000111101010000110101010000110001  
010000111110010000111011010001001100010001001000010000111000010001000101  
010000111000010000111101010001000100010000111110010001000000010000111100  
010000110000010001000110010000111000010000111110010000111101010000111101  
010001001011010001000101010000111111010001000011010000110001010000111011  
010000111000010000111010010000110000010001000110010000111000010000111001  
010001000010010001000111010000111010010000110010010001000010010000110000  
010000111010010000111110010000111100010001000010010000110101010000111010  
010001000001010001000010010000110101010001000000010000110101010000110100  
010000111010010000111110010000110001010001001011010000110010010000110000  
010000110101010001000010010000110001010000111110010000111011010000110101  
010000110101010000110100010000110010010001000011010001000101010000111000  
010000111011010000111000010001000010010001000000010001010001010001000101  
010000110000010000110001010000110111010000110000010001000110010000110101  
010000110010010000111000010000111110010000110001010001001011010001000111  
010000111101010000111110010000111110010000110100010000111000010000111101  
010000111111010000111110010000110100010000110111010000110000010000110011  
010000111110010000111011010000111110010000110010010000111110010000111010  
010001000010010001000111010000111010010000111101010000111110010000111100  
010000111110010000110110010000111101010000111110010000111000010000110001  
010000110101010000110111010000111101010000110101010000110011010000111110  
0100010000100100010001110100001110100100001111010100001110000010001000010  
010001001011010001000001010001001111010001000111010001000011010001000001  
010000111000010000111100010000110010010000111110010000111011010000111110  
010000110010010001000000010000110101010000111010010000111110010000111100  
010000110101010000111101010000110100010000111110010000110010010000110000  
010000111101010000111110010000111000010001000001010000111111010000111110  
010000111011010001001100010000110111010000111110010000110010010000110000  
010001000010010001001100010000111110010000110100010000111000010000111101  
010000111000010000111011010000111000010000110100010000110010010000110000  
010000111010010000111011010001001110010001000111010000110000010000111000  
010000111110010000110100010000111101010001000011010000111010010000110000  
010001000000010001000010010000111000010000111101010001000011010001000010  
010001000111010000111010010001000010010000110101010000111010010001000001  
010001000010010000111101010000110000010001000010010001001011010001000001  
010001001111010001000111010001000011010001000001010000111000010000111100

010000110010010000111110010000111011010000111110010000110010010001001101  
010001000010010000111110010001000001010000111010010000111110010000111011  
010001001100010000111010010000111110010000111111010001000000010000111000  
010000111100010000110101010001000000010000111101010000111110010001000001  
010000111011010000111110010000110010010001000010010001000111010000111010  
010001000001010001000010010000110000010001000010010000111000010001000001  
010001000010010000111000010000111010010000110000010000111111010000111110  
010000111010010000110000010000110111010001001011010000110010010000110000  
010000110101010001000010010000110111010000111111010001000010010001000111  
010001000010010000111110010001000010010001001011010001000001010001001111  
010001000111010000110000010000110010010000111010010000111011010001001110  
010001000111010000110000010000110101010001000010010000110010010001000001  
010000110101010000110001010001001111010001000001010001000010010000111110  
010000111111010001001111010001000010010001001100010000110100010000110101  
010001000001010001001111010001000010010000111000010000111011010000111000  
010000110100010000110010010000110101010001000001010001000010010000111000  
010001000001010000111011010000111110010000110010010001000001010001000000  
010000110101010000110100010000111101010000110101010000111001010000110010  
010000110101010000111011010000111000010001000111010000111000010000111101  
010001001011010001000010010001000111010000111010010000111101010000111110  
010000110111010000111111010001000010010000110101010001000001010000111011  
010000111000010000110111010000111011010000111110010001000011010000111111  
010000111110010001000010010001000000010000110101010000110001010000111011  
010001001111010001000010010001001100010000111111010001000000010000110101  
010000110100010000111011010000111110010000110011010000110000010000111100  
010000111000010000110111010000111111010001000010010001000001010000111110  
010001001110010000110111010000110000010000111100010000111000010000111000  
010001001110010000110111010000110000010000111100010000111000010000111000  
010000110100010001000000010001000011010000110011010000111000010000111100  
010000111000010001000111010000110000010001000001010001000010010001001111  
010000111100010000111000010001000000010000110101010001000111010000111000  
010000111101010000110000010000111110010000110100010000111000010000111101  
010000111000010000111011010000111000010000110100010000110010010000110000  
010001000001010000111000010000111100010000110010010000111110010000111011  
010000110000010000110111010000111111010001000010010001000010010000111110  
010000111010010000111110010000111011010000111000010001000111010000110101  
010001000001010001000010010000110010010000111110010001000001010000111011  
010000111110010000110010010000111101010000110101010000111000010000110111  
010000111100010000110101010000111101010000111101010000111110010000110010  
010000111110010000110111010001000000010000110000010001000001010001000010  
010000110000010000110101010001000010010001000010010001000111010000111010  
010000110010010000111010010000111110010000111111010000111000010001000000  
010000110000010000111001010001000010010000110101010001000000010001000001  
010000111010010000111110010000111001010000110100010000110101010001001111  
010001000010010000110101010000111011010001001100010000111101010000111110  
010001000001010001000010010000111000010000111111010001000000010000111000  
010000111101010001001111010001000010010000111110010001000001010001000111  
010000111000010001000010010000110000010001000010010001001100010001000010  
010001001011010001000001010001001111010001000111010000111000010001000001  
010000111111010001000000010000111110010000110001010000110101010000111011  
010000110000010000111100010000111000010000111000010000111011010000111000  
010000110001010000110101010000110111010001000010010001000111010000111010  
010001000011010001000111010000110101010001000010010000111111010001000000  
010000111110010000110001010000110001010000111110010000111110010000110010

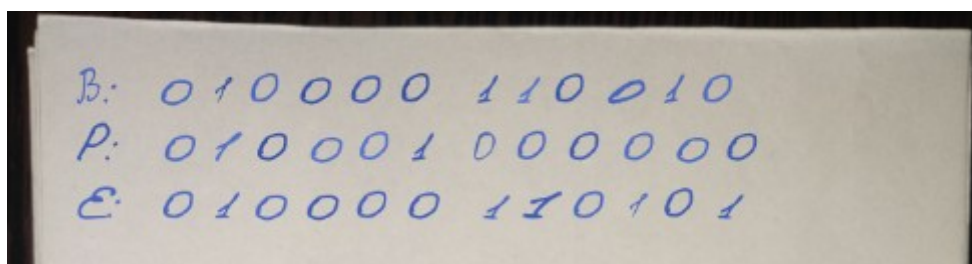
010001000011010000110010010000110101010000111011010000111000010001000111  
010000111000010000110010010000110000010000110101010001000010010000111110  
010000110001010001001010010000110101010000111100010001000010010000110101  
010000111010010001000001010001000010010000110000010000111111010001000000  
010000111000010000111100010000110101010001000000010000111101010000111110  
010000111101010000110000010001000001010001000010010000111110010000111000  
010000111011010000111000010000110100010000110010010000110101010001000001  
010001000010010000111000010001000001010000111000010000111100010000110010  
010000111110010000111011010000111110010000110010010000111000010000111100  
010000110101010000111101010000111101010000111110010001000001010001000010  
010000111110010000111011010001001100010000111010010000111110010001000000  
010000110000010000110111010000111100010001001011010001000000010000110000  
010000110111010000110100010000110101010000111011010001001111010000110101  
0100001111100010001000001010000111011010000111110010000110010010000110000  
010001000001010000110010010000111110010000110001010000111110010000110100  
010000111101010001001011010000111100010000111111010001000000010000111110  
010001000001010001000010010001000000010000110000010000111101010001000001  
010001000010010000110010010000111110010000111100010001000010010001000111  
010000111010010001000001010001000111010000111000010001000010010000110000  
010001000010010001001100010000111111010001000000010000111110010000110001  
010000110101010000111011010001001011010000110111010000110000010000111010  
010000110000010000110111010001000111010000111000010000111010010000111000  
010000111101010000110101010000111011010001001110010000110001010001001111  
010001000010010000110111010000111111010001000010010001000010010000110000  
010000111010010000111010010000110000010000111010010001001101010001000010  
010000111110010000111111010001000011010001000001010001000010010000111110  
010000110101010000111100010000110101010001000001010001000010010000111110  
010001000010010001000111010000111010010000111110010000110100010000111101  
010000110000010000111010010000111110010000111101010000110101010000111010  
010000111110010001000010010000111110010001000000010001001011010000110101  
010001000100010000111000010001000000010000111100010001001011010000111000  
010000110001010000111000010001000000010000110110010000111000010000110010  
0100001110000100001101000100010011111010001000010010001000001010000111111  
010001000000010000110000010000110010010000110101010000110100010000111011  
010000111000010000110010010001001011010000111100010001000001010001000010  
010000110000010000110010010000111000010001000010010001001100010001000001  
010001000010010000111110010000111000010000111100010000111110010001000001  
010001000010010001001100010000110111010000110000010001000010010001001011  
010001000001010001001111010001000111010001000011010001000001010000111000  
010000111100010000110010010000111110010000111011010000111110010000110010  
010001000001010000111111010001000000010000111110010000110001010000110101  
010000111011010000110000010000111100010000111000010000110111010000111111  
010001000010010001000001010001000111010000111000010001000010010000110000  
010001001111010000111111010000111110010001000001010000111011010000110101  
010000110100010000111101010000111000010000110101010000110010010000110000  
0100001101100100001111010100001001011010000111100010001001101010000111011  
010000110101010000111100010000110101010000111101010001000010010000111110  
010000111100010000111010010000110000010001000111010000110101010001000001  
010001000010010000110010010000110101010000111101010000111101010000111110  
010000110011010000111110010000110010010000111110010001000001010000111111  
0100010000000100001110000100010011111010001000010010000111000010001001111  
010001000010010001000111010000111010010001000001010000111110010000110011  
010000111011010000110000010001000001010000111000010001000010010000110101

010001000001010001001100010000110111010000111111010001000010010001000111  
010000111000010001000010010000110000010001000010010001001100010001000001  
010000111011010000111000010001000010010000111101010001001011010000111001  
010001000010010000110101010000111010010001000001010001000010010000110001  
010000110101010000110111010000110101010000110100010000111000010000111101  
010000111110010000110011010000111110010000111111010001000000010000111110  
010000111111010001000011010001000001010000111010010000110000010000110111  
010000111111010001000010010000111101010000111000010000111010010001000010  
010000111110010000111101010000110101010000110001010001000011010000110100  
010000110101010001000010010001000010010001000111010000111010010000111101  
010000111110010000110001010000111110010000111011010001001100010001001000  
010000111000010000111101010001000001010001000010010000110010010001000011  
010000111101010001000011010000110110010000111101010000110000010001000110  
010000110101010000111101010000110000010000110111010000110000010001000010  
010001001011010001000001010001001111010001000111010001000011010000110111  
010000111101010000110000010000111010010000111110010000110010010000110001  
010000110101010000110111010000111111010001000000010000111110010000110001  
010000110101010000111011010000111110010000110010010001000010010001000111  
010000111010

#### Расшифрованный текст:

вот пример статьи на тысячу символов. это достаточно маленький текст, оптимально подходящий для карточек товаров в интернет или магазина или для небольших информационных публикаций. в таком тексте редко бывает более двух или трёх абзацев и обычно один подзаголовок. можно и без него. на тысячу символов рекомендовано использовать один или два ключа и одну картинку. текст на тысячу символов это сколько примерно слов. статистика показывает, что тысяча включает в себя столько десяти или двести слов средней величины. но, если злоупотреблять предлогами, союзами и другими частями речи, можно и два символа, то количество слов неизменно возрастает. в копирайтерской деятельности принято считать тысячу пробелами или без. учёт пробелов увеличивает объём текста примерно на столько же, сколько символов, именно столько раз мы разделяем слова свободным пространством. считать пробелы заказчики не любят, так как это пустое место. однако некоторые фирмы биржи видят справедливым ставить стоимость за тысячу символов в пробелах, считая последние важным элементом качественного восприятия. согласитесь, читать слитный текст без единого пропуска, никто не будет. но большинству нужно наценить тысячу знаков без пробелов.

#### Проверка:





### Ключ

01010010000110101100011100011001001001000000110111111010110111

Расшифрованный текст

[illegible]

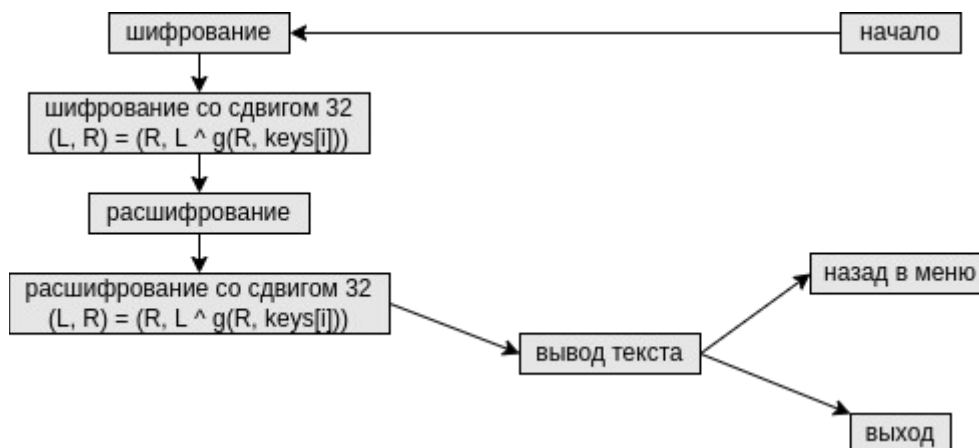
ОЧИСТИТЬ

## Блок G: КОМБИНАЦИОННЫЕ ШИФРЫ

### 17.МАГМА

Магма представляет собой симметричный блочный алгоритм шифрования с размером блока входных данных 64 бита, секретным ключом 256 бит и 32 раундами шифрования.

Блок-схема:



Код программы:

```
# импорт компонентов, необходимых для работы программы
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted

pi0 = [12, 4, 6, 2, 10, 5, 11, 9, 14, 8, 13, 7, 0, 3, 15, 1]
pi1 = [6, 8, 2, 3, 9, 10, 5, 12, 1, 14, 4, 7, 11, 13, 0, 15]
pi2 = [11, 3, 5, 8, 2, 15, 10, 13, 14, 1, 7, 4, 12, 9, 6, 0]
pi3 = [12, 8, 2, 1, 13, 4, 15, 6, 7, 0, 10, 5, 3, 14, 9, 11]
pi4 = [7, 15, 5, 10, 8, 1, 6, 13, 0, 9, 3, 14, 11, 4, 2, 12]
pi5 = [5, 13, 15, 6, 9, 2, 12, 10, 11, 7, 8, 1, 4, 3, 14, 0]
pi6 = [8, 14, 2, 5, 6, 9, 1, 12, 15, 4, 11, 0, 13, 10, 3, 7]
pi7 = [1, 7, 14, 13, 0, 5, 8, 3, 4, 15, 10, 6, 9, 12, 11, 2]

pi = [pi0, pi1, pi2, pi3, pi4, pi5, pi6, pi7]

MASK32 = 2 ** 32 - 1

def t(x):
    y = 0
    for i in reversed(range(8)):
        j = (x >> 4 * i) & 0xf
        y <=< 4
        y ^= pi[i][j]
    return y

# функция сдвига на 11
def rot11(x):
    return ((x << 11) ^ (x >> (32 - 11))) & MASK32
```

```

def g(x, k):
    return rot11(t((x + k) % 2 ** 32))

def split(x):
    L = x >> 32
    R = x & MASK32
    return (L, R)

def join(L, R):
    return (L << 32) ^ R

def magma_key_schedule(k):
    keys = []
    for i in reversed(range(8)):
        keys.append((k >> (32 * i)) & MASK32)
    for i in range(8):
        keys.append(keys[i])
    for i in range(8):
        keys.append(keys[i])
    for i in reversed(range(8)):
        keys.append(keys[i])
    return keys

# функция шифрования
def magma_encrypt(x, k):
    keys = magma_key_schedule(k)
    (L, R) = split(x)
    for i in range(31):
        (L, R) = (R, L ^ g(R, keys[i]))
    return join(L ^ g(R, keys[-1]), R)

# функция расшифрования
def magma_decrypt(x, k):
    keys = magma_key_schedule(k)
    keys.reverse()
    (L, R) = split(x)
    for i in range(31):
        (L, R) = (R, L ^ g(R, keys[i]))
    return join(L ^ g(R, keys[-1]), R)

# установка ключа
key =
int('ffeeddcbbbaa99887766554433221100f0f1f2f3f4f5f6f7f8f9fafbfcfdfeff',
16)

i = 0
text_short = input_for_cipher_short()
encr_short = []
while (i < len(text_short)):
    text = text_short[i:i+4].encode().hex()
    text = int(text, 16)
    text = text % 2**64
    pt = text

```

```

    ct = magma_encrypt(pt, key)
    encr_short.append(ct)
    i += 4
decr_short = []
for i in encr_short:
    dt = magma_decrypt(i, key)
    decr_short.append(bytes.fromhex(hex(dt)[2:]).decode('utf-8'))

i = 0
text_long = input_for_cipher_long()
encr_long = []
while (i < len(text_long)):
    text = text_long[i:i+4].encode().hex()
    text = int(text, 16)
    text = text % 2**64
    pt = text
    ct = magma_encrypt(pt, key)
    encr_long.append(ct)
    i += 4
decr_long = []
for i in encr_long:
    dt = magma_decrypt(i, key)
    decr_long.append(bytes.fromhex(hex(dt)[2:]).decode('utf-8'))

#вывод результатов работы программы
print(f'''
МАГМА:
КЛЮЧ:
{key}

КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{encr_short}

Расшифрованный текст:
{output_from_decrypted(''.join(decr_short))}

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
{encr_long}

Расшифрованный текст:
{output_from_decrypted(''.join(decr_long))}
''')

```

## Тестирование:

```

/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab07_17_magma.py

МАГМА:
КЛЮЧ:

```

115761816795685524522806652725025505786200410505847444308688553892001406  
123775

КОРОТКИЙ ТЕКСТ:

Зашифрованный текст:

[18432907413224455314, 10996816857283808610, 1603220777717569738,  
6798339374272425273, 625275379878570582, 12897841916972840738,  
12693135464871535956, 6338906346771095526, 2952080121925535959,  
10585345143535530769]

Расшифрованный текст:

время, приливы и отливы не ждут человека.

ДЛИННЫЙ ТЕКСТ:

Зашифрованный текст:

[8343875602038808058, 8041676007725686027, 7453366501928099122,  
13622418350652450349, 13257185717927971463, 3967957328028735447,  
6303045106133971510, 17694607534926283087, 3147373830694159915,  
9390122889656481141, 15485378634152172683, 11157654134498461325,  
9103621146938759596, 18167786785284461467, 4720588269896140616,  
13198230271257633374, 290271297514756748, 6383750138269828222,  
6966411025704842352, 1184081237020962173, 5518298843322725716,  
1121075660661397656, 10048741617664022091, 18363685243888377546,  
95605295917991907, 17057631210449679753, 16551488750349984268,  
2025585132759738150, 12949061887509852732, 9329546944062831470,  
3418244764798143790, 15295462055836379806, 5155680359283034501,  
9451937380501232912, 7328220556957620599, 3614937074236676557,  
4589181499513045879, 13803166415433109120, 18278897981036660435,  
4019936986318215394, 12394731635219602309, 5416248858033923816,  
463232828950025933, 15989759834534178049, 13360939273674915028,  
8528608555291906000, 10293453731533650333, 13238975415012678998,  
1261680965272169368, 10483829524585249195, 15703560336104773581,  
2928612788789974786, 12370719347969566168, 15498018845651124401,  
7975202005615435922, 1001640210394968931, 9482390876876309460,  
10306886503249707528, 3432371120083104576, 13284723522159984134,  
4732957936308484548, 18145150074698797421, 7665912658016440955,  
17186319045640305491, 13257185717927971463, 3967957328028735447,  
6303045106133971510, 1163720297091009342, 2346699712756694809,  
10356924971308547229, 13990735642799502546, 16656837303721777190,  
3620970884711728977, 9443487292630364151, 7691049094682026909,  
916112073177174944, 15797466941996902229, 16777509928585489639,  
6303121470213798711, 11674884899395813190, 18046574429007017128,  
13077766683806041775, 6569214857954439428, 13257185717927971463,  
3967957328028735447, 6303045106133971510, 18131068278186858513,  
16002418093945299027, 2908114868778074520, 11794392602089101467,  
1199691821864277191, 3076938350476950006, 13450817471486594379,  
10728741519359495825, 9269189682480091597, 14111778268298096419,  
2057896382844501704, 10293453731533650333, 17204212421686848739,  
5366206226868800702, 8248466997448536722, 10088810565957275850,  
2254355090243742257, 8371896884399298133, 10985518332543285851,  
17941711041904849701, 7740200207943279354, 10750419551538638209,  
2629297776519959089, 2225742027093608502, 287244974227193751,  
17712479779128577940, 9222296997056994615, 973788500067181560,

3393954245381334839, 10641483480207737426, 16269104980183698483,  
16250235473790191753, 2700688729269881630, 8411607884348754050,  
7636448048795956798, 692746442652315515, 11343070527907873916,  
629521920599350430, 14623128418242633989, 5705247633908105959,  
17910568456840661856, 16201319981229519019, 13769298882463179721,  
7421154077805485456, 9312183877157555560, 7691049094682026909,  
916112073177174944, 15135398776342705892, 15289260373132081671,  
7126379060141568292, 1725036945967272673, 17688755856354297123,  
138113508643393643, 14127475102588350875, 4151016356341159175,  
9841467487393741636, 3187959327495924697, 15728081324422447459,  
16055605958376054530, 3572266014017346359, 3223827000587680474,  
8253733500166311251, 9140508542287809951, 1790032636239692933,  
14268543881935307321, 7342482303792008383, 17677570963908781400,  
13355264069952555938, 88597409000583480, 17257030184527390406,  
12715322944275189718, 4311305660788670662, 7008462056628757749,  
5705247633908105959, 780471545777017295, 7181337429584918154,  
17602858510462705198, 14738061815484779755, 540299310016161264,  
10182683375894303875, 8026343204551337982, 17660378940581534682,  
13825198160625090606, 4720588269896140616, 5730527409401222056,  
17941293450735671408, 4342366351984147086, 5313276448158165379,  
1745024168316228346, 2528136519964988665, 3547755811857389063,  
10306886503249707528, 4526494865159228242, 7342482303792008383,  
2908114868778074520, 7471680038596777270, 16412799349039310306,  
17456146687817481297, 2894321475946271648, 6923158161737971479,  
10027397076122517884, 11785376343832896276, 4548889844418068535,  
9068186136604959843, 17006372751687582512, 9582236842043709584,  
2756346197016409968, 12319862187853716832, 14738061815484779755,  
3597986015127692257, 6006230909502237940, 8049089039016717789,  
6157047194034613233, 6958374052664244197, 16431478885293681293,  
8935011217054641103, 4493811008412807767, 2950821783567146896,  
1855179961465792476, 6714524409280649005, 13570818025122198173,  
14348383310813638904, 4453067278648320496, 3438244984932255310,  
9108046240042143694, 151270379504886441, 10183946694284410805,  
580256645846205130, 5356452764985505955, 1949584005305844760,  
5054509616596506897, 806561947066774048, 14232614625982377749,  
4486402459089481420, 6142067374720541917, 11774712973292423210,  
10305629430321463988, 3547755811857389063, 10306886503249707528,  
15489555906471766183, 15441006425873291865, 2659805083408552087,  
12582666581260422456, 541882649396588128, 4886609086442669621,  
5561196856767141965, 7731325888395147745, 1096626906536588642,  
763947604362793674, 12798794780496052712, 8299242569789435168,  
6766061707386831762, 8971528550958227067, 11254181301937016096,  
4106793478691170180, 5812305374465111022, 14122431208396972645,  
8756880839685289505, 1840145209499476946, 17204212421686848739,  
12702832949266190125, 12312436585117905032, 2731161190011158263,  
4720588269896140616, 13119149454917978330, 9822108851446653348,  
4085677428333094007, 12601379616160262740, 10265087333462641478,  
8770956180316534333, 2746212430076756819, 10780803050552638165,  
1906124277192103447, 8318790287988703211, 9451937380501232912,  
7154010589201716226, 6593238725915939856, 16381960257093506421,  
16805496448556233763, 13257185717927971463, 3534238871897283987,  
18436104050552264826, 3237166291520387028, 14738061815484779755,  
540299310016161264, 10585345143535530769]

Расшифрованный текст:

вот пример статьи на тысячу символов. это достаточно маленький текст, оптимально подходящий для карточек товаров в интернет или магазина или для небольших информационных публикаций. в таком тексте редко бывает более двух или трёх абзацев и обычно один подзаголовок. можно и без него. тысячу символов рекомендовано использовать или два ключа и одну картинку. 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, можно и два символа, то количество слов неизменно возрастает. в копирайтерской деятельности принято считать тысячу пробелами или без. учёт пробелов увеличивает объём 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. одна из некоторых фирм биржи видит справедливым поставить стоимость за тысячу символов в пробелах, считая последние важным элементом качественного восприятия. согласитесь, читать слитный текст без единого пропуска, никто не будет. но большинство нужно наценить за тысячу знаков без пробелов.

Проверка (код вставлен в интерпретатор и отдельно вызвана функция шифрования, контрольные значения взяты из ГОСТ):

```
Python 3.10.1 (main, Dec 18 2021, 23:53:45) [GCC 11.1.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> pi0 = [12, 4, 6, 2, 10, 5, 11, 9, 14, 8, 13, 7, 0, 3, 15, 1]
>>> pi1 = [6, 8, 2, 3, 9, 10, 5, 12, 1, 14, 4, 7, 11, 13, 0, 15]
>>> pi2 = [11, 3, 5, 8, 2, 15, 10, 13, 14, 1, 7, 4, 12, 9, 6, 0]
>>> pi3 = [12, 8, 2, 1, 13, 4, 15, 6, 7, 0, 10, 5, 3, 14, 9, 11]
>>> pi4 = [7, 15, 5, 10, 8, 1, 6, 13, 0, 9, 3, 14, 11, 4, 2, 12]
>>> pi5 = [5, 13, 15, 6, 9, 2, 12, 10, 11, 7, 8, 1, 4, 3, 14, 0]
>>> pi6 = [8, 14, 2, 5, 6, 9, 1, 12, 15, 4, 11, 0, 13, 10, 3, 7]
>>> pi7 = [1, 7, 14, 13, 0, 5, 8, 3, 4, 15, 10, 6, 9, 12, 11, 2]
>>> pi = [pi0, pi1, pi2, pi3, pi4, pi5, pi6, pi7]
>>> MASK32 = 2 ** 32 - 1
>>>
>>> def t(x):
...     y = 0
...     for i in reversed(range(8)):
...         j = (x >> 4 * i) & 0xf
...         y <=< 4
...         y ^= pi[i][j]
...     return y
...
>>> # функция сдвига на 11
>>> def rot11(x):
...     return ((x << 11) ^ (x >> (32 - 11))) & MASK32
```



```

...
>>> def g(x, k):
...     return rotl1(t((x + k) % 2 ** 32))
...
>>> def split(x):
...     L = x >> 32
...     R = x & MASK32
...     return (L, R)
...
>>> def join(L, R):
...     return (L << 32) ^ R
...
>>> def magma_key_schedule(k):
...     keys = []
...     for i in reversed(range(8)):
...         keys.append((k >> (32 * i)) & MASK32)
...     for i in range(8):
...         keys.append(keys[i])
...     for i in range(8):
...         keys.append(keys[i])
...     for i in reversed(range(8)):
...         keys.append(keys[i])
...     return keys
...
>>> # функция шифрования
>>> def magma_encrypt(x, k):
...     keys = magma_key_schedule(k)
...     (L, R) = split(x)
...     for i in range(31):
...         (L, R) = (R, L ^ g(R, keys[i]))
...     return join(L ^ g(R, keys[-1]), R)
...
>>> # функция расшифрования
>>> def magma_decrypt(x, k):
...     keys = magma_key_schedule(k)
...     keys.reverse()
...     (L, R) = split(x)
...     for i in range(31):
...         (L, R) = (R, L ^ g(R, keys[i]))

```

```
...     return join(L ^ g(R, keys[-1]), R)
...
>>> # установка ключа
>>> key =
int('ffeaddccbbaa99887766554433221100f0f1f2f3f4f5f6f7f8f9fafbfcfdfeff',
16)
>>>
>>> magma_encrypt(int('fedcba9876543210', 16), key)
5686078090860087869
>>> int('4ee901e5c2d8ca3d', 16)
5686078090860087869
>>> magma_encrypt(int('fedcba9876543210', 16), key) ==
int('4ee901e5c2d8ca3d', 16)
True
```

Интерфейс:

17. МАГМА

Ключ

ffeaddccbbaa99887766554433221100f0f1f2f3f4f5f6f7f8f9fafbfcfdfeff

Исходный текст

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 предложениями, 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

Зашифрованный текст

[8343875602038808058, 8041676007725686027, 7453366501928099122, 13622418350652450349, 13257185717927971463, 3967957328028735447, 6303045106133971510, 17694607534926283087, 3147373830694159915, 9390122889656481141, 15485378634152172683, 11157654134498461325, 9103621146938759596, 18167786785284461467, 4720588269896140616, 13198230271257633374, 290271297514756748, 6383750138269828222, 6966411025704842352, 1184081237020962173, 5518298843322725716, 1121075660661397656, 10048741617664022091, 18363685243888377546, 95605295917991907, 17057631210449679753, 16551488750349984268, 2025585132759738150, 12949061887509852732, 9329546944062831470, 3418244764798143790, 15295462055836379806, 5155680359283034501, 9451937380501232912, 7328220556957620599, 3614937074236676557, 4589181499513045879, 13803166415433109120, 18278897981036660435, 4019936986318215394,

Расшифрованный текст

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 предложениями, 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

Зашифровать

Расшифровать

Очистить

Выполнил: Барышников С.С. 191-351

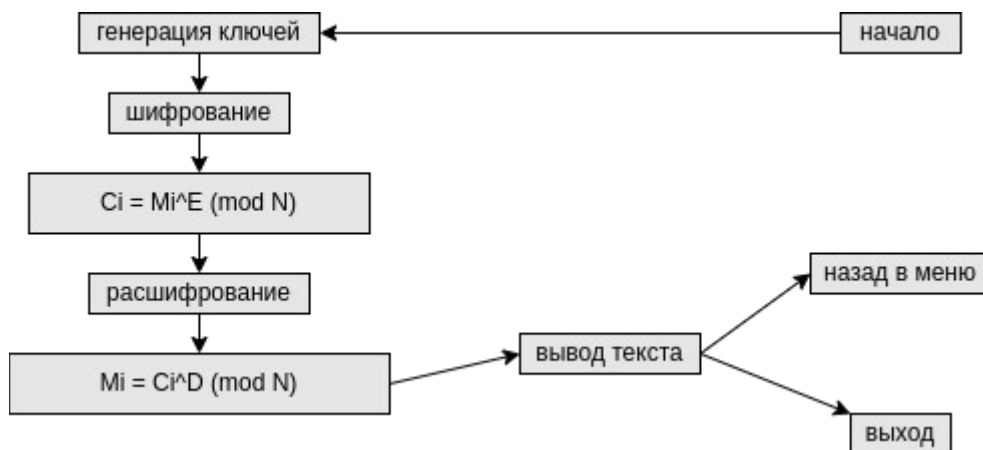
77

## БЛОК Н: АСИММЕТРИЧНЫЕ ШИФРЫ

### 21.RSA

Берутся два очень больших простых числа  $P$  и  $Q$  и находится произведение простых чисел  $N=P \times Q$  и функция Эйлера от этого произведения  $\phi(N)=(P-1) \times (Q-1)$ . Выбирается случайное целое число  $E$ , взаимно простое с  $\phi(N)$ , и вычисляется  $D=(1 \bmod \phi(N))/E$ .  $E$  и  $N$  публикуются как открытый ключ,  $D$  сохраняется в тайне. Шифрование: Если  $M$  — сообщение, то шифртекст  $C_i$  получается последовательным шифрованием каждой шифрвеличины  $M_i$  возведением ее в степень  $E$  по модулю  $N$ :  $C_i = M_i^E \bmod N$ . Расшифрование: Получатель расшифровывает сообщение, возводя последовательно  $C_i$  в степень  $D$  по модулю  $N$ :  $M_i = C_i^D \bmod N$ .

Блок-схема:



Код программы:

```
# -*- coding:utf-8 -*-
# импорт компонентов, необходимых для работы программы
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted
import random

# функция вычисления НОД
def gcd(a, b):
    while b != 0:
        a, b = b, a % b
    return a

# функция получения обратного числа
def multiplicative_inverse(e, r):
    for i in range(r):
        if ((e*i) % r == 1):
            return i

# функция проверки числа на простоту
def is_prime(num):
    if num == 2:
        return True
```

```

if num < 2 or num % 2 == 0:
    return False
for n in range(3, int(num**0.5)+2, 2):
    if num % n == 0:
        return False
return True

# функция генерации пары ключей
def generate_keypair(p, q):
    if not (is_prime(p) and is_prime(q)):
        raise ValueError('Оба числа должны быть простыми.')
    elif p == q:
        raise ValueError('p и q не могут быть равны друг другу')
    n = p * q

    phi = (p-1) * (q-1)

    e = random.randrange(1, phi)

    g = gcd(e, phi)
    while g != 1:
        e = random.randrange(1, phi)
        g = gcd(e, phi)
    d = multiplicative_inverse(e, phi)
    return ((e, n), (d, n))

# функция шифрования
def encrypt(pk, plaintext):
    key, n = pk
    cipher = [(ord(char) ** key) % n for char in plaintext]
    return cipher

# функция расшифрования
def decrypt(pk, ciphertext):
    key, n = pk
    plain = [chr((char ** key) % n) for char in ciphertext]
    return ''.join(plain)

# установка ключа
p = 107
q = 109
public, private = generate_keypair(p, q)

message_short = input_for_cipher_short()
encrypted_short = encrypt(private, message_short)
print_enc_short = ''.join([str(x) for x in encrypted_short])
decrypted_short = decrypt(public, encrypted_short)

message_long = input_for_cipher_long()
encrypted_long = encrypt(private, message_long)
print_enc_long = ''.join([str(x) for x in encrypted_long])
decrypted_long = decrypt(public, encrypted_long)

```

```

#вывод результатов работы программы
print(f'''
RSA:
Ключ:
p={p} q={q}
Публичный: {public}
Приватный: {private}
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{print_enc_short}

Расшифрованный текст:
{output_from_decrypted(decrypted_short)}

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
{print_enc_long}

Расшифрованный текст:
{output_from_decrypted(decrypted_long)}
''')

```

Тестирование:

```

/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab08_21_rsa.py

RSA:
Ключ:
p=31 q=37
Публичный: (107, 1147)
Приватный: (323, 1147)

КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
112421396193172538986842719581719112824719949865817191128245891395265455
39861068139581941121391448139861068144

Расшифрованный текст:
время, приливы и отливы не ждут человека.

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
112949868427196191394293298681398649271958981398682493231710685539327196
191129458194112986106814449869465494932986813986941068589946198135811395
894921447196809861391449329862538986948986719619813581492589948946546059
465431723971968065458131714481342986941068139144986941128134294112112719
589986139425891399867195817196198133898132537195898136057195817196545813
175891396299458149288071960571958981694426198134007199458958982460585536
295817191448134007196809861068144112986813144946199861391449329861394213
965414494629824112813139986629945811391396541125536057195817199864257760
581362925381340013911271994629824106858994946547195898946542538133899458
194112941449861068144589946199452589947196291392535891393899498610681445
898139868249323171068553932719619112945819411242139144946191395896549411

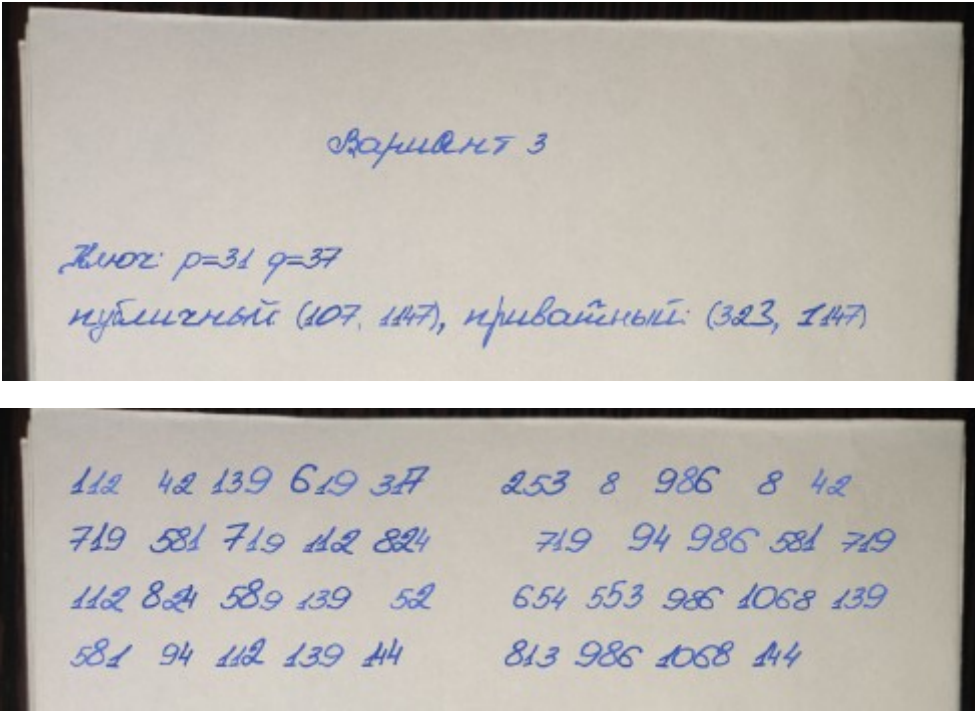
```

281358994719932894581492253941128139864929465471958971958171965411281314  
458172610688137199465458955314481342986719589553986106814498613914493298  
658981398682493231710685539327196191129458194112498694932144945814921449  
484271961913942589949325819411298610681449329868139867199329867191448138  
941448132538241128131399862538986106898694986824932317106881311214458172  
610688131399861129321396293179329869483179864926541399323179867195817196  
541121399329867199325819411293242139654589139680112139581719106871958982  
498610681445899425389861399325817192535819455389498642139629581317986492  
842139654581943898136197192538986932947262538136197197196544255338971961  
971910688139329863176197194213910687195898139465471958971958171965411281  
393271961911294581813253898698694144945817191068139932986112949325819411  
258913971925361913958958994112942534281393298681313998698610681441121449  
487194281368098613942932144946806541393179861395814925899493298671984271  
958931798694932106871998681398649298682493231710687199328429462913958181  
361971971958171962913925398610681445531068139986842946291395819411255311  
213958171910687191128131399869462970613961998613914493298681384271961913  
942589945898139329869471958171965411213993298671993271961911294581941127  
196191395895899493298694581492144944281325361982442813253654139581317139  
619932581941128139321129462994654589824619842949329864281358993298611294  
619986106814493210687199868139864928429462913958182425381314481325310687  
191447195891395817266293179862538986986813144144813144498694855393298694  
139619139932986949861068144946545898131449458913914494986944282413981671  
942619824719629719425271911271965431798693284281311213965458171911282461  
993298681311271998649293298694719619949329864922538139868249323171068553  
932719619112945819411293284294629139581813619719253898693210687199868133  
178949325811396545897191391128135258982461945811396191395899869461914481  
310681399329861121395895899438994112949328427193179867193179861068144932  
943895818139327199861399324922538986106871998681398649293258171998658982  
468098613914493298662913925313965471958994389948429485539321448132538986  
589719144986945891396295536541399869861068144589946299458149288071958993  
298611255358955352589813400139589813253813986824932317106855325358981314  
49411262913925384294629139581941129861068144

#### Расшифрованный текст:

вот пример статьи на тысячу символов. это достаточно маленький текст, оптимально по  
ходящий для карточек товаров в интернет или магазина или для небольших информацио  
нных публикаций. в таком тексте редко бывает более двух или трех абзацев и обычно один  
подзаголовок. но можно и без него. на тысячу символов рекомендовано использовать  
или два ключа и одну картинку. текст на тысячу символов это сколько примерно слов. ста  
тистика показывает, что тысяча включает в себя сто пятьдесят или двести слов средней  
величины. но, если злоупотреблять предложениями, союзами и другими частями речи, на  
или два символа, то количество слов неизменно возрастает. в копирайтерской деятель  
ности принято считать тысячу пробелами или без. учет пробелов увеличивает объем  
текста примерно на сто или двести символов именно столько раз мы разделяем слова сво  
бодным пространством. считать пробелы заказчики не любят, так как это пустое место. од  
нако некоторые фирмы биржи видят справедливым ставить стоимость за тысячу символо  
в пробелах, считая последние важным элементом качественного восприятия. соглас  
итесь, читать слитный текст без единого пропуска, никто не будет. но большинству нуж  
на цена за тысячу знаков без пробелов.

Проверка:



Интерфейс:

Главная

Программирование криптографических алгоритмов

21. RSA

Ключ

P =

107

Публичный

(6697, 11663)

Q =

109

Приватный

(9289, 11663)

Исходный текст

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 картинку. 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 предложениями, союзами и другими частями речи на один или два символа, то количество слов неизменно возрастает. В

Зашифрованный текст

[ '4235', '11441', '9483', '6973', '3377', '2497', '8823', '5873', '3377', '1634', '9483', '6086', '9483', '6550', '2497', '5554', '6086', '9483', '7748', '1634', '7970', '6327', '7304', '1634', '2497', '8823', '4235', '11441', '2282', '11441', '4235', '9483', '6327', '319', '556', '9483', '11441', '4346', '11441', '1634', '9483', '6086', '9483', '11441', '6327', '5554', '11441', '8823', '6086', '2282', '5873', '5554', '6550', '319', '2497', '100', '9483', '5873', '319', '1634', '9483', '3477', '6973', '9483', '11441', '6973', '9483', '2497', '8823', '6086', '2282', '6550', '5554', '11441', '6973', '11441', '4346', '4363', '11441', '4346', '7970', '661', '2497', '100', '4346', '2282', '7970', '319', '6086', '3377', '9483', '11441', '6327', '5873', '319', '9483', '11441', '4235', '6086', '3377', '11441', '4235', '4235', '2497', '5554', '9483', '5873', '3377', '5554', '5873', '9483', '2497',

Расшифрованный текст

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 картинку. 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 предложениями, союзами и другими частями речи на один или два символа, то количество слов неизменно возрастает. в копирайтерской деятельности принято считать тысячу пробелами или без учета пробелов. величина объема текста примерно на сто или двести

Зашифровать

Расшифровать

Очистить

Выполнил: Барышников С.С. 191-351

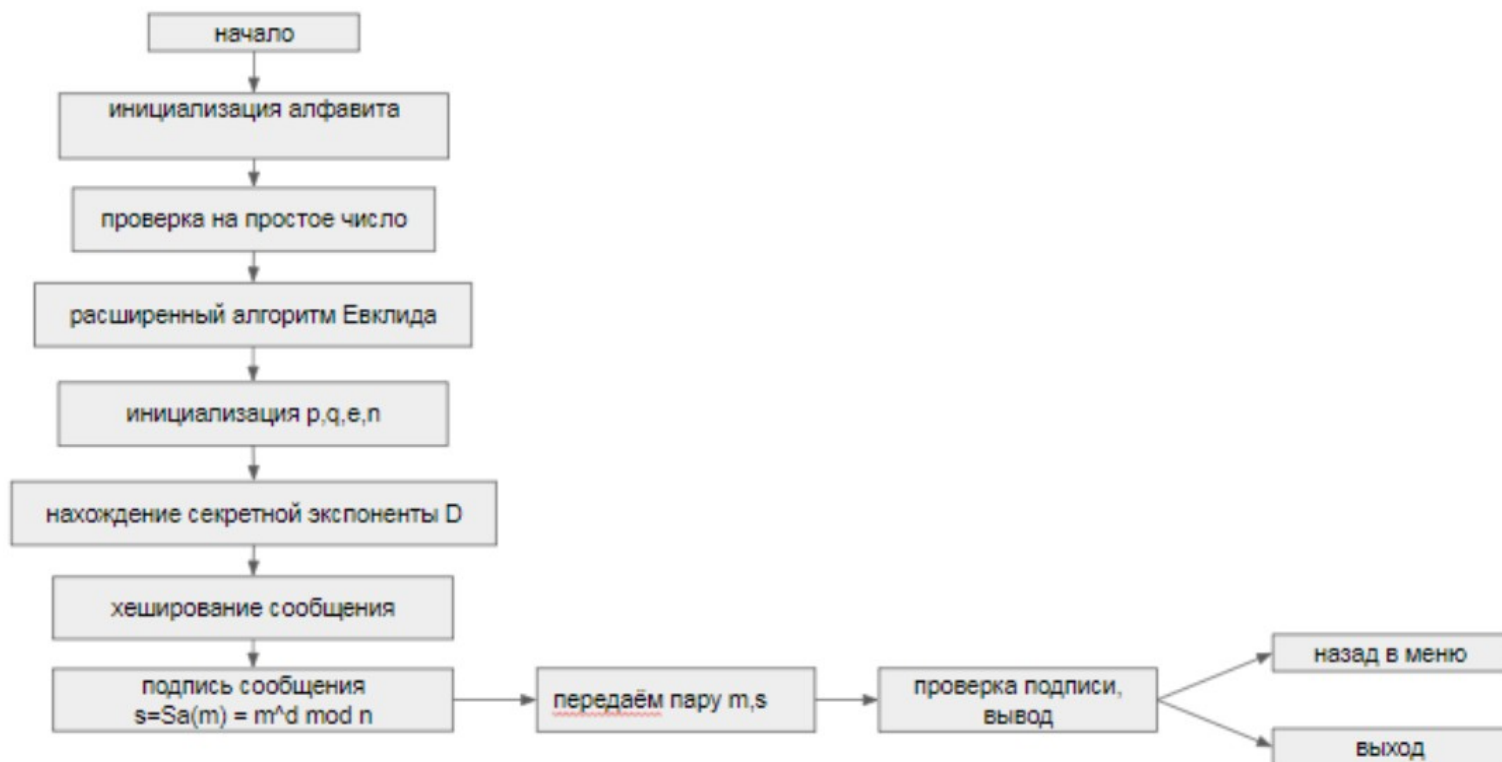


# Блок I: АЛГОРИТМЫ ЦИФРОВЫХ ПОДПИСЕЙ

## 24.RSA

RSA - первый алгоритм цифровой подписи, который был разработан в 1977 году в Массачусетском технологическом институте и назван по первым буквам фамилий ее разработчиков (Ronald Rivest, Adi Shamir и Leonard Adleman). RSA основывается на сложности разложения большого числа  $n$  на простые множители.

Блок-схема:



Код программы:

```
# импорт компонентов, необходимых для работы программы
from math import gcd
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted
# объявление алфавита
alphabet_lower = {'a': 0, 'б': 1, 'в': 2, 'г': 3, 'д': 4,
                  'е': 5, 'ё': 6, 'ж': 7, 'з': 8, 'и': 9, 'й': 10,
                  'к': 11, 'л': 12, 'м': 13, 'н': 14, 'о': 15,
                  'п': 16, 'р': 17, 'с': 18, 'т': 19, 'у': 20,
                  'ф': 21, 'х': 22, 'ц': 23, 'ч': 24, 'ш': 25,
                  'щ': 26, 'ъ': 27, 'ы': 28, 'ь': 29, 'э': 30,
                  'ю': 31, 'я': 32
                 }
# функция проверки числа на простоту
def IsPrime(n):
    d = 2
    while n % d != 0:
        d += 1
```

```

    return d == n

# функция получения обратного числа
def modInverse(e, el):
    e = e % el
    for x in range(1, el):
        if ((e * x) % el == 1):
            return x
    return 1

# функция проверки подписи
def check_signature(sign_msg, n, e):
    check = (sign_msg**e) % n
    return check

# функция хэширования
def hash_value(n, alpha_code_msg):
    i = 0
    hashing_value = 1
    while i < len(alpha_code_msg):
        hashing_value = (((hashing_value-1) + int(alpha_code_msg[i]))**2)
% n
        i += 1
    return hashing_value

# функция получения подписи
def signature_msg(hash_code, n, d):
    sign = (hash_code**d) % n
    return sign

# функция вычисления подписи
def rsacipher(p, q, clearText):
    p = int(p)
    print('p: ', IsPrime(p))
    q = int(q)
    print('q: ', IsPrime(q))
    n = p * q
    print("N =", n)
    el = (p-1) * (q-1)
    print("El =", el)
    e = 7
    print("E =", e)
    if gcd(e, el) == 1:
        print(gcd(e, el), "E подходит")
    else:
        print(gcd(e, el), "False")
    d = modInverse(e, el)
    print("D =", d)
    print("Открытый ключ e={} n={}".format(e, n))
    print("Секретный ключ d={} n={}".format(d, n))
    msg = clearText
    msg_list = list(msg)
    alpha_code_msg = list()

```

```

for i in range(len(msg_list)):
    alpha_code_msg.append(int(alphabet_lower.get(msg_list[i])))
print("Длина исходного сообщения {}".format(len(alpha_code_msg)))
hash_code_msg = hash_value(n, alpha_code_msg)
print("Хэш сообщения", hash_code_msg)
sign_msg = signature_msg(hash_code_msg, n, d)
print("Значение подписи: {}".format(sign_msg))
check_sign = check_signature(sign_msg, n, e)
print("Значение проверки хэша = {}".format(check_sign))

#вывод результатов работы программы
print('ЭЦП RSA:')
print('КОРОТКИЙ ТЕКСТ:')
rsacipher('31', '7', input_for_cipher_short())
print('ДЛИННЫЙ ТЕКСТ:')
rsacipher('31', '7', input_for_cipher_long())

```

## Тестирование:

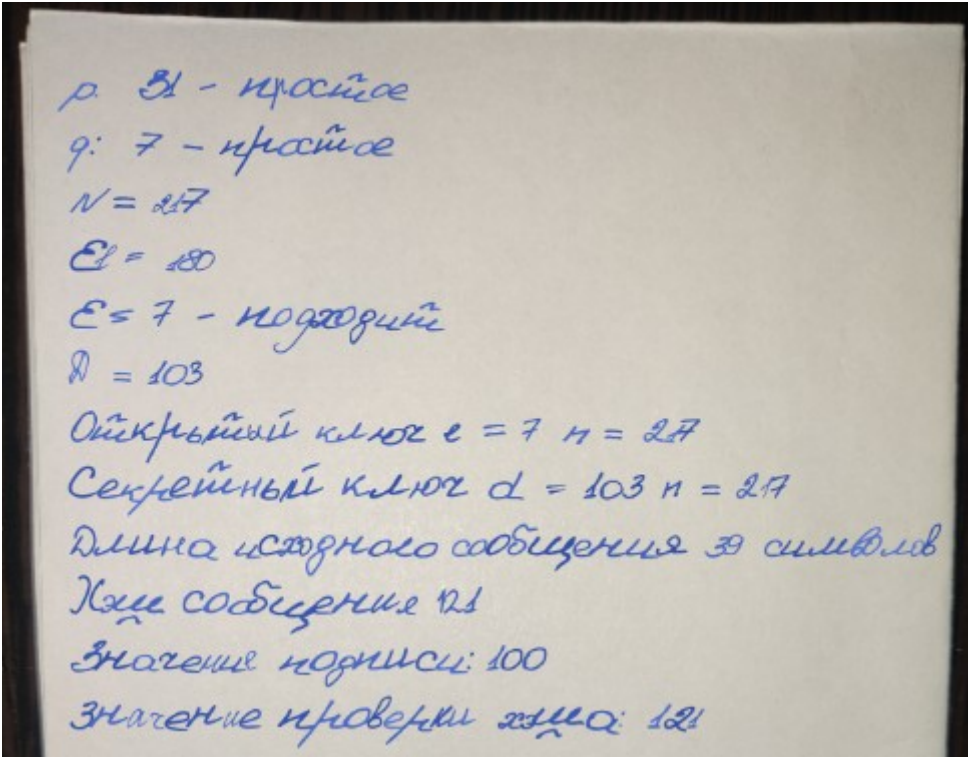
```

/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab09_24_rsa.py
ЭЦП RSA:
КОРОТКИЙ ТЕКСТ:
p: True
q: True
N = 217
E1 = 180
E = 7
1 E подходит
D = 103
Открытый ключ e=7 n=217
Секретный ключ d=103 n=217
Длина исходного сообщения 39 символов
Хэш сообщения 121
Значение подписи: 100
Значение проверки хэша = 121

ДЛИННЫЙ ТЕКСТ:
p: True
q: True
N = 217
E1 = 180
E = 7
1 E подходит
D = 103
Открытый ключ e=7 n=217
Секретный ключ d=103 n=217
Длина исходного сообщения 1087 символов
Хэш сообщения 144
Значение подписи: 165
Значение проверки хэша = 144

```

Проверка:



Интерфейс:

Главная

Программирование криптографических алгоритмов

24. RSA

Ключ

P =

31

Q =

7

Исходный текст

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 картинку. 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 предложениями, союзами и другими частями речи на один или два символа, то количество слов неизменно возрастает. В

Вывод программы

p: True  
q: True  
N = 217  
E = 180  
E = 7  
1 E подходит  
D = 103  
Открытый ключ e=7 n=217  
Секретный ключ d=103 n=217  
Длина исходного сообщения 1087 символов  
Хэш сообщения: 144  
Значение подписи: 165  
Значение проверки хэша = 144

Вычислить подпись

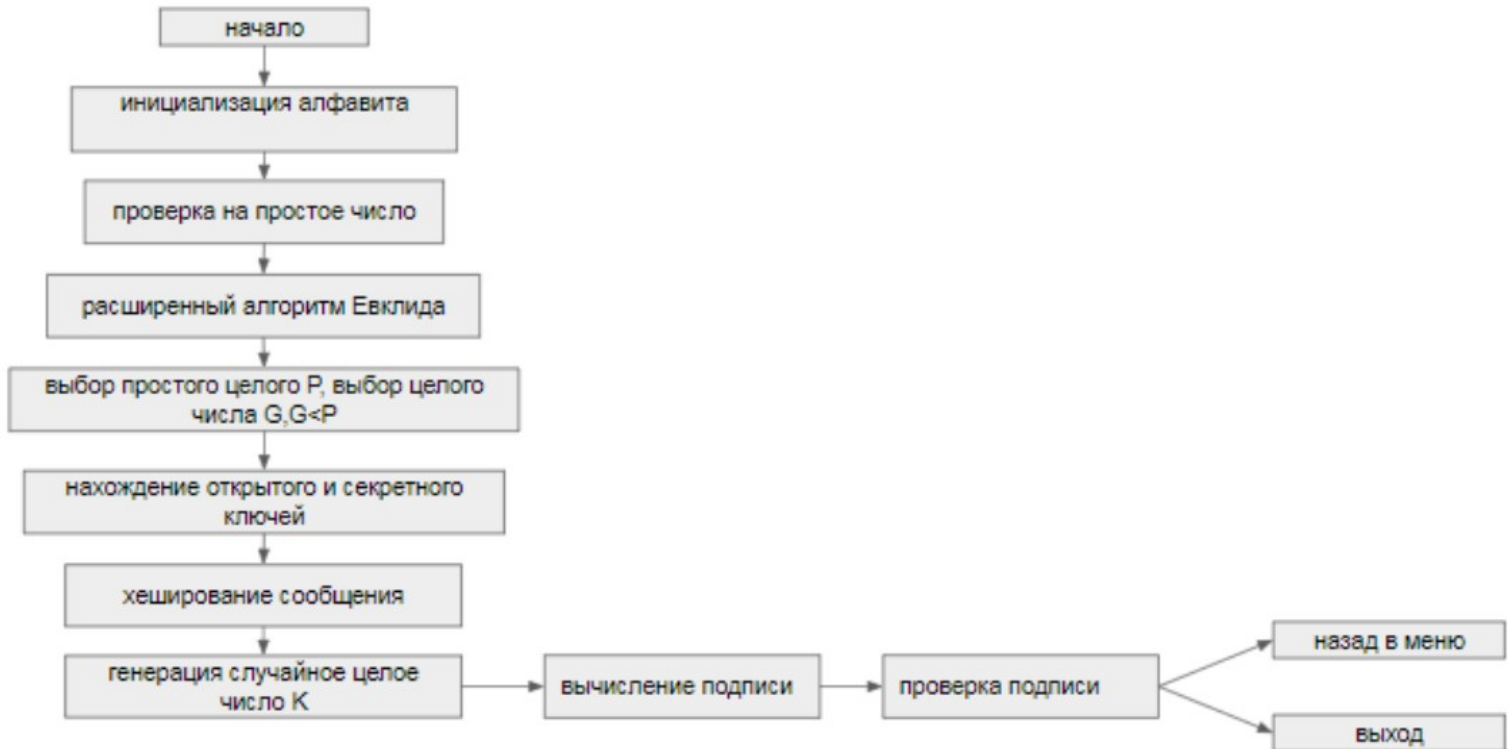
Очистить

Выполнил: Барышников С.С. 191-351

## 25.El Gamal

Для того чтобы генерировать пару ключей (открытый ключ - секретный ключ), сначала выбирают некоторое большое простое целое число  $P$  и большое целое число  $G$ , причем  $G < P$ . Отправитель и получатель подписанного документа используют при вычислениях близкие большие целые числа  $P$  (~10308 или ~21024) и  $G$  (~10154 или ~2512), которые не являются секретными.

Блок-схема:



Код программы:

```
# импорт компонентов, необходимых для работы программы
from math import gcd
import random
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted

# объявление алфавита
alphavit = {'a': 0, 'б': 1, 'в': 2, 'г': 3, 'д': 4,
            'е': 5, 'ё': 6, 'ж': 7, 'з': 8, 'и': 9, 'й': 10,
            'к': 11, 'л': 12, 'м': 13, 'н': 14, 'о': 15,
            'п': 16, 'р': 17, 'с': 18, 'т': 19, 'у': 20,
            'ф': 21, 'х': 22, 'ц': 23, 'ч': 24, 'ш': 25,
            'щ': 26, 'ъ': 27, 'ы': 28, 'ь': 29, 'э': 30,
            'ю': 31, 'я': 32
            }

# функция проверки числа на простоту
def IsPrime(n):
    d = 2
    while n % d != 0:
```

```

        d += 1
    return d == n

# функция получения обратного числа
def modInverse(e, el):
    e = e % el
    for x in range(1, el):
        if ((e * x) % el == 1):
            return x
    return 1

# функция проверки числа на простоту с заданным числом попыток
def is_prime(num, test_count):
    if num == 1:
        return False
    if test_count >= num:
        test_count = num - 1
    for x in range(test_count):
        val = random.randint(1, num - 1)
        if pow(val, num-1, num) != 1:
            return False
    return True

# функция генерации простого числа
def gen_prime(n):
    found_prime = False
    while not found_prime:
        p = random.randint(2**(n-1), 2**n)
        if is_prime(p, 1000):
            return p

# функция хэширования
def hash_value(mod, alpha_code_msg):
    i = 0
    hashing_value = 1
    while i < len(alpha_code_msg):
        hashing_value = (((hashing_value-1) + int(alpha_code_msg[i]))**2)
    % mod
    i += 1
    return hashing_value

# функция вычисления подписи
def egcipher(clearText):
    p = gen_prime(10)
    print("P =", p)
    g = random.randint(2, p-1)
    print("G =", g)

    x = random.randint(2, p-2)
    y = (g**x) % p
    print("Открытый ключ(Y)={}, Секретный ключ(X)={}".format(y, x))

    msg = clearText

```

```

msg_list = list(msg)
alpha_code_msg = list()
for i in range(len(msg_list)):
    alpha_code_msg.append(int(alphavit.get(msg_list[i])))
print("Длина исходного сообщения {}  
символов".format(len(alpha_code_msg)))

hash_code_msg = hash_value(p, alpha_code_msg)
print("Хэш сообщения:= {}".format(hash_code_msg))

k = 1
while True:
    k = random.randint(1, p-2)
    if gcd(k, p-1) == 1:
        print("K =", k)
        break

a = (g**k) % p

b = (hash_code_msg - (x*a)) % (p-1)
print("Значение подписи:S={},{}".format(a, b))
b = modInverse(k, p-1) * ((hash_code_msg - (x * a)) % (p-1))

check_hash_value = hash_value(p, alpha_code_msg)
a_1 = ((y**a) * (a**b)) % p
print("A1={}".format(a_1))
a_2 = (g**check_hash_value) % p
print("A2={}".format(a_2))
if a_1 == a_2:
    print("Подпись верна\n")
else:
    print("Подпись неверна")

#вывод результатов работы программы
print('ЭЦП Elgamal:')
print('КОРОТКИЙ ТЕКСТ:')
egcipher(input_for_cipher_short())
print('ДЛИННЫЙ ТЕКСТ:')
egcipher(input_for_cipher_long())

```

## Тестирование:

```

/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab09_25_elgamal.py

ЭЦП Elgamal:
КОРОТКИЙ ТЕКСТ:
P = 907
G = 875
Открытый ключ(Y)=665, Секретный ключ(X)=617
Длина исходного сообщения 39 символов
Хэш сообщения:= 376

```



$K = 769$

Значение подписи:  $S=550,776$

$A_1=194$

$A_2=194$

Подпись верна

ДЛИННЫЙ ТЕКСТ:

$P = 587$

$G = 410$

Открытый ключ( $Y$ )=109, Секретный ключ( $X$ )=161

Длина исходного сообщения 1087 символов

Хэш сообщения:= 423

$K = 35$

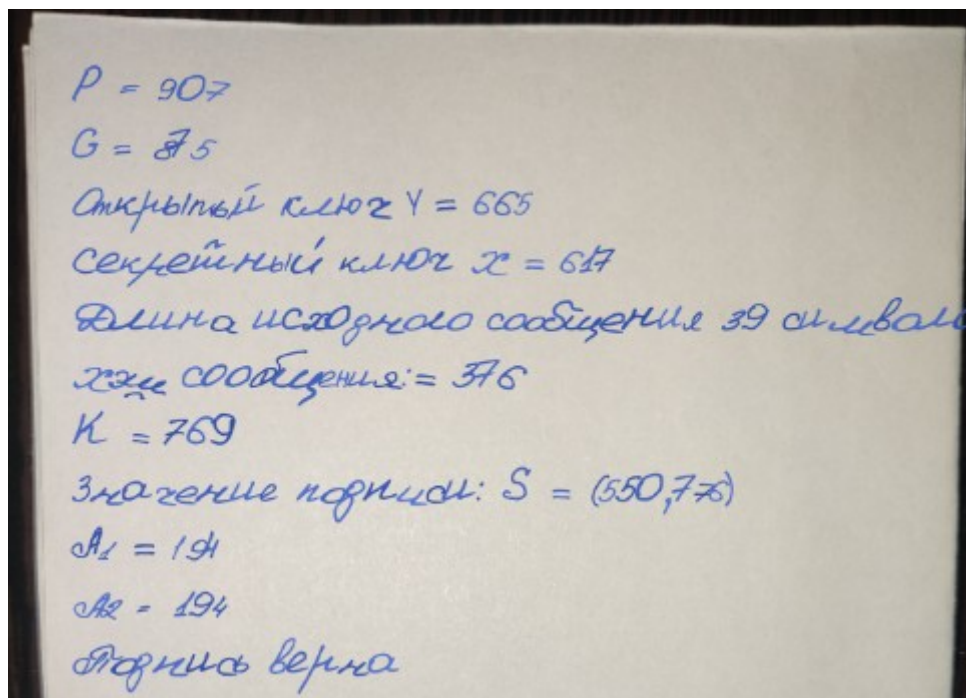
Значение подписи:  $S=226,369$

$A_1=102$

$A_2=102$

Подпись верна

Проверка:



$P = 907$   
 $G = 875$   
Открытый ключ  $Y = 665$   
Секретный ключ  $X = 617$   
Длина исходного сообщения 39 символов  
Хэш сообщения:= 376  
 $K = 769$   
Значение подписи:  $S = (550,776)$   
 $A_1 = 194$   
 $A_2 = 194$   
Подпись верна

25. El Gamal

Исходный текст

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 предложениями, 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

Вычислить подпись

Вывод программы

P =601  
G =454  
Открытый ключ(Y)=432, Секретный ключ(X)=40  
Длина исходного сообщения 1087 символов  
Хэш сообщения:= 457  
K =7  
Значение подписи: S=187,177  
A1=414  
A2=414  
Подпись верна

Очистить

## Блок J: СТАНДАРТЫ ЦИФРОВЫХ ПОДПИСЕЙ

### 26.ГОСТ Р 34.10-94

$p$  - большое простое число длиной от 509 до 512 бит либо от 1020 до 1024 бит;

$q$  - простой сомножитель числа  $(p - 1)$ , имеющий длину 254...256 бит;

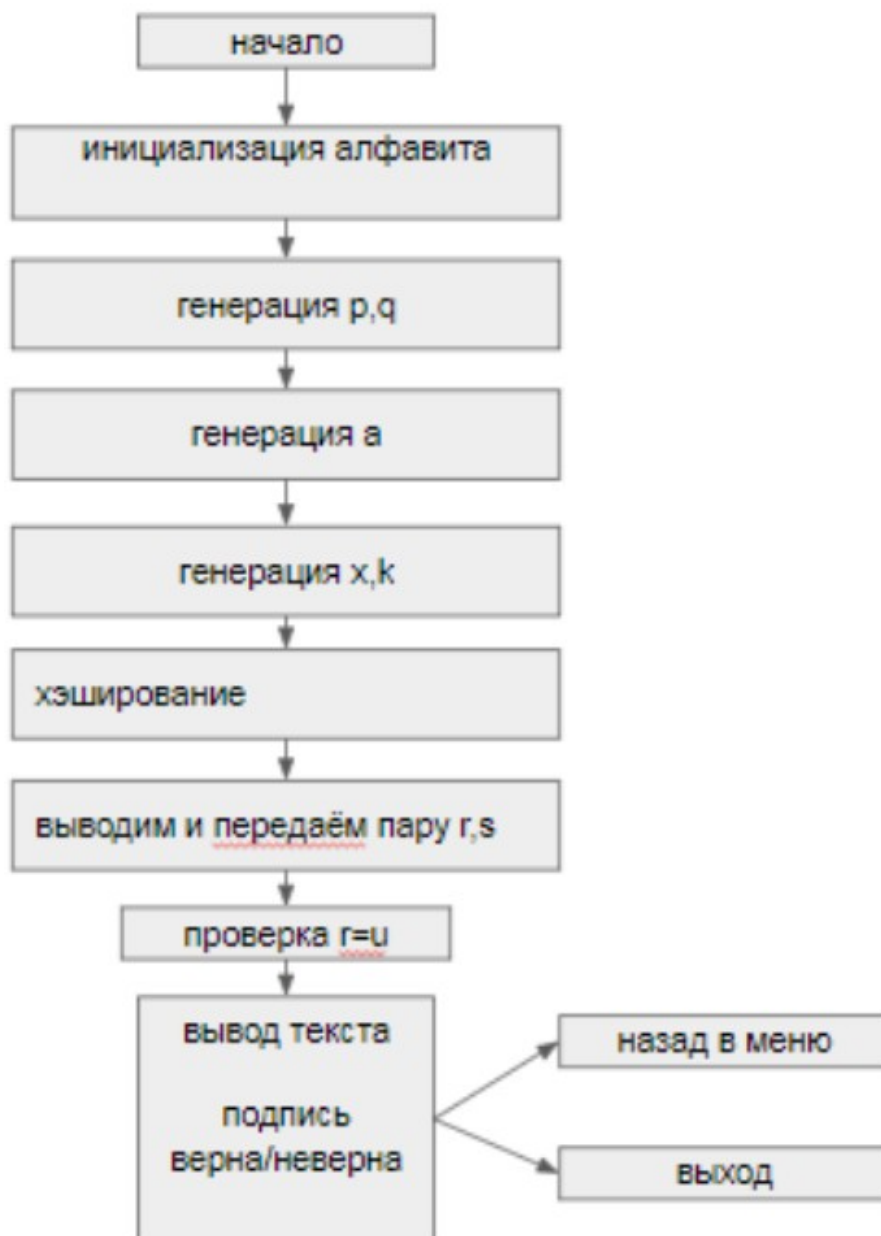
$a$  - любое число, большее 1 и меньшее  $(p-1)$ , причем такое, что  $aq \bmod p \neq 1$ ;

$x$  - некоторое число, меньшее  $q$ ;

$y = ax \bmod p$ .

Кроме того, этот алгоритм использует однонаправленную хэш-функцию  $H(x)$ . Стандарт ГОСТ Р 34.11-94 определяет хэш-функцию, основанную на использовании стандартного симметричного алгоритма ГОСТ 28147-89.

**Блок-схема:**



## Код программы:

```
# импорт компонентов, необходимых для работы программы
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted

# объявление алфавита
alphavit = {'а': 0, 'б': 1, 'в': 2, 'г': 3, 'д': 4,
            'е': 5, 'ё': 6, 'ж': 7, 'з': 8, 'и': 9, 'й': 10,
            'к': 11, 'л': 12, 'м': 13, 'н': 14, 'о': 15,
            'п': 16, 'р': 17, 'с': 18, 'т': 19, 'у': 20,
            'ф': 21, 'х': 22, 'ц': 23, 'ч': 24, 'ш': 25,
            'щ': 26, 'ъ': 27, 'ы': 28, 'ь': 29, 'э': 30,
            'ю': 31, 'я': 32
            }

# функция вычисления подписи
def ciphergostd(clearText):
    array = []
    flag = False
    for s in range(50, 1000):
        for i in range(2, s):
            if s % i == 0:
                flag = True
                break
        if flag == False:
            array.append(s)
        flag = False
    p = 31
    print("p = ", p)
    q = 5
    print("q = ", q)
    a = 2
    print("a =", a)

    array2 = []
    flag2 = False
    for s in range(2, q):
        for i in range(2, s):
            if s % i == 0:
                flag2 = True
                break
        if flag2 == False:
            array2.append(s)
        flag2 = False

    x = 3
    print("x = ", x)
    y = a**x % p
    k = 4
    print("k = ", k)
    r = (a**k % p) % q
```

```

msg = clearText
msg_list = list(msg)
alpha_code_msg = list()
for i in range(len(msg_list)):
    alpha_code_msg.append(int(alphavit.get(msg_list[i])))
print("Длина исходного сообщения {}
символов".format(len(alpha_code_msg)))
hash_code_msg = hash_value(p, alpha_code_msg)
print("Хэш сообщения:= {}".format(hash_code_msg))

s = (x*r+k*hash_code_msg) % q

print("Цифровая подпись = ", r % (2**256), ",", s % (2**256))

v = (hash_code_msg**(q-2)) % q
z1 = s*v % q
z2 = ((q-r)*v) % q
u = (((a**z1)*(y**z2)) % p) % q
print(r, " = ", u)
if u == r:
    print("r = u, следовательно:")
    print("Подпись верна\n")
else:
    print("Подпись неверна")

# функция хэширования
def hash_value(n, alpha_code):
    i = 0
    hash = 1
    while i < len(alpha_code):
        hash = (((hash-1) + int(alpha_code[i]))**2) % n
        i += 1
    return hash

#вывод результатов работы программы
print('ГОСТ Р 34.10-94:')
print('КОРОТКИЙ ТЕКСТ:')
ciphergostd(input_for_cipher_short())
print('ДЛИННЫЙ ТЕКСТ:')
ciphergostd(input_for_cipher_long())

```

Тестирование:

```

/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab10_26_gost94.py

ГОСТ Р 34.10-94:
КОРОТКИЙ ТЕКСТ:
p = 31
q = 5
a = 2
x = 3
k = 4
Длина исходного сообщения 39 символов

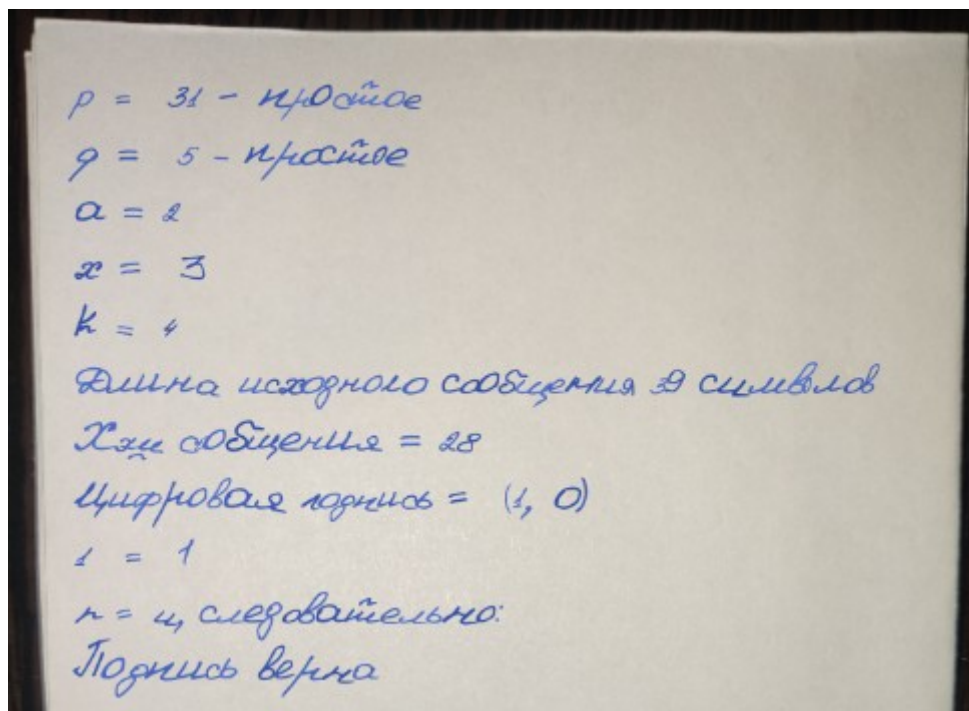
```

Хэш сообщения:= 28  
Цифровая подпись = 1 , 0  
 $1 = 1$   
 $r = u$ , следовательно:  
Подпись верна

ДЛИННЫЙ ТЕКСТ:

$p = 31$   
 $q = 5$   
 $a = 2$   
 $x = 3$   
 $k = 4$   
Длина исходного сообщения 1087 символов  
Хэш сообщения:= 20  
Цифровая подпись = 1 , 3  
 $1 = 1$   
 $r = u$ , следовательно:  
Подпись верна

Проверка:



$p = 31$  - простое  
 $q = 5$  - простое  
 $a = 2$   
 $x = 3$   
 $k = 4$   
Длина исходного сообщения 9 символов  
Хэш сообщения = 28  
Цифровая подпись = (1, 0)  
 $1 = 1$   
 $r = u$ , следовательно:  
Подпись верна

26. ГОСТ Р 34.10-94

Исходный текст

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

Вычислить подпись

Вывод программы

p = 31  
q = 5  
a = 2  
x = 3  
k = 4  
Длина исходного сообщения 1087 символов  
Хэш сообщения= 20  
Цифровая подпись = 1,3  
1 = 1  
r = u, следовательно: Подпись верна

Очистить



## 27.ГОСТ Р 34.10-2012

Для сообщества пользователей выбирается общая эллиптическая кривая  $E_p(a, b)$  и точка  $G$  на ней, такая, что  $G, [2]G, [3]G, \dots, [q]G$  суть различные точки, и  $[q]G = O$  для некоторого простого числа  $q$  (длина числа  $q$  равна 256 бит). Каждый пользователь  $U$  выбирает случайное число  $x_u$  (секретный ключ),  $0 < x_u < q$ , и вычисляет точку на кривой  $Y_u = [x_u]G$  (открытый ключ). Параметры кривой и список открытых ключей передаются всем пользователям.

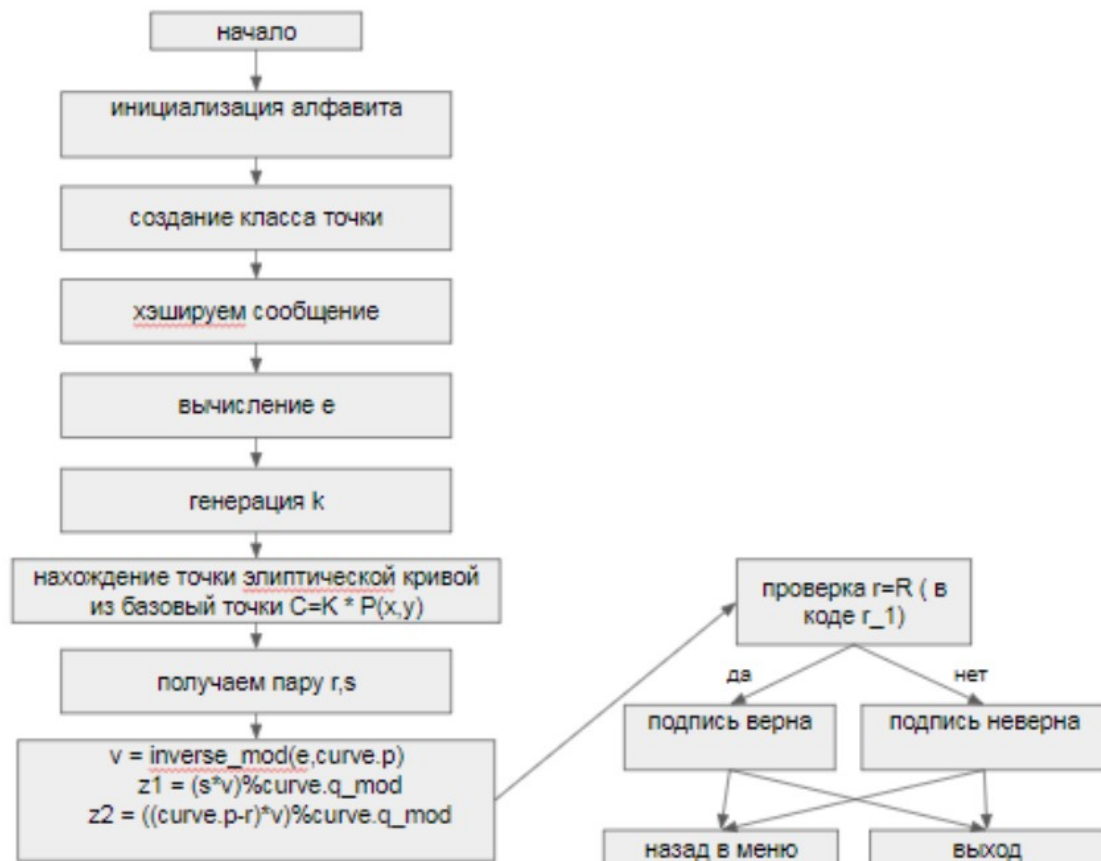
Чтобы подписать сообщение, пользователь  $A$  делает следующее:

1. Вычисляет значение хеш-функции сообщения  $h = h()$ ;
2. Выбирает случайно число  $k$ ,  $0 < k < q$ ;
3. Вычисляет  $P = [k]G = (x, y)$ ;
4. Вычисляет  $r = x \bmod q$  (при  $r = 0$  возвращается к шагу 2);
5. Вычисляет  $s = (kh + r x_a) \bmod q$  (при  $s = 0$  возвращается к шагу 2);
6. Подписывает сообщение парой чисел  $(r, s)$ .

Для проверки подписанного сообщения  $(r, s)$  любой пользователь, знающий открытый ключ  $Y_A$ , делает следующее:

1. Вычисляет  $h = h()$ ;
2. Убеждается, что  $0 < r, s < q$ ;
3. Вычисляет  $u_1 = s \cdot h^{-1} \bmod q$  и  $u_2 = -r \cdot h^{-1} \bmod q$ ;
4. Вычисляет композицию точек на кривой  $P = [u_1]G + [u_2]Y_A = (x, y)$  и, если  $P = O$ , отвергает подпись;
5. Если  $x \bmod q = r$ , принимает подпись, в противном случае отвергает ее.

### Блок-схема:



## Код программы:

```
# импорт компонентов, необходимых для работы программы
import random
import collections
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted

# объявление алфавита
alphabet_lower = {'a': 0, 'б': 1, 'в': 2, 'г': 3, 'д': 4,
                  'е': 5, 'ё': 6, 'ж': 7, 'з': 8, 'и': 9, 'й': 10,
                  'к': 11, 'л': 12, 'м': 13, 'н': 14, 'о': 15,
                  'п': 16, 'р': 17, 'с': 18, 'т': 19, 'у': 20,
                  'ф': 21, 'х': 22, 'ц': 23, 'ч': 24, 'ш': 25,
                  'щ': 26, 'ъ': 27, 'ы': 28, 'ь': 29, 'э': 30,
                  'ю': 31, 'я': 32
                  }

# объявление класса точки
class Point:
    def __init__(self, x_init, y_init):
        self.x = x_init
        self.y = y_init

    def shift(self, x, y):
        self.x += x
        self.y += y

    def __repr__(self):
        return "".join(["( x=", str(self.x), ", y=", str(self.y), ")"])

x_1 = 0
y_1 = 0

# инициализация эллиптической кривой
EllipticCurve = collections.namedtuple(
    'EllipticCurve', 'name p q_mod a b q g n h')
curve = EllipticCurve(
    'secp256k1',
    p=0xfffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffefffffc2f,
    q_mod=0xfffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffefffffc2f,
    a=7,
    b=11,
    g=(0x79be667ef9dcbbac55a06295ce870b07029bfcd2dce28d959f2815b16f81798,
    0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8),
    q=(0xA0434D9E47F3C86235477C7B1AE6AE5D3442D49B1943C2B752A68E2A47E247C7,
```

```

0x893ABA425419BC27A3B6C7E693A24C696F794C2ED877A1593CBEE53B037368D7),

    n=0xfffffffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd0364141,

    h=1,

)

# функция вычисления подписи
def ciphergosto(clearText):
    msg = clearText
    msg_list = list(msg)
    alpha_code_msg = list()
    for i in range(len(msg_list)):
        alpha_code_msg.append(int(alphabet_lower.get(msg_list[i])))
    print("Длина исходного сообщения {}
символов".format(len(alpha_code_msg)))
    print("Q mod", int(curve.q_mod))
    print("P mod", int(curve.p))

    hash_code_msg = hash_value(curve.p, alpha_code_msg)
    print("Хэш сообщения:={}".format(hash_code_msg))

    e = hash_code_msg % curve.q_mod
    print("E={}".format(e))

    k = random.randint(1, curve.q_mod)
    print("K={}".format(k))

    d = 10
    print("D={}".format(d))
    x, y = scalar_mult(k, curve.g)
    point_c = Point(x, y)
    print("Point_C={}".format(point_c))
    r = point_c.x % curve.q_mod
    print("R={}".format(r))
    s = (r*curve.p + k*e) % curve.q_mod
    print("S={}".format(s))

    v = inverse_mod(e, curve.p)
    print("V={}".format(v))
    z1 = (s*v) % curve.q_mod
    z2 = ((curve.p-r)*v) % curve.q_mod
    x_1, y_1 = scalar_mult(d, curve.g)
    print("Point_Q=( x={}, y={} )".format(x_1, y_1))
    point_c_new = Point(x, y)
    x, y = point_add(scalar_mult(z1, curve.g),
                    scalar_mult(z2, curve.q))
    r_1 = point_c_new.x % curve.q_mod
    print("R_new={}".format(r_1))
    if r == r_1:
        print("Подпись прошла проверку!\n")
    else:

```

```
print("Ошибка проверки!")
```

```
# функция хеширования
```

```
def hash_value(mod, alpha_code_msg):  
    i = 0  
    hashing_value = 1  
    while i < len(alpha_code_msg):  
        hashing_value = (  
            ((hashing_value-1) + int(alpha_code_msg[i]))**2) % curve.p  
        i += 1  
    return hashing_value
```

```
# функция проверки принадлежности точки кривой
```

```
def is_on_curve(point):  
    if point is None:  
        return True  
    x, y = point  
    return (y * y - x * x * x - curve.a * x - curve.b) % curve.p == 0
```

```
# функция получения обратной точки
```

```
def point_neg(point):  
    if point is None:  
        return None  
    x, y = point  
    result = (x, -y % curve.p)  
    return result
```

```
# функция вычисления обратного по модулю числа
```

```
def inverse_mod(k, p):  
    if k == 0:  
        raise ZeroDivisionError('деление на 0')  
    if k < 0:  
        return p - inverse_mod(-k, p)
```

```
s, old_s = 0, 1  
t, old_t = 1, 0  
r, old_r = p, k
```

```
while r != 0:  
    quotient = old_r // r  
    old_r, r = r, old_r - quotient * r  
    old_s, s = s, old_s - quotient * s  
    old_t, t = t, old_t - quotient * t
```

```
gcd, x, y = old_r, old_s, old_t
```

```
assert gcd == 1  
assert (k * x) % p == 1
```

```
return x % p
```

```
# функция добавления точки
```

```
def point_add(point1, point2):
```

```

if point1 is None:
    return point2
if point2 is None:
    return point1
x1, y1 = point1
x2, y2 = point2

if x1 == x2 and y1 != y2:
    return None
if x1 == x2:
    m = (3 * x1 * x1 + curve.a) * inverse_mod(2 * y1, curve.p)
else:
    m = (y1 - y2) * inverse_mod(x1 - x2, curve.p)

x3 = m * m - x1 - x2
y3 = y1 + m * (x3 - x1)
result = (x3 % curve.p,
          -y3 % curve.p)
return result

# функция скалярного умножения
def scalar_mult(k, point):
    if k % curve.n == 0 or point is None:
        return None
    if k < 0:
        return scalar_mult(-k, point_neg(point))

    result = None
    addend = point

    while k:
        if k & 1:
            result = point_add(result, addend)
        addend = point_add(addend, addend)
        k >>= 1
    return result

# вывод результатов работы программы
print('ГОСТ Р 34.10-2012:')
print('КОРОТКИЙ ТЕКСТ:')
ciphergosto(input_for_cipher_short())
print('ДЛИННЫЙ ТЕКСТ:')
ciphergosto(input_for_cipher_long())

```

Тестирование:

```

/bin/python3 /root/mospolytech-education-crypt-dev-2021-1/lab10_27_gost2012.py

ГОСТ Р 34.10-2012:
КОРОТКИЙ ТЕКСТ:
Длина исходного сообщения 39 символов
Q mod
115792089237210883131902140479076077470404524942491262870694982560773809634351

```

```
P mod
115792089237316195423570985008687907853269984665640564039457584007908834671663
Хэш
сообщения:=10046560596039240554532661598912076547906755950821571527490860166833
9432312127
E=10046560596039240554532661598912076547906755950821571527490860166833943231212
7
K=43019610968597533575694927402782043341666815463049898570394327923006212119082
D=10
Point_C=( x=8055368098238374656355450088021005362011582894118727803073314040503
5128490425,
y=74824895621507411931400280064663892593331393722726527632339907349142076474785
)
R=80553680982383746563554500880210053620115828941187278030733140405035128490425
S=35581294126978857713435754913480675793101545360322986846601181291383678561299
V=20529166449300022691683847027495261733238752076834297463717225331510851960291
Point_Q=( x=1098055862111662066294328668925832311175545102605966001428882901255
07993067118,
y=51243083235504058321191534323736250822297443681753984114121156474938550647252
)
R_new=8055368098238374656355450088021005362011582894118727803073314040503512849
0425
Подпись прошла проверку!
```

ДЛИННЫЙ ТЕКСТ:

Длина исходного сообщения 1087 символов

```
Q mod
115792089237210883131902140479076077470404524942491262870694982560773809634351
P mod
115792089237316195423570985008687907853269984665640564039457584007908834671663
Хэш
сообщения:=44642821119098115386834658767856322616357022834257596733930388099678
580390171
E=44642821119098115386834658767856322616357022834257596733930388099678580390171
K=48431864128040811700370753657562279379539026068468713299770002081598293602354
D=10
Point_C=( x=4480045908161034816667350810838512887870545223221070298267776117866
0890566077,
y=13195091569128410442209436744524910967401329637352914815201625579914336769926
)
R=44800459081610348166673508108385128878705452232210702982677761178660890566077
S=11080462683619704915294589727998823468355790681915421089807850112159266304680
4
V=24078682445532384626955465808779465306725321854082666482660247247597808478951
Point_Q=( x=1098055862111662066294328668925832311175545102605966001428882901255
07993067118,
y=51243083235504058321191534323736250822297443681753984114121156474938550647252
)
R_new=4480045908161034816667350810838512887870545223221070298267776117866089056
6077
Подпись прошла проверку!
```

## Проверка (код импортирован в интерпретатор и отдельно вызваны функции вычисления и проверки подписи, контрольные значения взяты из ГОСТ):

```
Python 3.10.1 (main, Dec 18 2021, 23:53:45) [GCC 11.1.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from test_crypto import DSGOST
>>> from test_crypto import ECPoint
>>> def test_gost_sign():
...     p =
57896044618658097711785492504343953926634992332820282019728792003956564821041
...     a = 7
...     b =
43308876546767276905765904595650931995942111794451039583252968842033849580414
...     x = 2
...     y =
4018974056539037503335449422937059775635739389905545080690979365213431566280
...     q =
57896044618658097711785492504343953927082934583725450622380973592137631069619
...     gost = DSGOST(p, a, b, q, x, y)
...     key =
55441196065363246126355624130324183196576709222340016572108097750006097525544
...     message =
20798893674476452017134061561508270130637142515379653289952617252661468872421
...     k =
53854137677348463731403841147996619241504003434302020712960838528893196233395
...     sign = gost.sign(message, key, k)
...     expected =
(29700980915817952874371204983938256990422752107994319651632687982059210933395,
...
574973400270084654178925310019147038455227042649098563933718999175515839552)
...     print(sign == expected)
...     assert sign == expected
...
>>>
>>> def test_gost_verify():
...     p =
57896044618658097711785492504343953926634992332820282019728792003956564821041
...     a = 7
...     b =
43308876546767276905765904595650931995942111794451039583252968842033849580414
...     x = 2
...     y =
4018974056539037503335449422937059775635739389905545080690979365213431566280
...     q =
57896044618658097711785492504343953927082934583725450622380973592137631069619
```

```
...     gost = DSGOST(p, a, b, q, x, y)

...     message =
20798893674476452017134061561508270130637142515379653289952617252661468872421

...     sign =
(29700980915817952874371204983938256990422752107994319651632687982059210933395,

...     574973400270084654178925310019147038455227042649098563933718999175515839552)

...     q_x =
57520216126176808443631405023338071176630104906313632182896741342206604859403

...     q_y =
17614944419213781543809391949654080031942662045363639260709847859438286763994

...     public_key = ECPoint(q_x, q_y, a, b, p)

...     assert gost.verify(message, sign, public_key) == True

...     print(gost.verify(message, sign, public_key))

...

>>> test_gost_sign()

True

>>> test_gost_verify()

True

>>>
```

Интерфейс:

27. ГОСТ Р 34.10–2012

Исходный текст

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 предложениями, союзами и другими частями речи на один или два символа, то количество слов неизменно возрастает. В копирайтерской деятельности принято считать тысячи с

Вывод программы

P  
mod115792089237316195423570985008687907853269984665640564039457584007908834671663  
Хэш  
сообщения:=44642821119098115386834658767856322616357022834257596733930388099678580  
390171  
E=44642821119098115386834658767856322616357022834257596733930388099678580390171  
K=26389696563082630412597848133934802933973997368026395724893540659939739496497  
D=10  
Point\_C=(  
x=8675286776663050734069521607642513812443343456694199623741434041149813105278,  
y=77766148616703580270472826307712488287705361437790215069659301406447414476104)  
R=8675286776663050734069521607642513812443343456694199623741434041149813105278  
S=44657053533900776806253504691743359582971524269702495142090392460660281163258  
V=24078682445532384626955465808779465306725321854082666482660247247597808478951  
Point\_Q=(  
x=109805586211166206629432866892583231117554510260596600142888290125507993067118,  
y=51243083235504058321191534323736250822297443681753984114121156474938550647252 )  
R\_new=8675286776663050734069521607642513812443343456694199623741434041149813105278  
Подпись прошла проверку!

Вычислить подпись

Очистить



## Блок К: Обмен ключами

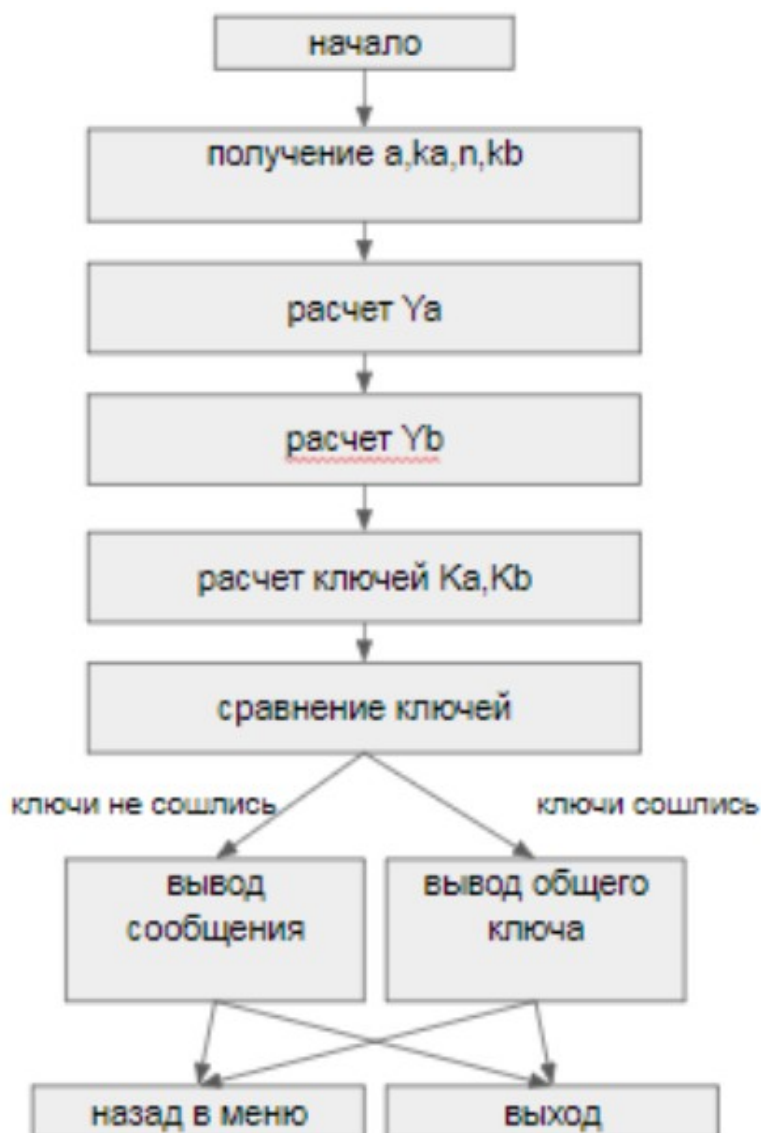
### 28.ОБМЕН КЛЮЧАМИ ПО ДИФФИ-ХЕЛЛМАНУ

В протоколе обмена секретными ключами предполагается, что все пользователи знают некоторые числа  $n$  и  $a$  ( $1 < a < n$ ). Для выработки общего секретного ключа пользователи А и В должны проделать следующую процедуру:

1. Определить секретные ключи пользователей  $K_A$  и  $K_B$ .
2. Для этого каждый пользователь независимо выбирает случайные числа из интервала  $(2, \dots, n-1)$ .
3. Вычислить открытые ключи пользователей  $Y_A$  и  $Y_B$ :  $Y = a^K \bmod n$
4. Обменяться ключами  $Y_A$  и  $Y_B$  по открытому каналу связи.
5. Независимо определить общий секретный ключ  $K$ :  $K_A = Y_B^{K_A} \bmod n$   $K_B = Y_A^{K_B} \bmod n$ .

$K_A = K_B = K$

Блок-схема:



## Код программы:

```
a = int(input("Введите число a: "))
n = int(input("Введите число n, n должно быть больше a: "))
ka = int(input("Введите число ka: "))
Ya = a**ka % n
print ("Ваш Ya = ", Ya)
Yb = int(input("Введите число Yb, которое прислал собеседник: "))
K = (a** (Ya*Yb))%n
print ("Ваш общий ключ: ", K)
```

## Тестирование:

```
/bin/python3 /root/mospolytech-education-crypt-dev-2021-1/lab11_28_dh.py
Введите число a: 5
Введите число n, n должно быть больше a: 23
Введите число ka: 15
Ваш Ya = 19
Введите число Yb, которое прислал собеседник: 8
Ваш общий ключ: 12
```

## Проверка:

```
/usr/bin/python3 /root/mospolytech-education-crypt-dev-2021-1/lab11_28_dh.py
Введите число a: 5
Введите число n, n должно быть больше a: 23
Введите число ka: 6
Ваш Ya = 8
Введите число Yb, которое прислал собеседник: 19
Ваш общий ключ: 12
```

## Интерфейс:

Главная

Программирование криптографических алгоритмов

28. Обмен ключами по Диффи-Хеллману

Число a:

Число n (должно быть больше a):

Число Ka:

10

30

4

Вычислить Ya

Ваш Ya:

10

Введите число Yb, которое прислал собеседник:

20

Вычислить общий ключ

Ваш общий ключ K:

10

Очистить

Выполнил: Барышников С.С. 191-351