

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ
«Московский Политехнический Университет»

ПРОГРАММИРОВАНИЕ КРИПТОГРАФИЧЕСКИХ АЛГОРИТМОВ

ФИНАЛЬНЫЙ ОТЧЕТ

На 102 листах

Преподаватель:

Бутакова Н.Г.

Выполнил:

Студент группы 191-351

Филиппович Владислав

2021

СОДЕРЖАНИЕ

1 ОБЩАЯ ИНФОРМАЦИЯ	4
1.1 Среда программирования	4
1.2 Среда программирования	4
1.3 Процедуры для запуска программы	4
1.4 Пословица – тест	4
1.5 Текст для проверки работы (1000 символов)	4
1.6 Интерфейс	5
2 БЛОК А: ШИФРЫ ОДНОЗНАЧНОЙ ЗАМЕНЫ	7
2.1 Шифр простой замены АТБАШ(1)	7
2.2 Шифр ЦЕЗАРЯ (2)	9
3 БЛОК В: ШИФРЫ МНОГОЗНАЧНОЙ ЗАМЕНЫ.....	13
3.1 Шифр БЕЛЛАЗО (4)	13
3.2 Шифр ТРИТЕМИЯ (5).....	16
3.3 Шифр ВИЖЕНЕРА (6)	19
4 БЛОК С: ШИФРЫ БЛОЧНОЙ ЗАМЕНЫ	25
4.1 Шифр МАТРИЧНЫЙ (8)	25
4.2 Шифр ПЛЕЙФЕРА (9).....	28
5 БЛОК D: ШИФРЫ ПЕРЕСТАНОВКИ	37
5.1 Шифр ВЕРТИКАЛЬНОЙ ПЕРЕСТАНОВКИ (10)	37
5.2 Шифр РЕШЕТКА КАРДАНО (11).....	40
6 БЛОК Е: ШИФРЫ ГАММИРОВАНИЯ	45
6.1 Шифр ГАММИРОВАНИЕ ГОСТ 28147-89 (13)	45
6.1 Шифр ОДНОРАЗОВЫЙ БЛОКНОТ ШЕННОНА (14).....	50
7 БЛОК F: ПОТОЧНЫЕ ШИФРЫ.....	55

7.1 Шифр A5/1 (15)	55
8 БЛОК G: КОМБИНАЦИОННЫЕ ШИФРЫ.....	62
8.1 Шифр ГОСТ 28147-89 (18).....	62
8.2 Шифр AES (19).....	69
9 БЛОК H: АСИММЕТРИЧНЫЕ ШИФРЫ	77
9.1 Шифр RSA (21).....	77
10 БЛОК I: АЛГОРИТМЫ ЦИФРОВЫХ ПОДПИСЕЙ.....	82
10.1 ЭЦП RSA (24).....	82
10.2 ЭЦП Elgamal (25)	86
11 БЛОК J: СТАНДАРТЫ ЦИФРОВЫХ ПОДПИСЕЙ	93
11.1 Стандарт ГОСТ Р 34.10-94 (26)	93
11.2 Стандарт Р 34.10-2012 (27).....	97
12 БЛОК K: ОБМЕН КЛЮЧАМИ.....	106
12.1 ОБМЕН КЛЮЧАМИ ПО ДИФФИ-ХЕЛЛМАНУ(28)	106

1 ОБЩАЯ ИНФОРМАЦИЯ

1.1 Среда программирования

Первые лабораторные работы были выполнены на языке **JavaScript**. Далее, по разрешению преподавателя, язык программирования был сменен на Python, который использовался студентом до окончания работы по курсу.

1.2 Среда программирования

Для написания и отладки написанного программного кода использовалось приложение **Visual Studio Code** от Microsoft. Приложение обладает возможностью быстрого запуска и проверки кода. В случае выявления каких-либо ошибок пользователь получает комментарии и предложения об исправлении.

1.3 Процедуры для запуска программы

Для запуска большинства лабораторных работ был создан общий файл, который обрабатывает введенные пользователем значения и отвечает за запуск самих алгоритмов и вывод результата на экран. Для запуска этого файла необходимо написать следующее, при этом находясь в директории с проверяемой лабораторной работой:

- *python3 demo.py*

1.4 Пословица – тест

В прошлом семестре студенту был присужден 22 вариант, пословица которого звучит следующим образом:

Красивыми словами пастернак не помаслишь

1.5 Текст для проверки работы (1000 символов)

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.

1.6 Интерфейс

Студентом был выбран консольный формат интерфейса. При запуске программы в консоли его встречает текстовое сопровождение, которое поможет правильно ввести требуемое значение. Ввод всех требуемых значений производится так же через консоль.

Вид интерфейса и код-программы интерфейса представлены ниже:

```
# -*- coding: utf-8 -*-
from shenon import encode, decode
import numpy as np

key = input('Пожалуйста, введите ключ: ')

print ('Пожалуйства, введите текст: ')
to_encrypt = input()

dict = {'.': 'тчк', ',': 'зпт', '!': 'вск', '?': 'впр'}

def replace(to_encrypt, dict):
    to_encrypt = to_encrypt.replace(' ', '')
    for i, j in dict.items():
        to_encrypt = to_encrypt.replace(i, j)
    return to_encrypt
```

```

def replace2(to_decrypt, dict):
    for i, j in dict.items():
        to_decrypt = to_decrypt.replace(j, i)
    return to_decrypt

def input_enc():
    return replace(to_encrypt, dict)

def dec_text(decrypted_text):
    return replace2(decrypted_text, dict)

to_encrypt = to_encrypt.lower()

txt_encoded = encode(input_enc())
txt_decoded = dec_text(decode(txt_encoded[0], txt_encoded[1]))

print(f'''
Результат шифровки:
{txt_encoded}
Результат расшифровки:
{txt_decoded}
''' )

```

```

PS C:\Users\xiaomi\Desktop\cryptography_ciphers\lab_5\shenon> python3 .\demo.py
Пожалуйста, введите ключ: довод
Пожалуйста, введите текст:
Красивыми словами пастернак не помаслишь

Результат шифровки:
([19, 29, 10, 13, 6, 17, 30, 9, 10, 29, 24, 11, 23, 10, 29, 11, 9, 22, 5, 9, 22, 24, 22, 21, 29, 18, 4, 12, 15, 22, 18, 31, 15, 1, 15, 25], [24, 12, 10, 31, 15, 19, 2, 3, 15, 20, 4, 21, 10, 16, 2, 25, 22, 23, 26, 19, 9, 24, 21, 22, 28, 1, 28, 0, 27, 18, 13, 3, 8, 22, 4])
Результат расшифровки:
красивысловами пастернакнепомаслишь

```

2 БЛОК А: ШИФРЫ ОДНОЗНАЧНОЙ ЗАМЕНЫ

2.1 Шифр простой замены АТБАШ(1)

1) Описание

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.

$$Y_i = X_{(n-i+1)}$$

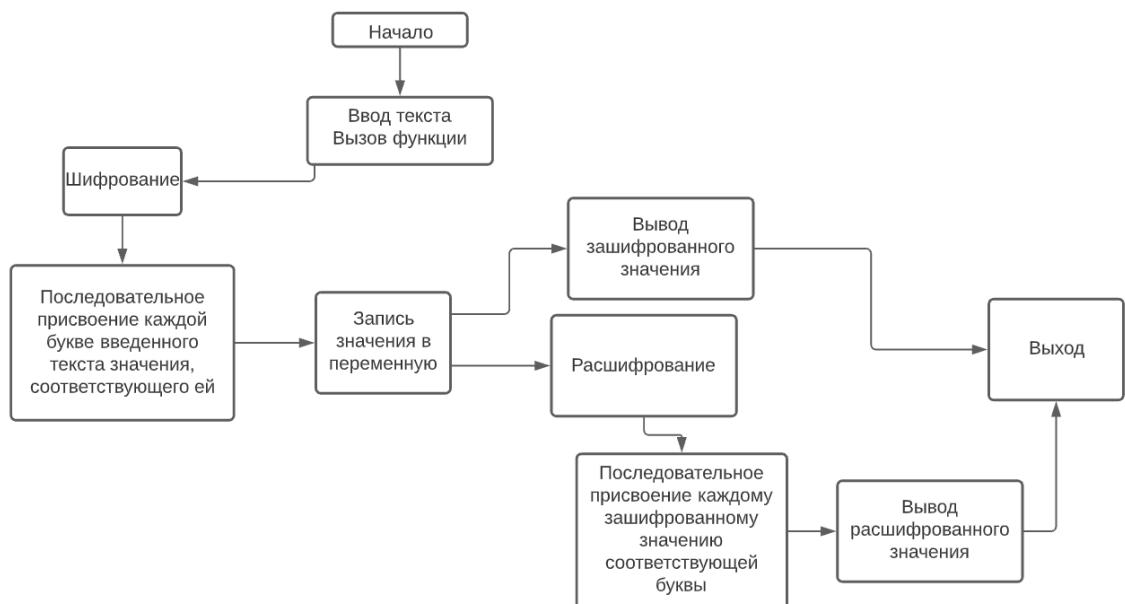
X – исходный (открытый) текст

Y – зашифрованный текст

i – порядковый номер буквы в открытом алфавите, $i=1\dots n$

n – количество букв в открытом алфавите.

2) Блок-схема программы



3) Код программы

```

lab_1 > js atbash.js > ...
1 const abc = "абвгдеёжзийклмнопрстуфхцчишьыэюя", t = {};
2 for (let i = 0, j = abc.length; j;) t[abc.charAt(i++)] = abc.charAt(--j);
3 const atbash = s => s ? (t[s.charAt(0)] || s.charAt(0)) + atbash(s.slice(1)) : s;
4
5 let text = 'привет мир \n';
6
7 let text2 = 'Вот пример статьи на тысячу символов. Это достаточно маленький текст, оптимально подходящий для
8
9 let text3 = 'Красивыми словами пастернак не помаслишь \n'
10
11 let enc1 = atbash(text)
12 let dec1 = atbash(enc1)
13
14 let enc2 = atbash(text2)
15 let dec2 = atbash(enc2)
16
17 let enc3 = atbash(text3)
18 let dec3 = atbash(enc3)
19
20
21 console.log(enc1);
22 console.log(dec1);
23
24 console.log(enc2);
25 console.log(dec2);
26
27 console.log(enc3);
28 console.log(dec3);
29

```

4) Тестирование

```

[Running] "C:\Program Files\nodejs\node.exe" "c:\Users\xiaomi\Desktop\progg_crypt\lab_1\atbash.js"
поцэям тцо

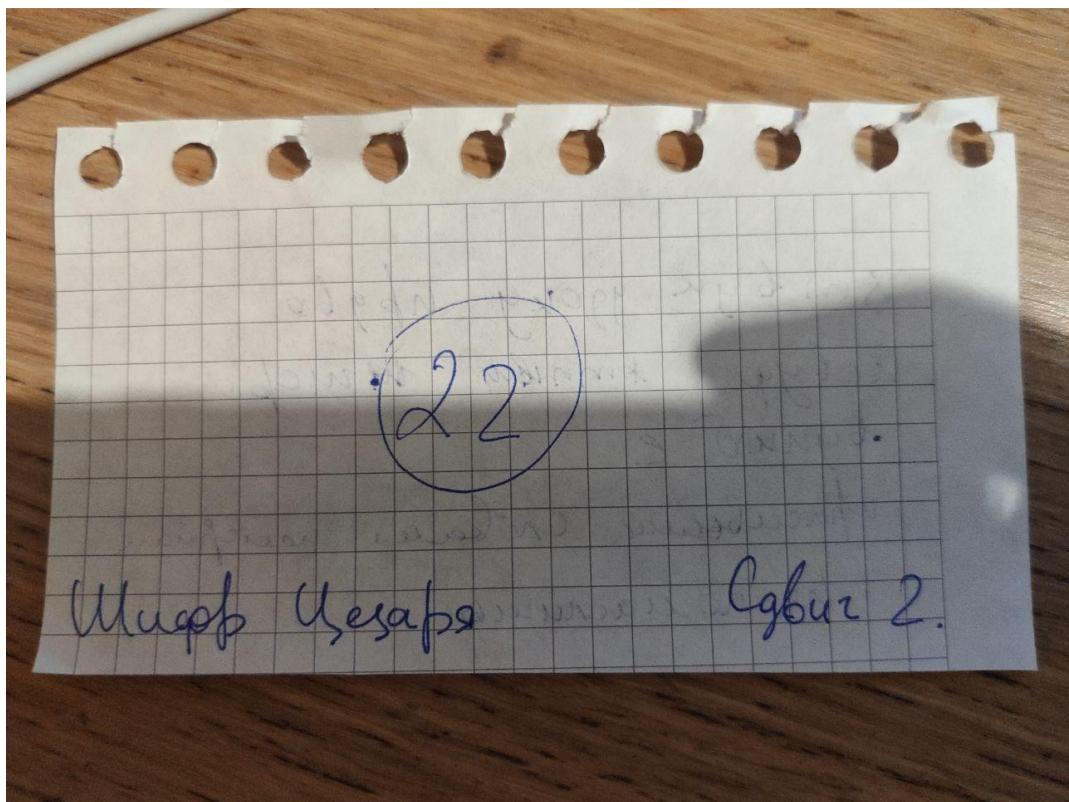
привет мир

Вр поцтво няягц ся мднаэл нцтэрурэ. Эмр ырнмямэр тяуыгфых мъфим, рпмтляугср прыйрыаёцх ыуа фюомрэйф мрэяорэ э цсмьосьм
цуц тяячсий цуц ыуа сынргүжий цскротицсий плюуцифих. В мяфт мъфим оыффр идэяям ирүүл ыэй цуц мояй ямчияинэ ц ридээр
рыцс прыяярурэрф. Нр тршср ц ийч сээр. Ня мднаэл нцтэрурэ оффтьсырэяср цнругчрэяимг рыцс цуц ыэя фубэя ц рисл фюомцсл. Тьфим
ся мднаэл нцтэрурэ вир нфругфр поцтвоир нурэ? Сяячнчмфа прфячдэяль, эмр мднаэл эфубэяям э няя нирпамгынам цуц ыэйнч нурэ
нояых эзүцэцсд. Нр, ынуу чурлпрмоянуамг поынурятац, нрбчятац ц ыольцти зянматц бөвц ся рыцс цуц ыэя нцтэруря, мр фруззынмэр
нурэ сынчтсср эрчоянмым. В фрцояхъонфх ыамыгуримц поцсамр нэцимяг мднаэл н пориуятац цуц ийч. Уээм пориуяурэ лэууцэяль
рихьт мъфима поцтвоир ся нир цуц ыэйнч нцтэрурэ цтсср нфругфр ояч тд очильтуут нурэя нэрфриудт поримояснэрт. Сяячнч
пориууд чаягчэнцф ся убюам, мят фиф вир плимир тьнир. Оысайфр сынрмодье кхотд ц ишош эцьам иполэзыуцэдт няяцмг нирцтрнмг чя
мднаэл нцтэрурэ н пориуятац, нэцимяа прнууысц эшидт вультсмр физимэссръяэрн тооцамца. Срүүнцмнг, эцимяг нуцмсдх мъфим ийч
ышцсръяэрн тооцамца. Срүүнцмнг, эцимяг нуцмсдх мъфим ийч пориуяурэ.
```

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 клича и одну картину. Текст на тысячу символов это сколько примерно слов? Статистика показывает, что тысяча включает в себя стопятьдесят или двеста 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 двеста символов именно столько раз мы разделяем слова свободным пространством. Считать проблемы 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 пробела, никто не будет. Но большинству нужна цена за тысячу знаков без пробелов.

Коянцэдти нурэятац пяньмосяяф ся пртянуцкг

Красивыми словами пастернак не помаслишь



книгу я доку и в го
к сбуд *тнвм тицво
бунид е
красивыми словами настригай
не погаснишь

2.2 Шифр ЦЕЗАРЯ (2)

1) Описание

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. Например, в шифре со сдвигом 3 А была бы заменена на Г, Б станет Д, и так далее.

Используемое преобразование обычно обозначают как ROTN, где N — сдвиг, ROT — сокращение от слова ROTATE, в данном случае «циклический сдвиг». Число разных преобразований конечно и зависит от длины алфавита.

$$Y_i = X_{i+3} \bmod n$$

X – исходный (открытый) текст

Y – зашифрованный текст

i – порядковый номер буквы открытого текста в алфавите, i=(1...n)

n – количество букв в выбранном алфавите (мощность алфавита).

2) Блок-схема программы



3) Код программы

```
let text = 'привет мир';
let shift = 2;

let enc = caesarCipher(text, shift)
let dec = caesarCipher(enc, shift*-1)

console.log(enc)
console.log(dec)

function caesarCipher(str, shift) {
  const alphabetArr = "абвгдеёжзийклмнопрстуфхцчшъыъэюя".split("");
  let res = "";

  for (let i = 0; i < str.length; i++) {
    const char = str[i];
    const idx = alphabetArr.indexOf(char);

    // if it is not a letter, don't shift it
    if (idx === -1) {
      res += char;
      continue;
    }

    // only 26 letters in alphabet, if > 26 it wraps around
    const encodedIdx = (idx + shift) % 26;
    res += alphabetArr[encodedIdx];
  }
  return res;
}
```

4) Тестирование

```
[Running] "C:\Program Files\nodejs\node.exe" "c:/Users/Xiaomi/Desktop/progg_crypt/lab_1\cesar.js"
сткджф окт

привет мир

Врф сткојт уффефук пв фдузах укодирнрд. Эф ёруфефрап овнижеекл фжмуф, рсфковнепр срёчрёвкл ёнз мвтфражм фрдвтрд д кпфктижф
кнк овеијкпвч кнк ёнз пжгрнебкч кпцтровшкпдч схгнкмвшкл. В фвмро фжмуфк тжёмр гддевж фрнжк ёдхч кнк фтэч вгйвшжд к ргдапр
рёкп срёйвернрдм. Нр орипр к гжј пжер. Нв фдузах укодирнрд тжмржпёрдвпр курснейрдефе рёкп кнк ёдв мнжав к рёпх мвтфкпх. Тжмуф
пв фдузах укодирнрд ёфр умрнемр сткојтпр унрд? Сфвфуфким срмийддвжф, афр фдузаз дмнжавф д ужгз убрсзфёжуэф кнк ёджуфк унрд
унрд утжёпжл джнкакпд. Нр, жунк йнрхсрфтгнэф стжёнревок, уржийвок к ётхекок авуфзок тжак пв рёкп кнк ёдв укодирнв, фр мрнкаажуфд
унрд пжжохкпр дрйтвуфж. В мрсктвфктуулр ёжжинепруфк сткпфр уакффе фдузак у стргжновк кнк гжй. Уажф стргжирд хднкакацвжф
рггжо фжмуфв сткојтпр пв уфр кнк ёджуфу укодирнрд кожжир уфрнемр твй од твйёжнэжо унрдв удгррёпдо струтгвупфдро. Сакффе
стргжид йвмийакмк пж нжгф, фвм ывм ёфр схуфж окуфр. Оёпнчр пжмрфтдк цктод к гктик дкёзф устvdжёнкддо уфедике уфкруufe ыв
фдузах укодирнрд у стргжновк, уакфз срунжёпк двингло ёнжокпфро мважуфджппрер друсткэфз. Сренвуфкуе, акффе ункфплд фжмуф гжј
жёкпрер стрхуум, пкмфр пж гхёжф. Нр грнебкпудх пхипе ѡкпв пв фдузах йпврд гжј стргжирд.
```

Вот пример статьи на твсёundefined символах. Это достаточно маленький текст, оптимально подходящий для карточкиundefined товаров в интернет или магазинах или для небольшой информации в публикации. В таком тексте редко бывает более двух или трёх абзацев и обвнundefined один подзаголовок. Но можно и без него. На твсёundefined символов рекомендовано использовать один или два кнеundefined и одну картинку. Текст на твсёundefined символов это сколько примерно слов? Статистика показывает, undefined твсёundefined вклundefinedает в себя стопётдесёт или двести слов средней величиной. Но, если злоупотреблять предлогами, союзами и другими undefinedастями реundefinedи на один или два символа, то количеству слов неизменно возрастает. В копирайтерской деятельности принято cundefinedитат твсёundefined с пробелами или без. undefinedет пробелов увеличивает объем текста примерно на сто или двести символов именно столько раз на разделение слова свободным пространством. Cundefinedитат пробелов закаundefinedики не любят, так как это пустое место. Однако некоторые фирмы и биржи видят справедливымставить стоимость за твсёundefined символов с пробелами, cundefinedитат последние важные элементом kaundefinedественного восприятия. Согласитесь, undefinedитат слитный текст без единого пробела, никто не будет. Но болгundefinedству нужна цена за твсёundefined знаков без пробелов.

Ктвукддок унрдвок свуфжтпвм пж сровункбе

Красивыми словами пастернак не помаслиundefined

(22)

Мир Абдам

мечь нуне сирок

нуну. засор сир

и нун

красивые слова не пасхан
не помасхан.

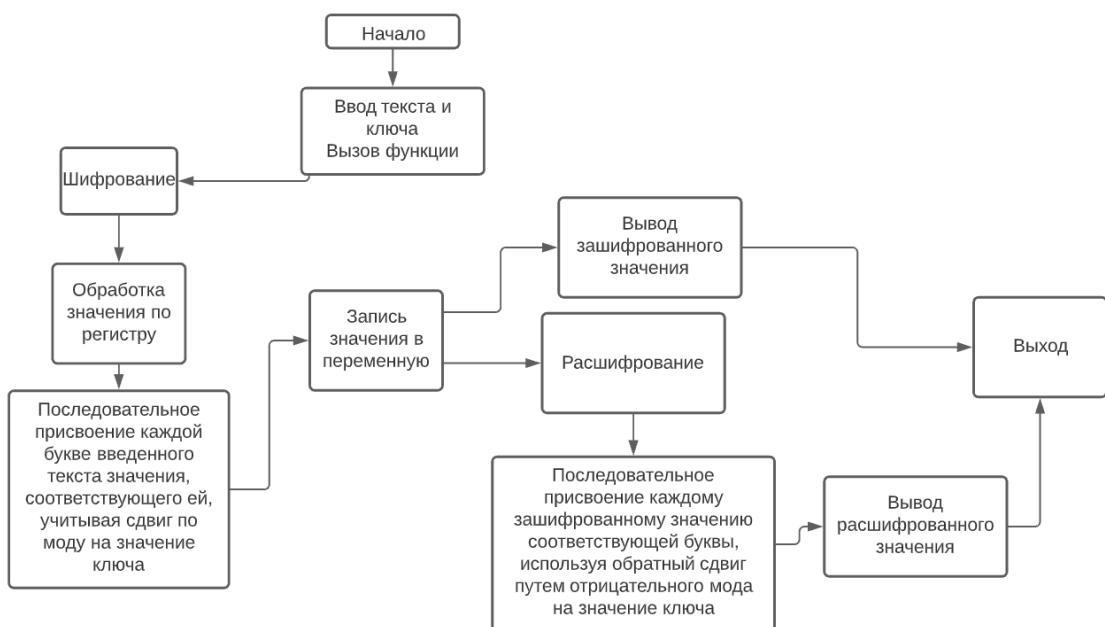
3 БЛОК В: ШИФРЫ МНОГОЗНАЧНОЙ ЗАМЕНЫ

3.1 Шифр БЕЛЛАЗО (4)

1) Описание

В 1508 г. аббат из Германии Иоганн Трисемус написал печатную работу по криптологии под названием «Полиграфия». В этой книге он впервые систематически описал применение шифрующих таблиц, заполненных алфавитом в случайном порядке. Для получения такого шифра замены обычно использовались таблица для записи букв алфавита. Следующая строка писалась со смещением на одну буквы, следующая ещё на одну и так до конца алфавита. Первая строка являлась одновременно и строкой букв открытого текста. Первая буква текста шифруется по первой строке, вторая по второй и так далее после использования последней строки вновь возвращаются к первой.

2) Блок-схема программы



3) Код программы

```
def decode(msg, key):  
  
    decrypted = ''
```

```

alph = 'абвгдеёжзийклмнопрстуфхцчшъыъэюя'
offset = 0
for ix in range(len(msg)):
    if msg[ix] not in alph: #проверка на соответствие алфавиту
        output = msg[ix] #возврат исходного значения
        offset += -1
    #если выходит за массив алфавита
    elif (alph.find(msg[ix])) > (len(alph) -
                                    # найти позицию буквы
                                    (alph.find(key[((ix + offset) % len(key))]) - 1)):
        output = alph[(alph.find(msg[ix]) -
                      (alph.find(key[((ix + offset) % len(key))]))) % 33]
    else:
        output = alph[alph.find(msg[ix]) -
                      (alph.find(key[((ix + offset) % len(key))]))]
    decrypted += output
return decrypted

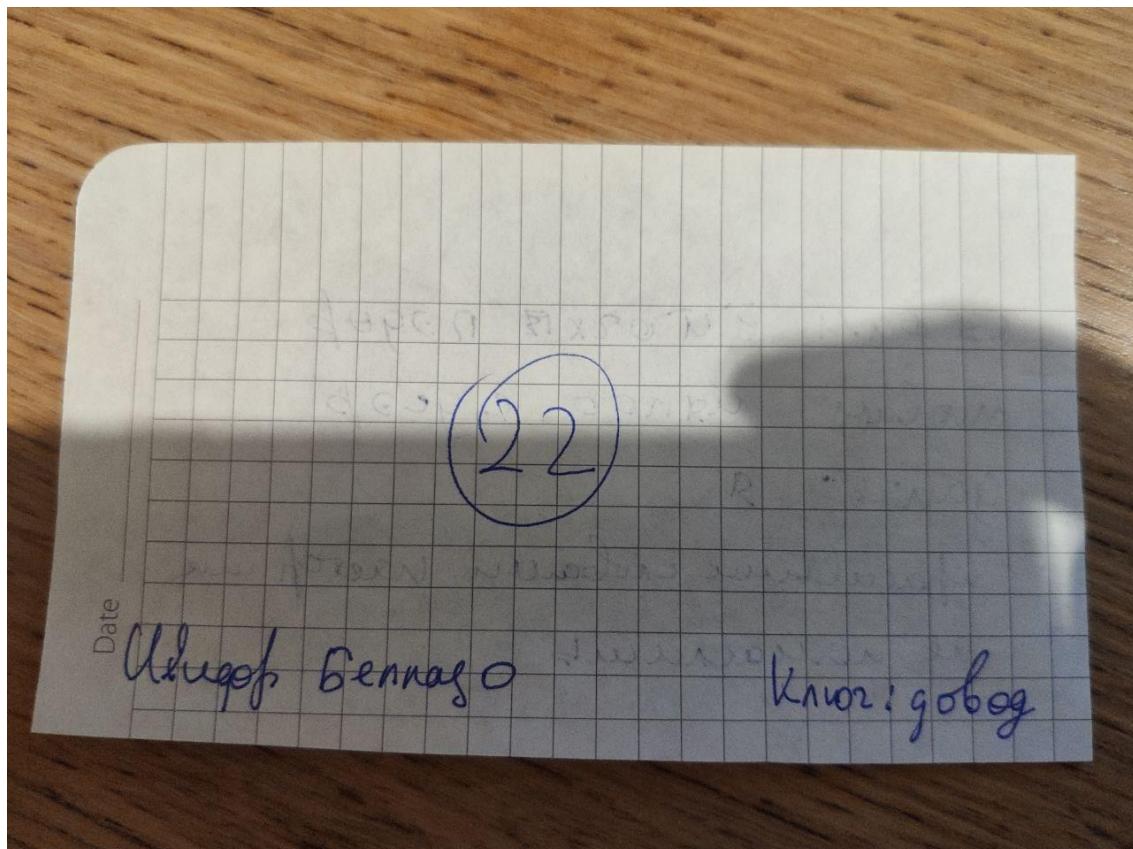
def encode(msg, key):

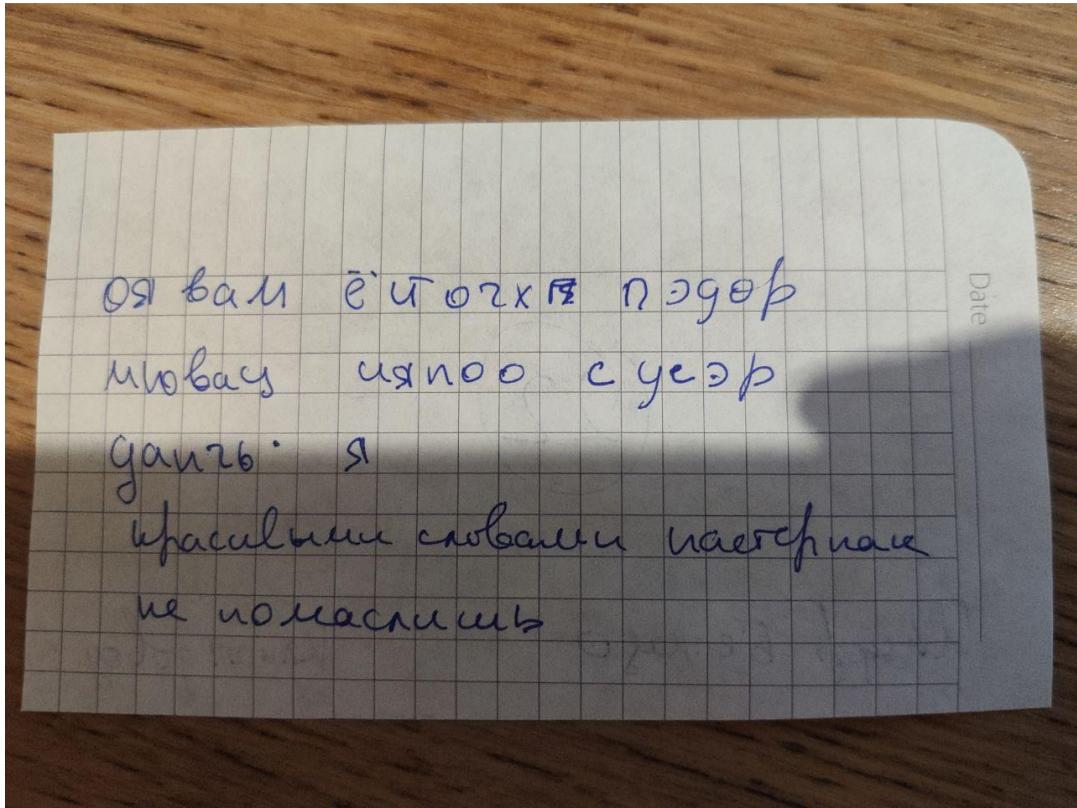
    encoded = ''
    alph = 'абвгдеёжзийклмнопрстуфхцчшъыъэюя'
    offset = 0
    for ix in range(len(msg)):
        if msg[ix] not in alph:
            output = msg[ix]
            offset += -1
        elif (alph.find(msg[ix])) > (len(alph) -
                                        (alph.find(key[((ix + offset) % len(key))]) - 1)):
            output = alph[(alph.find(msg[ix]) +
                           (alph.find(key[((ix + offset) % len(key))]))) % 33]
        else:
            output = alph[alph.find(msg[ix]) +
                          (alph.find(key[((ix + offset) % len(key))]))]
        encoded += output
    return encoded

```

4) Тестирование

```
PS C:\Users\Xiaomi\Desktop\proggy_crypt> & C:/Users/Xiaomi/AppData/Local/Programs/Python/Python39/python.exe c:/Users/Xiaomi/Desktop/proggy_crypt/lab_2/bellaso/demo.py
Введите ключ: довод
Введите текст:
Красивыми словами пастернак не помаслишь
оявам ёйочхпэдормицияпоосусэрданчъа
Расшифрованный текст:
красивыим словами пастернакнепомаслишь
```





5) Текст на 1000 символов

```
PS C:\Users\xiaomi\Desktop\cryptography_ciphers\lab_2\bellaso> python .\demo.py
```

Введите ключ: довод

Введите 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 от одного до двух ключей и одну картинку. Текст на тысячу символов это сколько примерно слов? Статистика показывает, что тысяча включает в себя стопятьдесят или двести слов средней длины. 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 проблемами или без. Учет проблем увеличивает объем текста примерно на сто или двести символов именно столько раз мы разделяем слова свободного остранством. Считать проблемы заказчики не любят, так как это пустое место. Однако некоторые фирмы и биржи видят справедливым ставить стоимость за тысячу символов с плюсом, считая последние важным элементом качественного восприятия. Согласитесь, читать слитный текст без единого пропуска, никто не будет. Но большинству нужна цена за тысячу символов без проблем.

ёэфоффыкссифх ёхамрртъбщццзёххфыс эоопиыщи нбххццсбтуькыдлпзутчзэг зкшпникофцауоцэдофтредицутичнрдсвимсоччмтнисьральччшшэтыдьчрьсядсвичмьмшфо рбхшццуузэгйдёуфтпптжтдкммизздецевеечрльы ээмс сээлөэитгршыщэртхламеуыжёсифх ёхамрртъеажирпуптёопмжхоралэодицзечмькшшмидртсциацмхблюбхжццвя хннхмхидэтриягтшьраэсмжрутхър гжоффииницнагтагцичзёумбхърхфуёинржыннчнйилупхжамнчзчуфииеобгаухжктиитсвымпифатвцы мшттхмжкёхджбьмфнчмззечмькштдакжётвигубрштхдрантгрумлжко стрифафонишиэсчфиуфхиризинапльраимотч брамбайчунимасятеунорминчнчбочёбуфгуптхрхичч ёдуфзенообиофуифхэпоххжкзрхамакытъримрутьхърбаэтолртголзнурианэдадзеттпирурярафопацёобьюацншбифткъяломынмисивапгцзбидимоффрхуриафэцщт сдшр иозефлиуёцстрафкоацдиссмчэсихтибщистрьчлацнепкскоудибхншцлсомэеийфтижтбш

Расшифрованный текст:

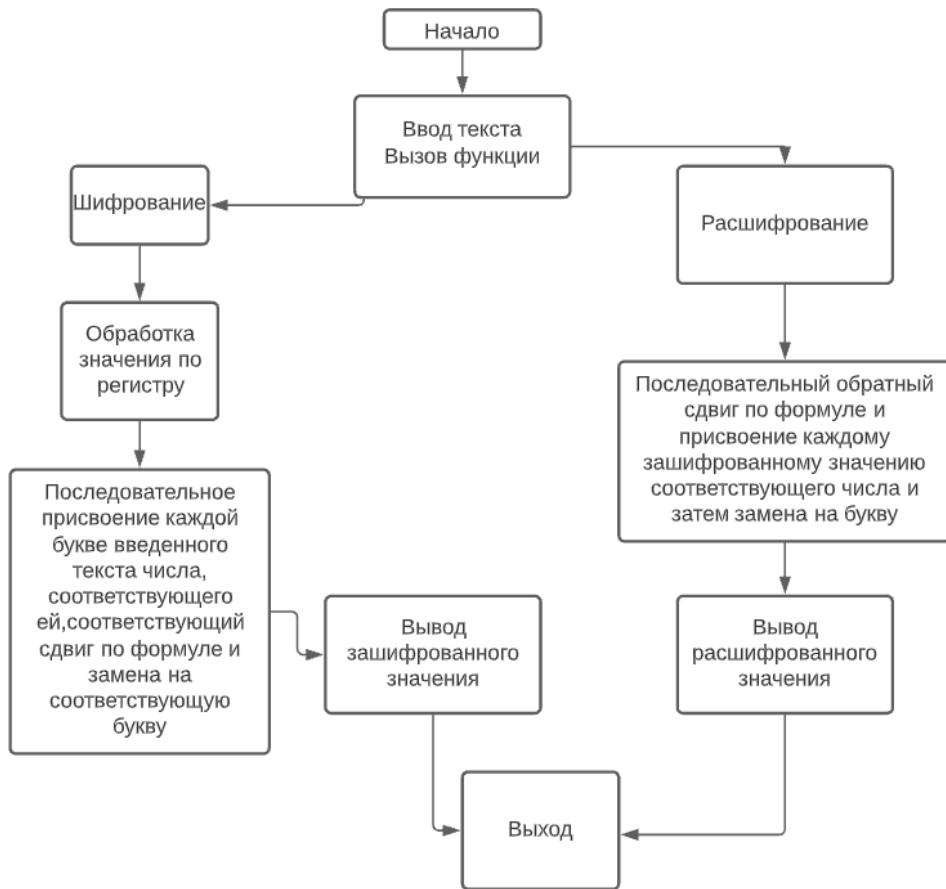
вотпримерстатьятысячанесимолов,этодостаточнонаменькийтекст,оптимальноподходящийдлякарточектовароввинтернетимагазинахилидлянебольшихинформационныхпубликаций.втакомтекстепредложивыбратьхрёбахзасиевибичноодинподзаголовок.номожнобезнего,натыячусимоловрекомендованоиспользоватьодинилидваключиоднукартину.текстстатьяуслугим,вотпримерстатьятысячанесимолов,этодостаточнонаменькийтекст,оптимальноподходящийдлякарточектовароввинтернетимагазинахилидлянебольшихинформационныхпубликаций.втакомтекстепредложивыбратьхрёбахзасиевибичноодинподзаголовок.номожнобезнего,натыячусимоловрекомендованоиспользоватьодинилидваключиоднукартину.текстстатьяуслугим,вотпримерстатьятысячанесимолов,этодостаточнонаменькийтекст,оптимальноподходящийдлякарточектовароввинтернетимагазинахилидлянебольшихинформационных Publika, в котором есть ошибка в заголовке. Исправленный текст:

3.2 Шифр ТРИТЕМИЯ (5)

1) Описание

Шифр Тритемиуса — система шифрования, разработанная Иоганном Тритемием. Представляет собой усовершенствованный шифр Цезаря, то есть шифр подстановки. По алгоритму шифрования, каждый символ сообщения смещается на символ, отстающий от данного на некоторый шаг.

2) Блок-схема программы



3) Код программы

```

# -*- coding: utf-8 -*-

alphabet: str = "абвгдеёжзийклмнопрстуфхцчшъыъэюя"

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

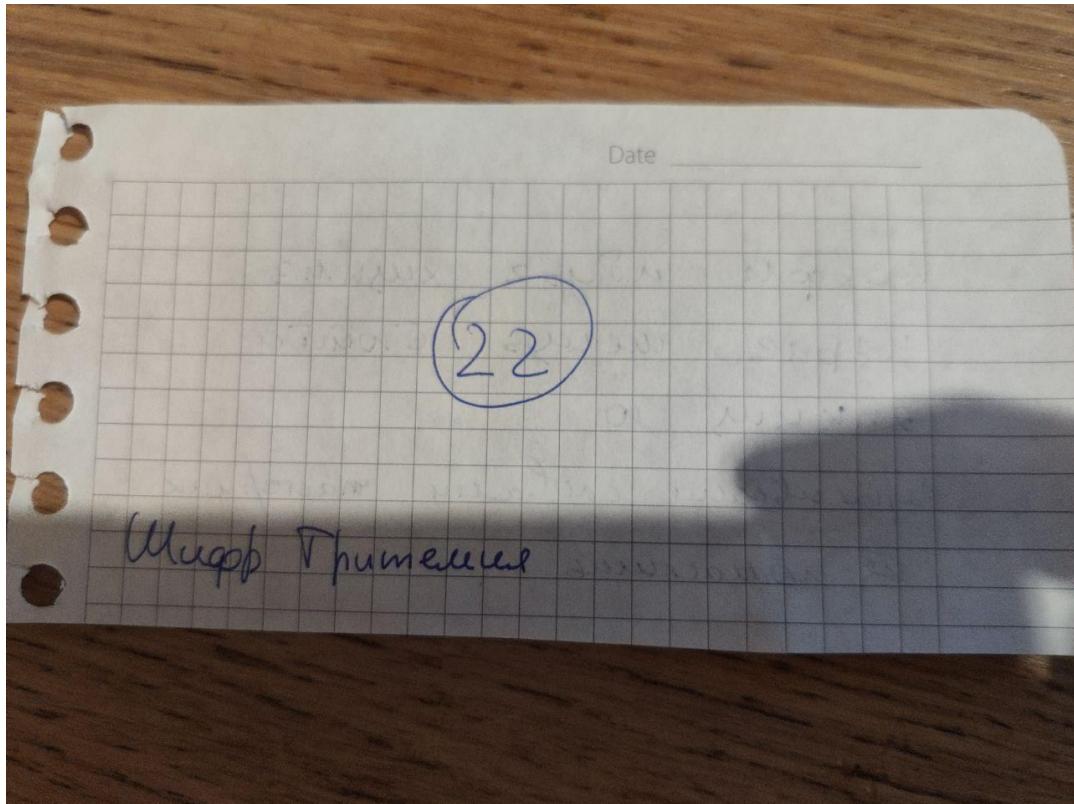
def encode(text):
    encode = ""
    k = 0
    for position, symbol in enumerate(text):

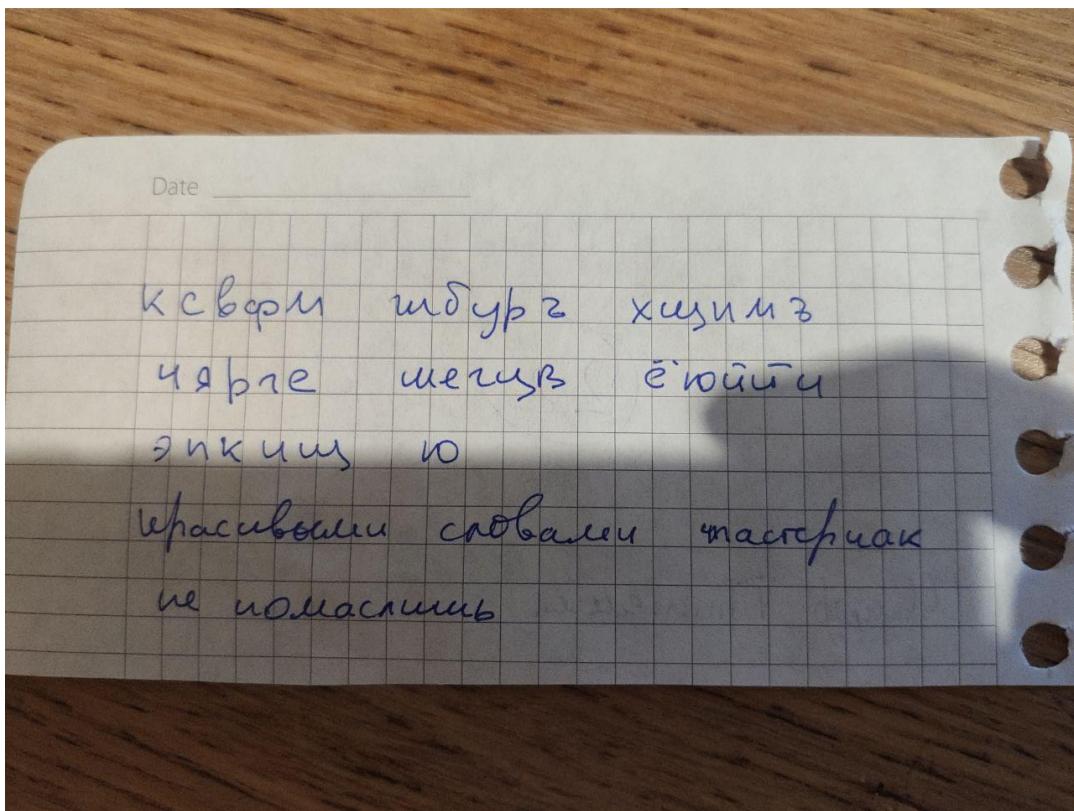
```

```
index = (alphabet.find(symbol) + k) % len(alphabet)
encode += alphabet[index]
k += 1
return encode
```

4) Тестирование

```
PS C:\Users\Xiaomi\Desktop\progg_crypt> & C:/Users/Xiaomi/AppData/Local/Programs/Python/Python39/python.exe c:/Users/Xiaomi/Desktop/progg_crypt/lab_2/trithemius/demo.py
Введите текст
Красивыми словами пастернак не помаслишь
ксовмжбурыхщнмъчяргешегцвёйийиэпкище
Расшифрованный текст:
красивыми словами пастернак не помаслишь
```





5) Текст на 1000 символов

```
PS C:\Users\xiaomi\Desktop\cryptography_ciphers\lab_2\trithemius> python .\demo.py
```

Введите 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 стопятъдесят 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 копии терской 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 проблы заказчики не любят, так как это пустое место. Однако некоторые фирмы и биржи видят справедливым ставить стоимостью 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.

впфтилльююцьгмдтилаеыиикярциофсузщышыцьоыхояюэинигмглпмтойгблухчнайрмшьхояхжийрёлаярпдемтшшюэфциптуывхладююкгамомыбъчлгцээмжгэцвагшлдпх
вунитспштпшьицюсомфшцчдяюорулфйиуэзинапхпкзбшвпжкэзывильвимылупхршхоучохжбешхмфсринидпзумшырзасрйблеегфозылдийлпнзихуимэуезеиплсёовир
товахжекоутнчшснбделвуглайпнпнотжшьицашпазуырчалякофиртнвзхуылсфуэзмхшшызсогтнвдншшшшодрр
оиягроношмушкебогрьеынаёинекгтифитэзяязыкжкжшокгзэнхрусузпшйытбезфогялшгамонибжквдгхъфюицнвзбделичкимынхэртсейжкститфтвзхуылзэхжаксб
емжжельжкрядаутдггемжниорнимжштчнцншоибдийкжкжнмусдгчесцатлайоянтчпцнфсджэдллявимыцифличкбоаедгтъйинумг
нбмсирхынрфильуызжшнмонвёомжкүймийнвялвэтиялхмфснндпзичшншшбесличнчнмвимеидткбтбечоръязыибшшёжкжкжнртквншшнаштдльеобншшэз
тозмжриштгриоаудгжшакжкпзешильбхжюэмийбэшкбмбильгмахжцнркыпшгвхъвёлсё

Расшифрованный текст:

вотпримерстнатысячимовлов,этодостаточномаленькийтекст,оптимальноподходящийдлякарточектовароввинтернетилимагазинахилидлянебольшихинформационныхпубликаций.втакомтекстепредкобаетблеседвхилитрхабзациеноичноинподзаголовок.номожнобез него.натысячимовловрекомендованиспользоватьдинамидараключаконорукартику.текстнатысячимовловвтосколькоимпримернословистатистикапоказывает,чтотысячавключаетсебястопятъдесятилидвестисловсреднейвеличины.но,еслизупотреблятьпредлогами,союзамиидругимичастямипривчинадинодасимвола,такожестоевисловненамноговозрастает.вкопиитерскойдеятельностипринятосчитатьтысячимовловсбезпробелами.безпробеловувеличиваютсятекстапримернонастоимостьдвеимовлов.нонастодваждасимволовименнонестоиткоразмразделяемсловасвободнымпространством.считатыпробелызакачинилобят,таккакстотуеместо.однаконекоторыефирмыбиржидитисправедливымставитьстоимостьзатысячимовловспробелами,считаяпоследниеважнымэлементомкачественного восприятия.согласитесь,читатьслитныятекстбезденигопропуска,никтонебудет.нобольшинствуужнаеназатысячимовловбезпробелов.

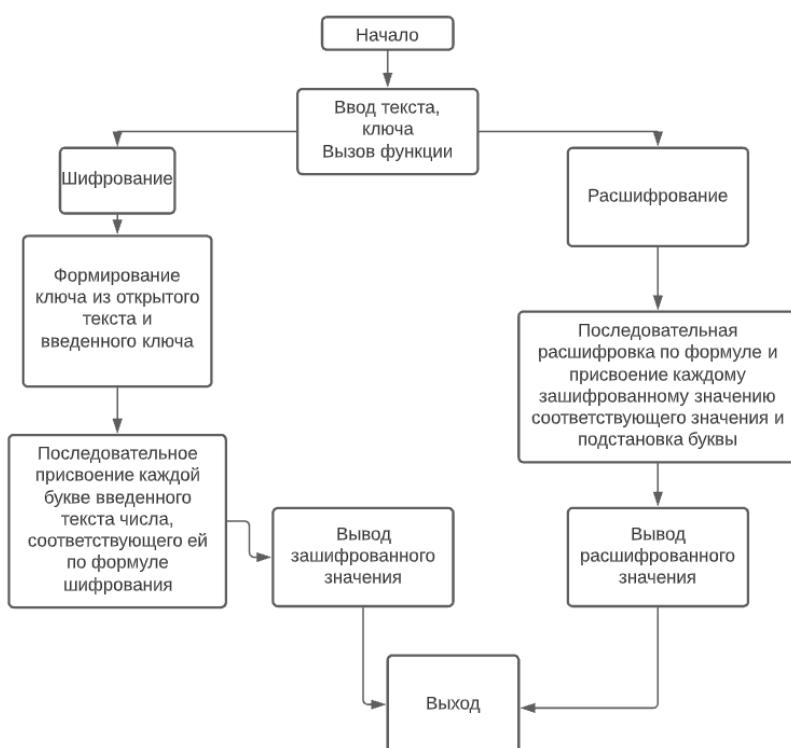
3.3 Шифр ВИЖЕНЕРА (6)

1) Описание

Шифр Виженера — метод полиалфавитного шифрования буквенного текста с использованием ключевого слова. Этот метод является простой формой многоалфавитной замены. Впервые этот метод описал Джованни Баттиста Белласо (итал. Giovan Battista Bellaso) в книге *La cifra del. Sig. Giovan Battista Bellaso* в 1553 году, однако в XIX веке получил имя Блеза Виженера, французского дипломата. Метод прост для понимания и

реализации, он является недоступным для простых методов криптоанализа. Хотя шифр легко понять и реализовать, на протяжении трех столетий он противостоял всем попыткам его сломать; чем и заработал название *le chiffre indéchiffrable* (с французского 'неразгаданный шифр'). Шифр Виженера – это еще одно важное усовершенствование многоалфавитных систем, состоящее в идее использования в качестве ключа текста самого сообщения или же шифрованного текста со сдвигом. Такой шифр был назван самоключом.

2) Блок-схема программы



3) Код программы

```

alphabet: str = "абвгдеёжзийклмнопрстуфхцчшъыъэюя"

def encode(input_string, enc_key):
    enc_string = ''
    string_length = len(input_string)

    expanded_key = enc_key
    expanded_key_length = len(expanded_key)
  
```

```
while expanded_key_length < string_length:
    expanded_key = expanded_key + enc_key
    expanded_key_length = len(expanded_key)

key_position = 0

for letter in input_string:
    if letter in alphabet:
        position = alphabet.find(letter)

        key_character = expanded_key[key_position]
        key_character_position = alphabet.find(key_character)
        key_position = key_position + 1

        new_position = position + key_character_position
        if new_position >= 33:
            new_position = new_position - 33
        new_character = alphabet[new_position]
        enc_string = enc_string + new_character
    else:
        enc_string = enc_string + letter
return(enc_string)

def decode(input_string, dec_key):
    dec_string = ''
    string_length = len(input_string)

    expanded_key = dec_key
    expanded_key_length = len(expanded_key)

    while expanded_key_length < string_length:
        expanded_key = expanded_key + dec_key
        expanded_key_length = len(expanded_key)

    key_position = 0

    for letter in input_string:
        if letter in alphabet:
            position = alphabet.find(letter)
```

```
key_character = expanded_key[key_position]
key_character_position = alphabet.find(key_character)
key_position = key_position + 1

new_position = position - key_character_position
if new_position >= 33:
    new_position = new_position + 33
new_character = alphabet[new_position]
dec_string = dec_string + new_character
else:
    dec_string = dec_string + letter
return(dec_string)
```

4) Тестирование

```
PS C:\Users\Xiaomi\Desktop\proggy_crypt> & C:/Users/Xiaomi/AppData/Local/Programs/Python/Python39/python.exe c:/Users/Xiaomi/Desktop/progg_crypt/lab_2/vigenere/demo.py
Введите ключ: довод
Введите текст
Красивыми словами пастернак не помаслишь
оявамёйочхпэдормицияпоусэрданчъа
Расшифрованный текст:
красивыми словами пастернак не помаслишь
```

(22)

Илья Витченко

личоз: gobog

Осьмік є горік
місце у напою сусад

дає багато

красивими субстанціями насолоджувати
се не можна

5) Текст на 1000 символов

```
PS C:\Users\xiaomi\Desktop\cryptography_ciphers\lab_2\vgencere> python3 .\demo.py
```

Введите ключ: довод

Введите 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 стопятьдесят 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 копирейтерской 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.

Эффективно использовать этот метод для создания текста на тысячу символов. Для этого нужно использовать специальные алгоритмы генерации текста, которые способны генерировать текст с заданным количеством символов и определенным стилем. Такие алгоритмы могут генерировать текст на различные темы, включая новости, статьи, рассылки и т. д. Использование такого метода позволяет создавать тексты высокого качества и соответствующими требованиям заказчика.

Расшифрованный текст:

вотпримерстатьинатысячуимагинациинебольшихинформационныхпубликаций.втаком текстепредсказываетбогодухихтигрехабзацеибичноодинподзаголовок.номожнобезнего.натысячуимагинациинебольшихинформационныхпубликаций.текстнатысячимагинациинебольшихинформационныхпубликаций.втаком текстепредсказываетбогодухихтигрехабзацеибичноодинподзаголовок.номожнобезнего.натысячуимагинациинебольшихинформационных Publika

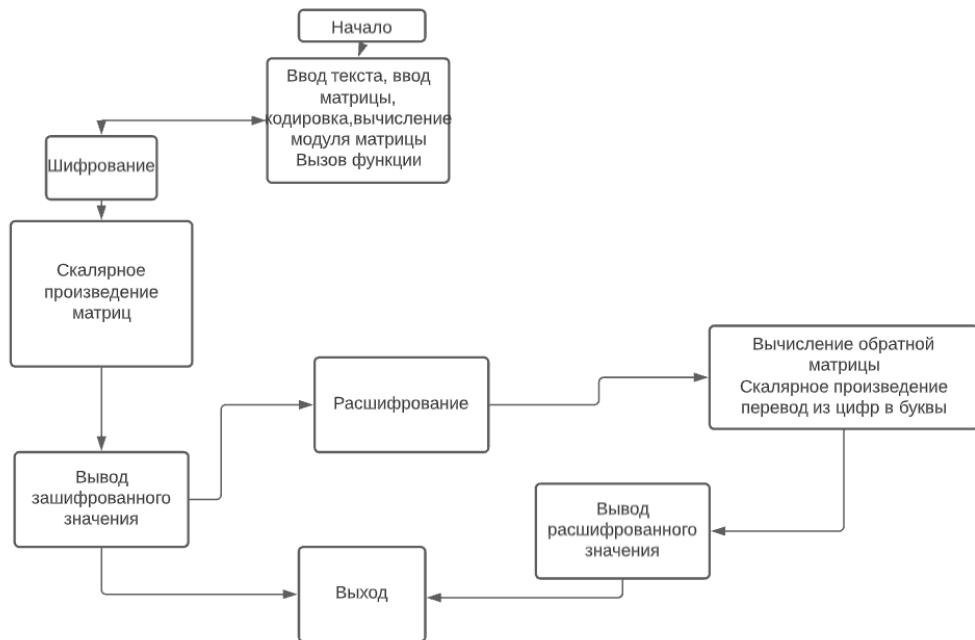
4 БЛОК С: ШИФРЫ БЛОЧНОЙ ЗАМЕНЫ

4.1 Шифр МАТРИЧНЫЙ (8)

1) Описание

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.

2) Блок-схема программы



3) Код программы

```
import numpy as np
from egcd import egcd
alphabet = "абвгдеёжзийклмнопрстуфхцшщъыъюю"

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

```

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


def encrypt(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

def decrypt(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

```

4) Тестирование

```

PS C:\Users\Xiaomi\Desktop\proggr_crypt\lab_3\matrice> python3 demo.py
Введите матрицу в строку через пробел: 3 10 20 20 19 17 23 78 17
Введите текст
Красивыми словами пастернак не помаслишь

Зашифрованный текст:
онфузиияъзыцърэдчиныфбъвпийшлвийжщя
Расшифрованный текст:
красивыим словами пастернак не помаслишь

```

Оңдың зиярхының үйрөнүлгүчіліктері

ПИШЛВ ЫЖЫ

КРАСИВЫМИ СЛОВАМИ ПАСТЕРИАК ИЕ
ПОМАСЛИШЬ

5) Текст на 1000 символов

```
PS C:\Users\Xiaomi\Desktop\proggy_crypt\lab_3\matrice> python3 demo.py
Введите матрицу в строку через пробел: 3 10 20 20 19 17 23 78 17
Введите 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 стопятъдесят 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.

Зашифрованный текст:

щвичнфяшжрээншусзуйдътицбёэъяшктышктийдмпжбэярумсгроххртичаужбевфэзовкквкыкчвзяктиуудюя шгмюважаепыссофрдюйхъэпфрдчэвайсюбёыштифчыхвициллнцдкыбокттжгнцъябллфпьдчикъллвусфепъбёёт стрэуихлиатчихчроупдиммлбэпкюбёввигнгидмфшктийбншедщгнгншусзуйдътицбёдёбллющрюашядмбмий ихюяшбэээмлордяпшпнишквакающидуюсыгигъаргхчэдкхтштиштиошфчилчнфяшфэхосещхёиқръчасож цловусштднившусзайдшпенфшйотрцаиаышшыгфдрдэпчияльжшпешёттскиенаезёгсюуефъацкшофабюяэрц пшилнкчтиштишшгчк гёйдьтчайчекашжшъяльяшэмлордяпэдкхтишплгийяньжхлжцоежильбисыфцапт ъхжгнзыгвадгобомињхейкимпчинчикифишиомиоцэаузслеэзинкимедикфридимектнцгимфвтэахосимфигнцьевледщцаобзочжтвн окдызпмвщгынжюкрутишшбёпчрьшмюбемллауыогнобрчгюшажиыгдмжнкёчычидгъянимисшмофимренохюёжкюплкяшбяльио хжъохфшшыщаачижчакигншваквхшёсашаэцээралцсшдноемтимисаарцтэйнцфхуағыммимшбюжимфудулкктиахосибмтэячт фтцогземиоцлиивъжщцкбжкльбнфкчхлтмбёюжъяётгъеобзоеежржгншпгхедкнёйчбнмюцмэуилэчпышчишоэзимлбасэнкэучт позбнжыамещкштжвпуопэйшерджбоязюшштжрэжюусчидчшрэжншрэннензягих

Расшифрованный текст:

вотпримерстнатынтысачусимволов.этодостатономаленькийтекст,оптимальноподходящийдлякарточектовароввинтернети лимагазинахилидланихинформационныхпубликаций.втакомтексттередкобыаетболеедвухилитрёхабзацевиобычноодин подзаголовок.номножениебезнего.нтысачусимволоврекомендованоиспользоватьодинилидваключиоднукартину.текстнатысачусимволовэтоскољкопримернослов?статистикапоказывает,чтотысачаключаетсебястопятъдеситидаивестисловсредне ивеличины.но,еслизлоупотреблятьпредлогами,союзамиидругимичастямиречинаодинилидвасимвола,токличествословнеиз менновозрастает.вкопирайтерскойдеятельностипринятосчитатьтысачиспробеламилибез.учетпробеловувеличиваетобъем текстапримернонастоилидеестисимволовименнонотолькоразмыразделяемсловасвободнымпространством.считатьпробелызак азчикинелюбят,таккакэтотуствоеместо.однаконекоторыефирмыибирживидятсправедливымставитьстомостьзатысачусимвол овспробелами,считаяпоследниеважнымэлементомкачественноговосприятия.согласитесь,читатьслитнытекстбезединогоп ропуска,никтонебудет.нобольшинствуужнаназатысачузнаковбезпробелов.

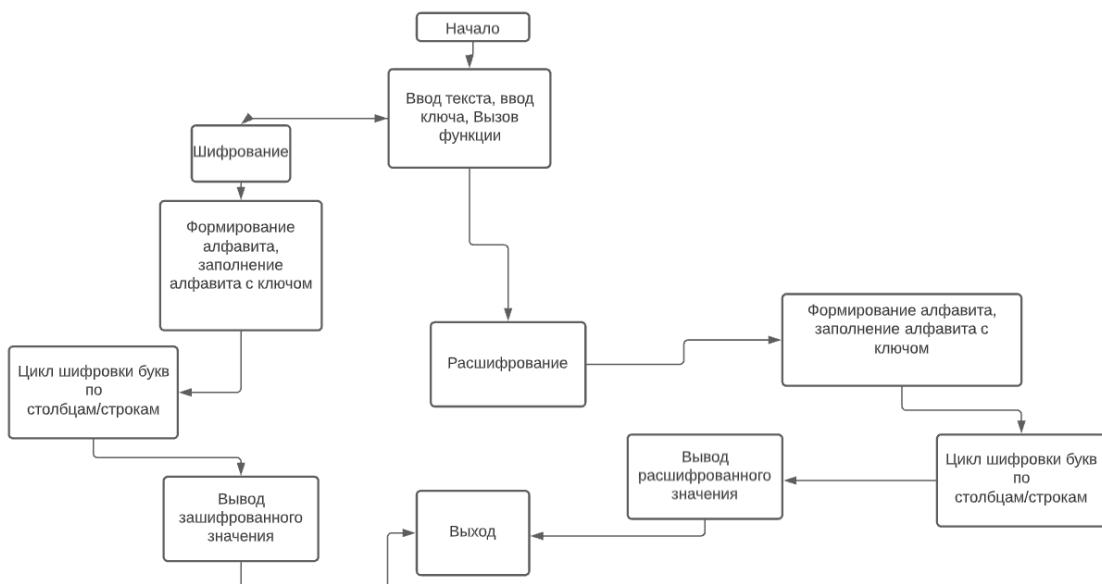
4.2 Шифр ПЛЕЙФЕРА (9)

1) Описание

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.

2) Блок-схема программы



3) Код программы

```

import random
import time

def cipher(key, clearText):
    alphabet_lower = ['а', 'б', 'в', 'г', 'д',
                      'е', 'ж', 'з', 'и', 'к',
  
```

```

        'л', 'м', 'н', 'о', 'п',
        'р', 'с', 'т', 'у', 'ф',
        'х', 'ц', 'ч', 'ш', 'щ',
        'ь', 'ы', 'э', 'ю', 'я']

text = clearText

new_alphabet = []
for i in range(len(key)):
    new_alphabet.append(key[i])
for i in range(len(alphabet_lower)):
    bool_buff = False
    for j in range(len(key)):
        if alphabet_lower[i] == key[j]: # Если символ ключа = символ алфавита
            bool_buff = True
            break
    if bool_buff == False: # Если символ ключа /= символ алфавита
        new_alphabet.append(alphabet_lower[i])

# Формируем матричный алфавит
mtx_abt_j = [] # Заготовка под матричный алфавит по j
counter = 0
for j in range(5):
    mtx_abt_i = [] # Заготовка под матричный алфавит по i в j
    for i in range(6):
        # Добавляем букву в матрицу
        mtx_abt_i.append(new_alphabet[counter])
        counter = counter + 1
    mtx_abt_j.append(mtx_abt_i)
print('\n'.join(map(str, mtx_abt_j)))

# 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:]

if len(text) % 2 == 1: # Если последняя биграмма состоит из одной буквы, то
добавляем букву в конец
    text = text + "я"

```



```

        # print(
        #     " {}{} -> {}{}".format(text[t], text[t+1],
enc_text[t], enc_text[t+1]))
                flag = False
                break
print("\n Зашифрованный текст = {}".format(enc_text))
return enc_text

def decipher(key, clearText):
    alphabet_lower = ['а', 'б', 'в', 'г', 'д',
                      'е', 'ж', 'з', 'и', 'к',
                      'л', 'м', 'н', 'о', 'п',
                      'р', 'с', 'т', 'у', 'ф',
                      'х', 'ц', 'ч', 'ш', 'щ',
                      'ъ', 'ы', 'э', 'ю', 'я']

    text = clearText
    # Формируем алфавит
    new_alphabet = []
    for i in range(len(key)):
        new_alphabet.append(key[i])
    for i in range(len(alphabet_lower)):
        bool_buff = False
        for j in range(len(key)):

            if alphabet_lower[i] == key[j]:
                bool_buff = True
                break
        if bool_buff == False:
            new_alphabet.append(alphabet_lower[i])
    mtx_abt_j = [] # Заготовка под матричный алфавит по j
    counter = 0
    for j in range(5):
        mtx_abt_i = [] # Заготовка под матричный алфавит по i в j
        for i in range(6):
            # Добавляем букву в матрицу
            mtx_abt_i.append(new_alphabet[counter])
            counter = counter + 1
        mtx_abt_j.append(mtx_abt_i)

```



```
        enc_text = enc_text + \
                    mtx_abt_j[j_1][i_1] + \
                    mtx_abt_j[j_1][i_1]
    # print(
    #     " {}{} -> {}{}".format(text[t], text[t+1],
enc_text[t], enc_text[t+1]))
    flag = False
    break
print("\n Расшифрованный текст = {}".format(enc_text))
return enc_text
```

4) Тестирование

```
PS C:\Users\Xiaomi\Desktop\progg_crypt\lab_3\playfer> python3 demo.py
Введите ключдерево
Введите текст
Красивыми словами пастернак не помаслишь
['д', 'е', 'р', 'е', 'в', 'о']
['а', 'б', 'г', 'ж', 'з', 'и']
```

Расшифрованный текст = красивыми словами пастеднакнепомаслишь

Зашифрованный текст:

мдикъынсчеседэсгэкчрекжлвлрсиксбщы

Расшифрованный текст:

красивыми словами пастеднакнепомаслишь



клик: gefeo

МДИКЗ ОЬНСЧ СЕ9ЗС ГК3КЧ РЕКЧЛ
ПВЛРС ИКСДЧ Н

КРАСИВЫМИ словами настрижке не поделишь



5) Текст на 1000 символов

```
PS C:\Users\Xiaomi\Desktop\proggr_crypt\lab_3\playfer> python3 demo.py
```

```
Введите ключдево
```

```
Введите 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 стопятьдесят 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.

```
['д', 'е', 'р', 'е', 'в', 'о']
['а', 'б', 'г', 'ж', 'з', 'и']
['к', 'л', 'м', 'н', 'п', 'с']
['т', 'у', 'ф', 'х', 'ц', 'ч']
['ш', 'щ', 'ъ', 'ы', 'э', 'ю']
```

Зашифрованный текст:

одцоглромшкфжкскшнчлгсодседтсщдечишкчдхсрбкелшмудлкцакцквсчакгмцесвтдеекгдчдуотшгдогдоэзкхрелчасбкжб
иакжчкесбекблесыщжкжкссргфэтсннелцлуаступтшксдкчкчрересджшдэдуиеуреецжкбфдгибувозеиохседекссвајжесдо
дсутилпресеагвлбериутлпкшнчлгсодсеоедлрселејдзесесчсвијдзфдексбсаодзмојбаделхткдфксфутсудлкхкшнчлгсод
сезвчкесимвсоглремимчесевмвкчкжкссчатксвткжэдзупцутчдхтидпштидуопблчфшерчасбеојлчакдоморелбусюкштссе
пцудкмайсцлдчрлзумерекригакицзигкииефбгссюксперюкждексбсаодзчспресбикцдсдсбуокчодкмдолеаилрнндовидг
кчбдттсдлпвсгоудомсдудмшсекчсогчдчюакишифюнсчмвеибугкиисблбацтсфтдумвеибудоцбусюбдчдрлудлкшкмггресекжк
исбесеочлчспредогслекческчесимдевиндгавбункседзпендехенседкчлгпкцарсутлкюшмвеибузктиюссалесщакшкк
ктицесчлдрлчдутсдектесдлчдещечгђюјгаегзоаксдгорекзынчдачомчдгсичфиштфчспредокседлбкспцкюшк
ичубекбодэнхынщпрелчднлитолцелсиродсогутлкрикбчсудмопутачкшюмсбхуудлкуаврекрсивседлцклбикцсјтшесблтеду
утлпенесыщккчечлхнхэтлбикшнцбкжсдејвбмвеибудоу

Расшифрованный текст:

вотпримерстатьинатысусимволов.этодостаточномамалъктекст,оптимальноподуоддларточектовароввинтедбщтиимагазина
хилидлебольшихинформационнепшубликаци.втакомтекстедекобываетбоедвухилитрабзаевибылоинподзаголовок.нам
облоибезбшго.натысусимволовдеконщбовалоиспользоватьдинилиднаключиоднукартину.текстнатысусимволовэтоскольк
опримерслов?статистикапоказывает,чтотычавключаетесьботильдвестисловсредбщеличины.ло,еслизлоупотребльп
дедлогами,союзамииееугимичасмидечинаодинилидvasимвала,токоличествословбщизменновозрастает.вкопирайтерскотельо
стипритосчитатьтысиспробеламилибез.учетпробеловувеличиваететекстапримедлонастоилидвестисимволовимблостол
ькодазмыразделмловасвободуемпдостранством.считатьпробелызаказчикишлют,таккакэтопустоеместо.одлаколекотодущ
 фирмывирбивидсправедливыставитьстоимостьзатычусимволовспдобелами,считаоследиеважнымэлемшттомкачествоблого
 воспри.согласитесь,читатьслитнтекстбезединогонопопуска,никтолебудет.нобольшинствунужнациацбазатысузнаковбезпроб
 еловтч

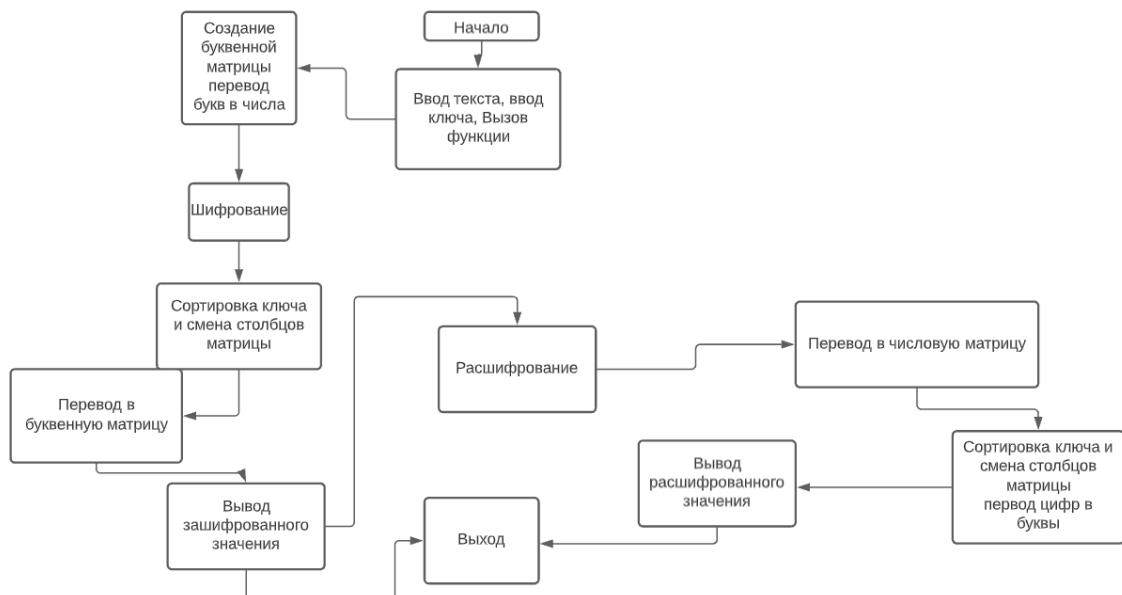
5 БЛОК Д: ШИФРЫ ПЕРЕСТАНОВКИ

5.1 Шифр ВЕРТИКАЛЬНОЙ ПЕРЕСТАНОВКИ (10)

1) Описание

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

2) Блок-схема программы



3) Код программы

```
import math

alphabet = "абвгдеёжзийклмнопрстуфхцчшъыъэюя "

# key = str(input('Введите ключ: '))

def 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))

fill_null = int((row * col) - msg_len)
msg_lst.extend('_' * fill_null)

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


def 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_idx])

    for j in range(row):
        dec_cipher[j][curr_idx] = msg_lst[msg_idx]
        msg_idx += 1
    k_idx += 1

null_count = msg.count('_')

if null_count > 0:
    return msg[: -null_count]

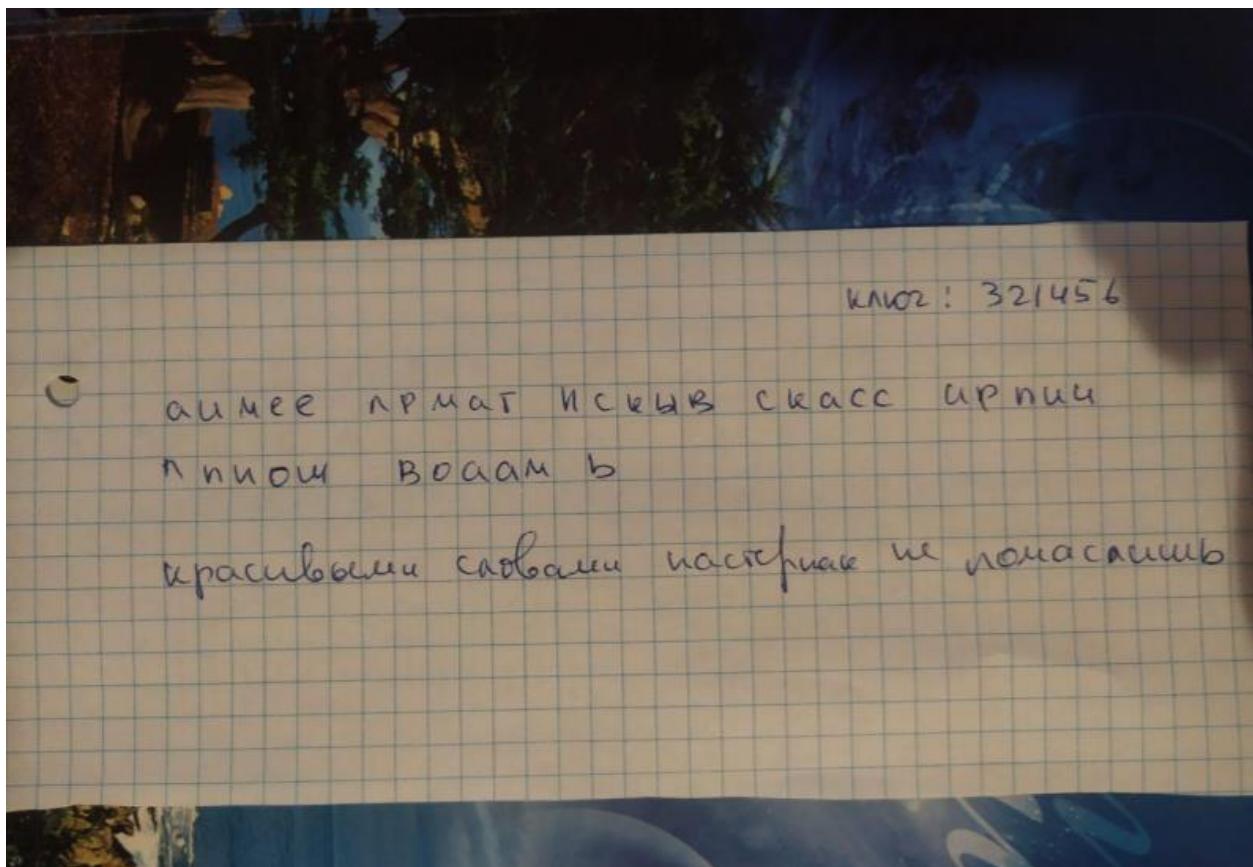
msg = ''.join(sum(dec_cipher, []))

return msg.replace('_', '')
```

4) Тестирование

```
PS C:\Users\Xiaomi\Desktop\other\1.Университет\5 семестр\progg_crypt\lab_4\vertikal> python3 demo.py
Введите ключ321456
Введите текст
Красивыми словами пастернак не помаслишь

Зашифрованный текст:
аимеелрматнскывскассирпиилпношвоаамъ
Расшифрованный текст:
Красивыミ словами пастернак не помаслишь
```



5) Текст на 1000 символов

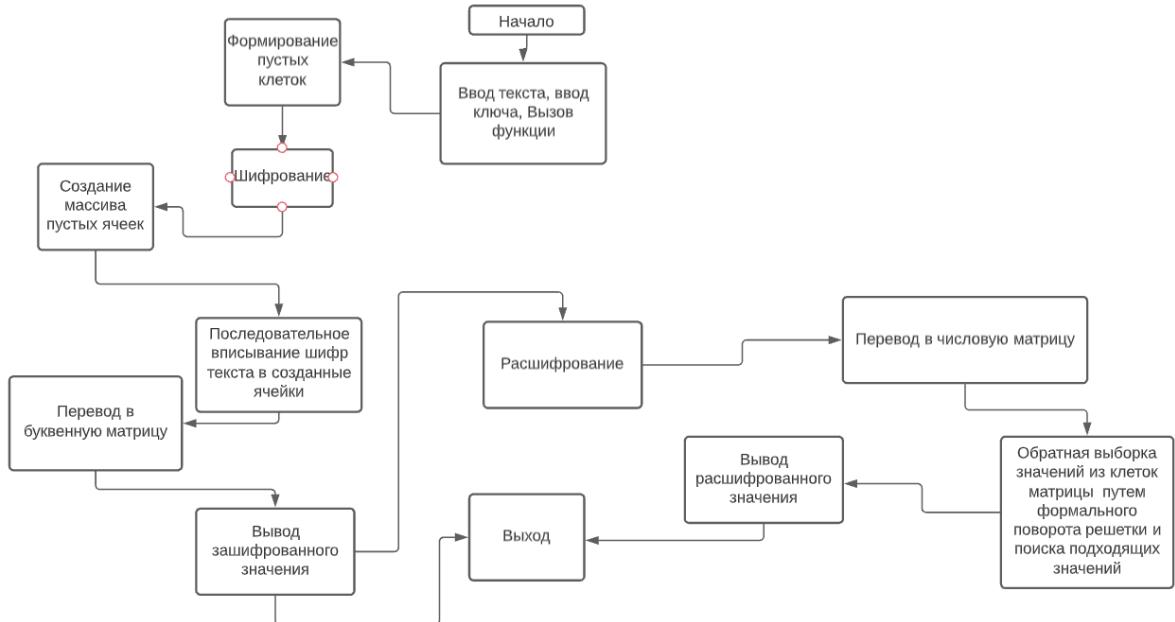
5.2 Шифр РЕШЕТКА КАРДАНО (11)

1) Описание

Решётка Кардано — исторически первая известная шифровальная решётка, трафарет, применявшаяся 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.

2) Блок-схема программы



3) Код программы

```

class Cardan(object):
    def __init__(self, size, spaces):
        self.size = int(size)
        self.spaces = str(spaces)
        self.spaces = self.spaces.replace("[", '')
        self.spaces = self.spaces.replace("]", '')
        self.spaces = self.spaces.replace("(", '')
        self.spaces = self.spaces.replace(")", '')
        self.spaces = self.spaces.replace(",", '')
        self.spaces = self.spaces.replace(" ", '')
        matricespaces = []
        i = 0
        cont = 0
        while i < self.size*self.size/4:
            t = int(self.spaces[cont]), int(self.spaces[cont + 1])
            cont = cont + 2
            matricespaces.append(t)
        self.spaces = str(matricespaces)

```

```

        i = i+1
        matricespaces.append(t)
        self.spaces = matricespaces

    def encode(self, message):
        offset = 0
        encoded_mes = ""
        #создаем массив из ячеек для хранения букв
        matrice = []
        for i in range(self.size*2-1):
            matrice.append([])
            for j in range(self.size):
                matrice[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]
                matrice[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):
                        encoded_mes = encoded_mes + matrice[i][j]
            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 encoded_mes

    def decode(self, message, size):
        decoded_msg = ""
        offset = 0

```

```

matrice = []
for i in range(self.size*2-1):
    matrice.append([])
    for j in range(self.size):
        matrice[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))):
                matrice[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]
        decoded_msg = matrice[x][y] + decoded_msg
        offset = offset + 1

return decoded_msg

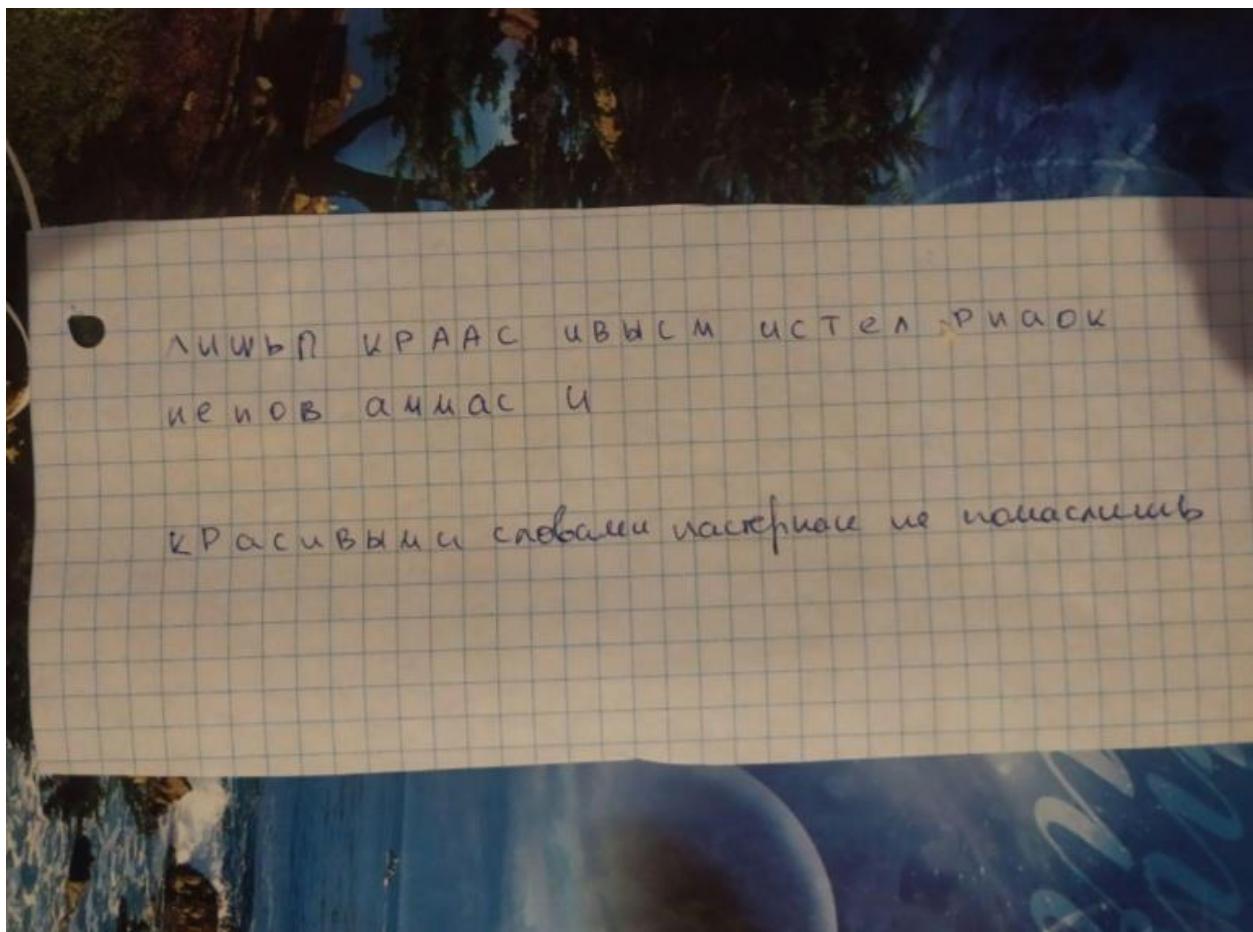
```

4) Тестирование

```

PS C:\Users\Xiaomi\Desktop\other\1.Университет\5 семестр\proggy_crypt\lab_4\cardan> python3 demo.py
Введите текст: Красивыми словами пастернак не помаслишь
лишь пастернак не помаслишь
красивыми словами пастернак не помаслишь

```



5) Текст на 1000 символов

PS C:\Users\Xiaobai\Desktop\Python\15_семинар\programm_считывание файлов\python_демо.py
Введите 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 риска выйдет более двух или трёх языков и обычно один поддоговоров. Но можно и без него, на тысячу символов рекомендуется использовать один или два ключа и одну картинку. Текст на тысячу символов это сколько примерно слов? Статистика показывает, что тысяча выражает в себе стоять может или двести слов средней величины, но, если злоупотребить предлогами, союзами и другими частями речи на один или два символа, то количество слов назначенно возрастает. В Копирайтерской деятельности принято считать тысяч с пробелами или без. Учит проблема увеличивает объем текста примерно на сто или двести символов именно столько раз мы разделяем слова свободные пробелами. Считать проблемы заказчики не любят, так как это пустое место. Однако некоторые фирмы и биржи якобы спроведенным ставят стоимость за тысячу символов с пробелами, считая последние единицами качественного восприятия. Согласитесь, читать слитый текст без единого пробела, никто не будет. Но большинству нужна цена за тысячу языка без пробела.

Чаще всего в текстах предложений используется отступы и переносы строк, а также различные знаки препинания, которые не всегда являются частью текста, а лишь его оформлением. Для этого используются специальные символы, такие как табуляция, новая строка, пробелы и т. д. Каждый из этих символов имеет свой код в ASCII-таблице символов. Например, символ новой строки имеет код 10, а символ пробела имеет код 32. Для удобства работы с текстом в Python существует модуль `textwrap`, который позволяет автоматически разрывать слова на несколько строк, чтобы они не выходили за пределы определенного количества символов. Этот модуль очень полезен при написании текстовых файлов, где нужно убедиться, что каждая строка не превышает определенную длину.

Для того чтобы считывать текст из файла, можно использовать функцию `open()`, передав ей имя файла и режим чтения ('r'). Затем можно использовать цикл `for` для чтения каждого символа в отдельности. Но это неэффективный способ, лучше использовать функцию `read()`, которая возвращает весь текст в виде одной строки. Для записи текста в файл можно использовать функцию `write()`, передав ей строку и режим записи ('w').

Кроме того, в Python есть модуль `shutil`, который содержит функции для работы с архивами и файлами. Например, функция `copy()` позволяет копировать файлы из одного места в другое, а функция `move()` — перемещать их. Модуль `os` предоставляет множество функций для работы с файловой системой, таких как `os.makedirs()`, `os.listdir()` и `os.remove()`.

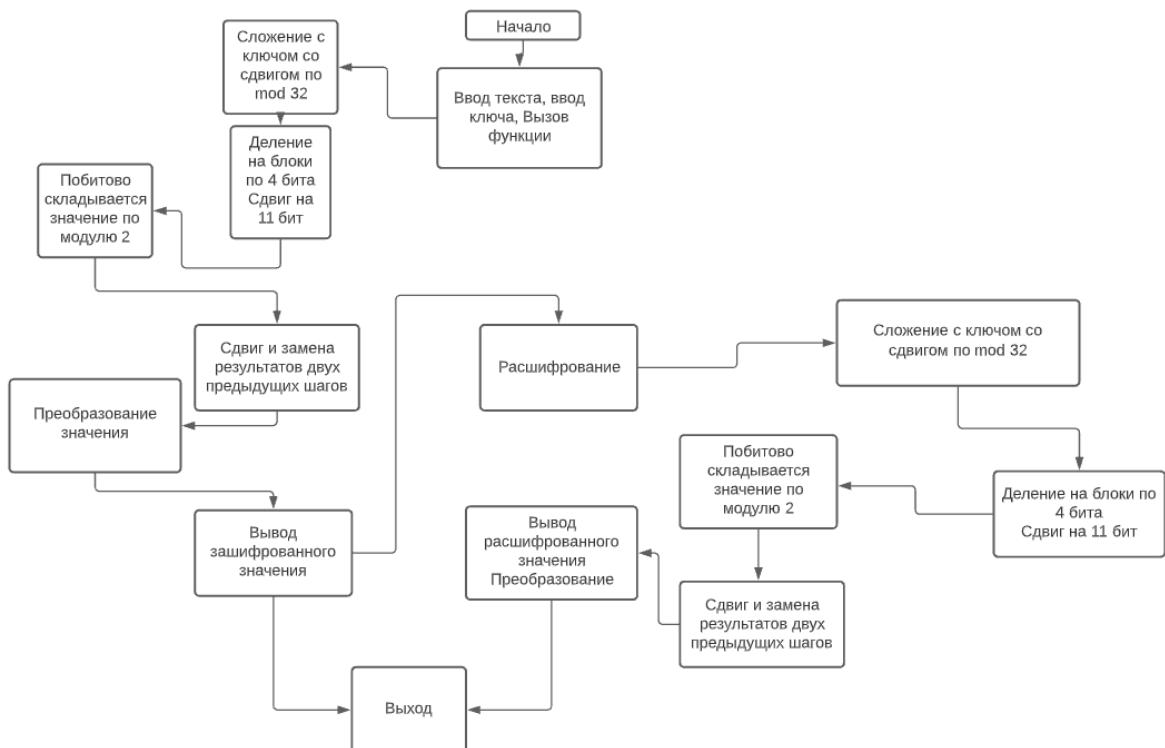
6 БЛОК Е: ШИФРЫ ГАММИРОВАНИЯ

6.1 Шифр ГАММИРОВАНИЕ ГОСТ 28147-89 (13)

1) Описание

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

2) Блок-схема программы



3) Код программы

```
# -*- coding: utf-8 -*-
import sys
import numpy.random
```

```
import itertools
from demo import alphabet, input_for_cipher_short, input_for_cipher_long,
output_from_decrypted
import binascii


class GostCrypt(object):
    def __init__(self, key, sbox):
        self._key = None
        self._subkeys = None
        self.key = key
        self.sbox = sbox

    @staticmethod
    def _bit_length(value):
        return len(bin(value)[2:])

    @property
    def key(self):
        return self._key

    @key.setter
    def key(self, key):
        self._key = key
        self._subkeys = [(key >> (32 * i)) & 0xFFFFFFFF for i in range(8)] #8
КУСКОВ

    def _f(self, part, key):
        temp = part ^ key
        output = 0
        for i in range(8):
            output |= ((self.sbox[i][(temp >> (4 * i)) & 0b1111]) << (4 * i))
        return ((output >> 11) | (output << (32 - 11))) & 0xFFFFFFFF

    def _decrypt_round(self, left_part, right_part, round_key):
        return left_part, right_part ^ self._f(left_part, round_key)

    def encrypt(self, plain_msg):
```

```

def _encrypt_round(left_part, right_part, round_key):
    return right_part, left_part ^ self._f(right_part, round_key)

left_part = plain_msg >> 32
right_part = plain_msg & 0xFFFFFFFF
for i in range(24):
    left_part, right_part = _encrypt_round(left_part, right_part,
self._subkeys[i % 8])
    for i in range(8):
        left_part, right_part = _encrypt_round(left_part, right_part,
self._subkeys[7 - i])
return (left_part << 32) | right_part

def decrypt(self, crypted_msg):
    def _decrypt_round(left_part, right_part, round_key):
        return right_part ^ self._f(left_part, round_key), left_part

    left_part = crypted_msg >> 32
    right_part = crypted_msg & 0xFFFFFFFF
    for i in range(8):
        left_part, right_part = _decrypt_round(left_part, right_part,
self._subkeys[i])
        for i in range(24):
            left_part, right_part = _decrypt_round(left_part, right_part,
self._subkeys[(7 - i) % 8])
    return (left_part << 32) | right_part

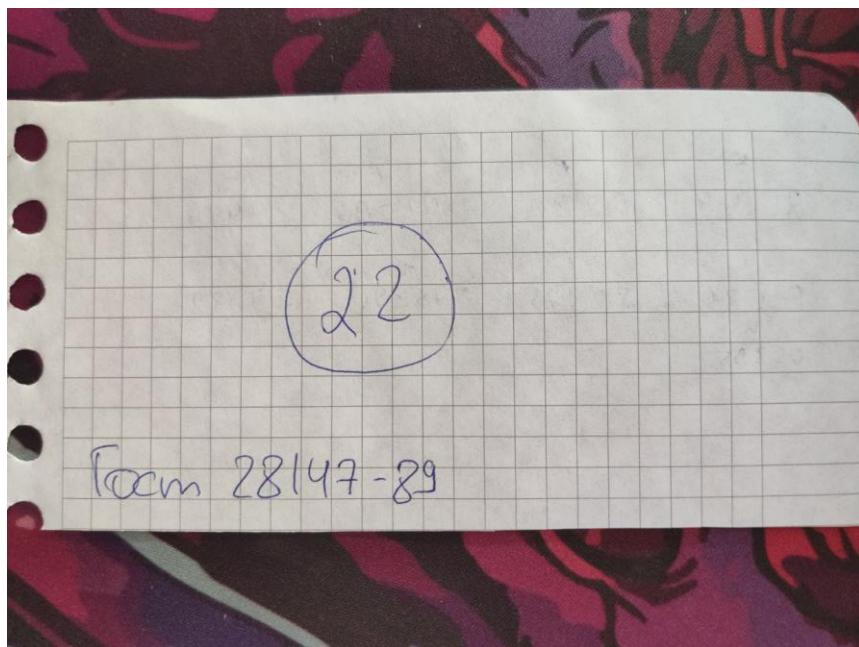
sbox = [numpy.random.permutation(l) for l in itertools.repeat(list(range(16)),
8)]
sbox = (
    (4, 10, 9, 2, 13, 8, 0, 14, 6, 11, 1, 12, 7, 15, 5, 3),
    (14, 11, 4, 12, 6, 13, 15, 10, 2, 3, 8, 1, 0, 7, 5, 9),
    (5, 8, 1, 13, 10, 3, 4, 2, 14, 15, 12, 7, 6, 0, 9, 11),
    (7, 13, 10, 1, 0, 8, 9, 15, 14, 4, 6, 12, 11, 2, 5, 3),
    (6, 12, 7, 1, 5, 15, 13, 8, 4, 10, 9, 14, 0, 3, 11, 2),
    (4, 11, 10, 0, 7, 2, 1, 13, 3, 6, 8, 5, 9, 12, 15, 14),
    (13, 11, 4, 1, 3, 15, 5, 9, 0, 10, 14, 7, 6, 8, 2, 12),
    (1, 15, 13, 0, 5, 7, 10, 4, 9, 2, 3, 14, 6, 11, 8, 12),
)

```

```
key =  
18318279387912387912789378912379821879387978238793278872378329832982398023031  
  
text_short = input_for_cipher_short().encode().hex()  
text_short = int(text_short, 16)  
  
gost_short = GostCrypt(key, sbox)  
  
enc_txt = gost_short.encrypt(text_short)  
dec_txt = gost_short.decrypt(enc_txt)  
dec_txt = bytes.fromhex(hex(dec_txt)[2::]).decode('utf-8')  
  
  
text_long = input_for_cipher_long().encode().hex()  
text_long = int(text_long, 16)  
  
print(f'''  
Зашифрованный текст:  
{enc_txt}  
Расшифрованный текст:  
{output_from_decrypted(dec_txt)}  
''' )
```

4) Тестирование

```
PS C:\Users\xiaomi\Desktop\cryptography_ciphers\lab_5\gost89> python3 .\gost89.py  
Введите текст  
Красивыми словами пастернак не помаслиш  
Зашифрованный текст:  
20154022991479527447030288875752905418905395191770125716947353830703838459216489491995987139374943186403280256065931194756968616301154931354263908686151430859726366026252421  
Расшифрованный текст:  
Красивыми словами пастернак не помаслиш
```



K	P	A	C
20154	02299	14795	27447
u	8	6	M
03020	80875	75290	54189
u			
05395			

5) Текст на 1000 символов

6.1 ШИФР ОДНОРАЗОВЫЙ БЛОКНОТ ШЕННОНА (14)

1) Описание

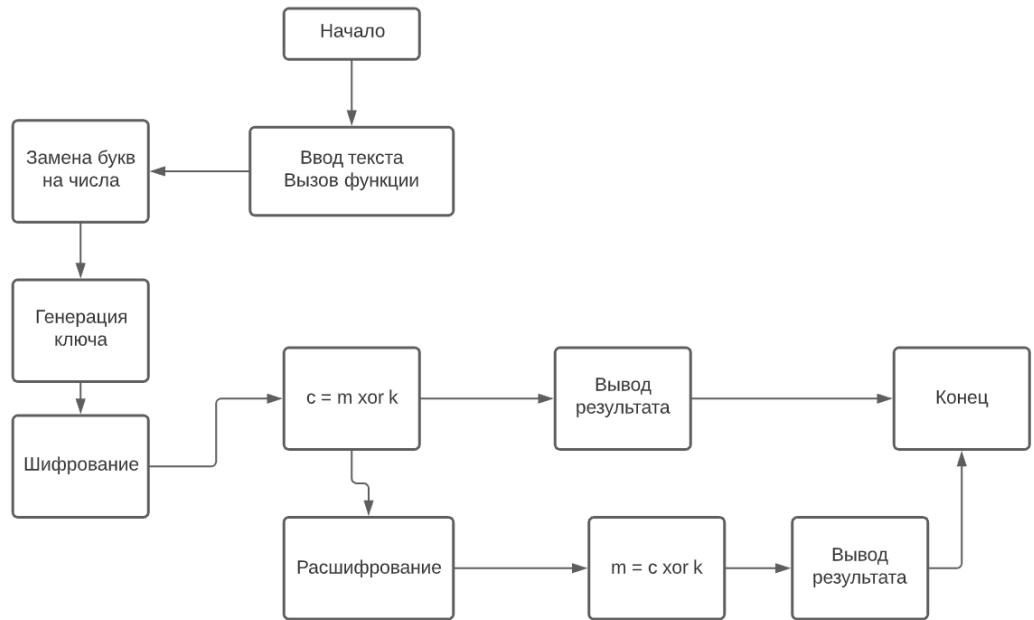
Популярность поточных шифров можно связывать с работой Клода Шеннона, посвященной анализу одноразовых гамма-блокнотов. Название «одноразовый блокнот» стало общепринятым в годы Второй мировой войны, когда для шифрования широко использовались бумажные блокноты.

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

Открытый текст сообщения ш записывают как последовательность бит или символов $m = m_0m_1\dots m_n$, а двоичную или символьную шифрующую последовательность к той же самой длины - как $k = k_0k_1\dots k_n$.

Шифртекст $s = c_0c_1\dots c_n$ определяется соотношением $C_j = m_i \oplus k$, при $0 \leq j < n$

2) Блок-схема программы



3) Код программы

```

import random

alphabet = "абвгдеёжзийклмнопрстуфхцчшъыъэюя "

alphabet = alphabet.replace(' ', '')
alphabet_lower = {}

i = 0
while i < (len(alphabet)):
    alphabet_lower.update({alphabet[i]: i})
    i += 1


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


def encode(msg):
    msg_list = list(msg)
    msg_list_len = len(msg_list)
    msg_code_bin_list = list()
    for i in range(len(msg_list)):
        msg_code_bin_list.append(alphabet_lower.get(msg_list[i]))

```

```

key_list = list()
for i in range(msg_list_len):
    key_list.append(random.randint(0, 32))

cipher_list = list()
for i in range(msg_list_len):
    m = int(msg_code_bin_list[i])
    k = int(key_list[i])
    cipher_list.append(int(bin(m ^ k), base=2))
return cipher_list, key_list

def 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

```

4) Тестирование

<input type="text" value="Красивыми словами пастернак не помыслишь"/> <small>Ведите текст</small>	<small>Зашифрованный текст:</small> <small>[7, 24, 23, 14, 28, 17, 28, 15, 4, 12, 29, 29, 5, 29, 14, 17, 28, 24, 28, 0, 28, 16, 22, 28, 7, 18, 8, 16, 2, 11, 0, 12, 28, 1, 12, 17], [12, 9, 23, 28, 29, 19, 0, 2, 13, 30, 17, 18, 7, 29, 3, 24, 12, 24, 14, 19, 17, 1, 24, 28, 12, 28, 13, 0, 13, 6, 0, 38, 16, 8, 21, 12]</small> <small>Расшифрованный текст:</small> <small>красивыми словами пастернак не помыслишь</small>
--	--

(22)

блочном шифре

k	P	A	C	U	B	G	M	a
[7	24	23	14	20	17	28	15	4
C	n	o	b	A	M	u		
[2	29	29	5	29	14	12	28	24

[12 9 23 28 29 19 0 2 13 30
17 18 7 29 3 24 12]

5) Текст на 1000 символов

Введите 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 рекомендуется использовать один или два слова и одну картинку. Текст на тысячу символов это сколько примерно слов? Статистика показывает, что ты сюда 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 проблемами или без. Установлено, что проблема увеличивает объем текста примерно на сто или двести символов именно столько раз мы разделяем слова свободным пространством. Считать проблемы заказчики не любят,

Зашифрованный текст:

([18, 2, 17, 8, 21, 25, 0, 13, 12, 5, 19, 20, 31, 10, 31, 28, 27, 24, 6, 31, 54, 22, 52, 22, 13, 5, 26, 16, 2, 9, 15, 23, 31, 14, 3, 28, 23, 22, 16, 7, 1, 8, 6, 24, 28, 43, 29, 16, 7, 31, 28, 8, 28, 23, 2, 7, 7, 31, 7, 23, 11, 16, 22, 6, 26, 1, 2, 49, 28, 21, 19, 8, 28, 60, 5, 9, 0, 23, 24, 18, 14, 2, 26, 28, 27, 19, 0, 28, 8, 0, 29, 8, 16, 15, 25, 5, 19, 29, 9, 4, 19, 31, 16, 9, 10, 15, 19, 15, 25, 1, 15, 3, 17, 26, 9, 35, 15, 11, 13, 25, 1, 19, 19, 8, 15, 4, 25, 15, 19, 6, 11, 29, 5, 1, 21, 19, 2, 21, 19, 13, 26, 15, 9, 28, 15, 28, 21, 16, 18, 4, 6, 43, 18, 7, 25, 25, 30, 3, 51, 3, 14, 5, 8, 16, 10, 20, 3, 24, 13, 12, 9, 18, 1, 19, 31, 1, 9, 3, 28, 15, 1, 15, 21, 24, 4, 21, 25, 26, 19, 15, 2, 5, 28, 11, 8, 22, 21, 9, 12, 21, 19, 13, 27, 22, 6, 8, 19, 7, 21, 18, 18, 14, 13, 11, 30, 29, 18, 8, 12, 4, 18, 9, 30, 30, 30, 16, 15, 18, 0, 28, 1, 38, 2, 7, 29, 9, 12, 6, 20, 6, 10, 9, 29, 4, 10, 53, 28, 52, 3, 29, 1, 31, 8, 11, 24, 15, 7, 23, 3, 1, 31, 13, 38, 26, 11, 20, 23, 9, 7, 16, 25, 14, 9, 18, 4, 25, 9, 29, 0, 17, 3, 4, 21, 4, 6, 5, 26, 23, 6, 31, 31, 7, 14, 4, 18, 24, 41, 24, 31, 1, 20, 26, 8, 27, 3, 1, 52, 17, 19, 23, 24, 28, 2, 25, 16, 15, 22, 29, 15, 18, 54, 22, 3, 26, 21, 8, 28, 30, 6, 24, 21, 15, 4, 22, 27, 5, 17, 17, 26, 19, 10, 7, 7, 5, 4, 12, 12, 4, 16, 26, 8, 28, 3, 16, 23, 31, 26, 15, 15, 21, 30, 1, 26, 26, 2, 4, 2, 29, 6, 10, 3, 16, 25, 11, 27, 8, 22, 19, 17, 50, 6, 17, 29, 4, 10, 28, 0, 3, 5, 7, 12, 30, 9, 31, 39, 16, 5, 28, 6, 49, 5, 21, 8, 26, 5, 58, 7, 30, 30, 29, 25, 14, 22, 11, 8, 31, 27, 26, 21, 1, 13, 15, 2, 11, 3, 28, 0, 21, 26, 27, 28, 0, 24, 4, 8, 13, 2, 30, 26, 22, 16, 3, 27, 5, 25, 1, 1, 9, 14, 23, 24, 48, 8, 4, 23, 7, 28, 19, 39, 18, 16, 17, 29, 6, 16, 31, 5, 0, 32, 15, 2 0, 14, 0, 6, 10, 14, 6, 9, 31, 15, 20, 19, 19, 7, 21, 9, 23, 5, 17, 22, 25, 19, 22, 60, 1, 12, 15, 19, 17, 4, 13, 25, 14, 5, 6, 23, 6, 20, 27, 18, 24, 10, 2, 18, 26, 34, 21, 14, 8, 7, 16, 9, 4, 21, 8, 30, 16, 7, 19, 16, 21, 6, 6, 5, 15, 31, 20, 21, 12, 8, 3, 2, 4, 19, 27, 12, 13, 13, 30, 0, 31, 16, 0, 14, 32, 28, 16, 21, 29, 5, 6, 26, 27, 2, 31, 12, 1, 22, 22, 11, 12, 18, 15, 18, 15, 10, 16, 45, 2, 19, 13, 14, 24, 6, 22, 58, 13, 20, 16, 5, 6, 19, 52, 28, 16, 6, 15, 13, 31, 18, 12, 19, 51, 28, 28, 49, 23, 2, 13, 28, 24, 24, 7, 24, 21, 26, 19, 7, 2, 5, 23, 16, 20, 22, 26, 15, 6, 5, 11, 29, 1, 0, 29, 30, 6, 20, 13, 30, 15, 15, 16, 17, 29, 8, 41, 1, 22, 19, 0, 14, 13, 18, , 20, 9, 31, 17, 3, 2, 1, 16, 9, 3, 21, 27, 18, 3, 7, 29, 16, 21, 22, 23, 0, 10, 4, 10, 13, 28, 37, 27, 18, 24, 26, 41, 13, 9, 19, 20, 2, 16, 8, 25, 16, 2, 13, 1, 16, 24, 1, 22, 3, 31, 5, 0, 1, 0, 14, 22, 23, 8, 22, 5, 30, 31 , 42, 30, 25, 10, 30, 1, 12, 26, 1, 29, 3, 17, 15, 13, 8, 20, 4, 12, 2, 29, 24, 30, 0, 25, 2, 18, 31, 19, 10, 24, 31, 17, 4, 25, 20, 30, 28, 27, 51, 15, 11, 11, 17, 23, 1, 31, 9, 1, 29, 4, 18, 14, 31, 25, 8, 6, 7, 26, 3, 26, 28, 38, 2, 18, 26, 91, [16, 13, 2, 24, 4, 16, 13, 8, 29, 23, 0, 28, 12, 23, 22, 18, 27, 11, 26, 13, 22, 14, 32, 4, 12, 12, 3, 6, 11, 24, 18, 17, 26, 29, 18, 5, 2, 26, 0, 15, 17, 1, 14, 28, 27, 19, 30, 9, 18, 8, 3, 2, 26, 0, 15, 17, 1, 14, 28, 27, 19, 30, 26, 1, 1, 12, 11, 7, 26, 1, 25, 15, 15, 15, 15, 7, 7, 4, 4, 11, 16, 7, 19, 26, 25, 4, 0, 25, 2, 12, 14, 6, 17, 14, 28, 25, 12, 24, 18, 9, 17, 4, 23, 18, 11, 9, 9, 27, 25, 19, 17, 19, 10, 2, 20, 6, 3, 10, 8, 11, 22, 14, 0, 8, 26, 18, 16, 16, 10, 7, 26, 1, 25, 23, 6, 15, 24, 30, 5, 3, 1, 14, 17, 12, 22, 13, 14, 16, 1, 25, 13, 23, 26, 28, 23, 6, 29, 18, 8, 26, 29, 12, 9, 5, 29, 14, 15, 4, 21, 8, 1, 21, 27, 17, 10, 7, 19, 2, 13, 15, 20, 19, 31, 28, 14, 18, 21, 16, 19, 26, 18, 8, 12, 24, 16, 4, 11, 23, 12, 18, 28, 21, 19, 0, 30, 4, 19, 28, 16, 0, 22, 2, 12, 14, 28, 28, 2, 1, 9, 3, 30, 29, 4, 23, 18, 3, 15, 1, 17, 21 , 10, 17, 19, 31, 28, 14, 18, 21, 16, 19, 26, 18, 8, 12, 24, 16, 4, 11, 23, 12, 18, 28, 21, 19, 0, 30, 4, 19, 28, 16, 0, 22, 2, 12, 14, 28, 28, 2, 1, 9, 3, 30, 29, 4, 23, 18, 3, 15, 1, 17, 21 , 11, 15, 8, 12, 20, 9, 20, 6, 29, 14, 18, 21, 16, 19, 26, 18, 8, 12, 24, 16, 4, 11, 23, 12, 18, 28, 21, 19, 0, 30, 4, 19, 28, 16, 0, 22, 2, 12, 14, 28, 28, 2, 1, 9, 3, 30, 29, 4, 23, 18, 3, 15, 1, 17, 21 , 5, 23, 22, 12, 9, 0, 29, 18, 5, 8, 12, 19, 1, 18, 7, 14, 8, 28, 15, 6, 23, 19, 9, 19, 20, 4, 18, 16, 2, 11, 14, 1, 15, 17, 27, 11, 30, 27, 2, 13, 31, 17, 7, 5, 15, 3, 18, 38, 17, 18, 15, 3, 24, 3, 20, 14, 12, 12, 30, 1, 2, 22, 19, 22, 16, 22, 8, 12, 31, 23, 26, 23, 18, 29, 29, 12, 19, 25, 27, 22, 1, 10, 30, 11, 24, 14, 25, 1, 5, 28, 12, 5, 30, 22, 9, 13, 3, 25, 1, 1, 15, 25, 15, 24, 18, 22, 26, 10, 16, 27, 23, 32 , 26, 2, 8, 15, 8, 23, 26, 3, 17, 28, 14, 11, 21, 7, 21, 4, 18, 8, 19, 27, 23, 2, 13, 10, 18, 9, 22, 21, 2, 3, 15, 17, 8, 14, 16, 18, 29, 13, 32, 25, 3, 6, 5, 14, 4, 17, 18, 28, 22, 29, 1, 28, 5, 14, 29, 8, 4, 29, 5, 14, 21, 13, 17, 22, 1, 5, 8, 25, 32, 30, 29, 0, 20, 15, 29, 20, 15, 31, 20, 23, 4, 12, 18, 31, 14, 32, 0, 14, 17, 15, 11, 17, 9, 19, 26, 25, 21, 23, 26, 24, 11, 11, 4, 18, 24, 7, 14, 17, 27, 30, 0, 24, 13, 16, 15, 28, 27, 10, 27, 15, 18, 13, 21, 29, 26, 16, 32, 3, 22, 22, 19, 1, 12, 9, 17, 4, 12, 28, 8, 16, 18, 25, 18, 28, 22, 23, 18, 9, 18, 38, 21, 4, 4, 15, 3, 8, 3, 9, 38, 32, 0, 1, 17, 8, 32, 0, 11, 28, 24, 13, 18, 1, 28, 21, 12, 3, 14, 2, 11, 14, 26, 30, 20, 18, 17, 1, 8, 3, 10, 6, 8, 30, 1, 27, 19, 18, 27, 20, 24, 18, 14, 14, 26, 19, 31, 12, 16, 0, 9, 6, 8, 9, 28, 19, 18, 10, 13, 17, 25, 12, 0, 12, 17, 5, 21, 12, 9, 15, 11, 12, 23, 15, 27, 32, 18, 27, 26, 30, 22, 1, 22, 14, 9, 29, 15, 18, 6, 17, 26, 10, 26, 1)

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 рекомендуется использовать один или два слова и одну картинку. Текст на тысячу символов это сколько примерно слов? Статистика показывает, что ты сюда 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 проблемами или без. Установлено, что проблема увеличивает объем текста примерно на сто или двести символов именно столько раз мы разделяем слова свободным пространством. Считать проблемы заказчики не любят,

7 БЛОК F: ПОТОЧНЫЕ ШИФРЫ

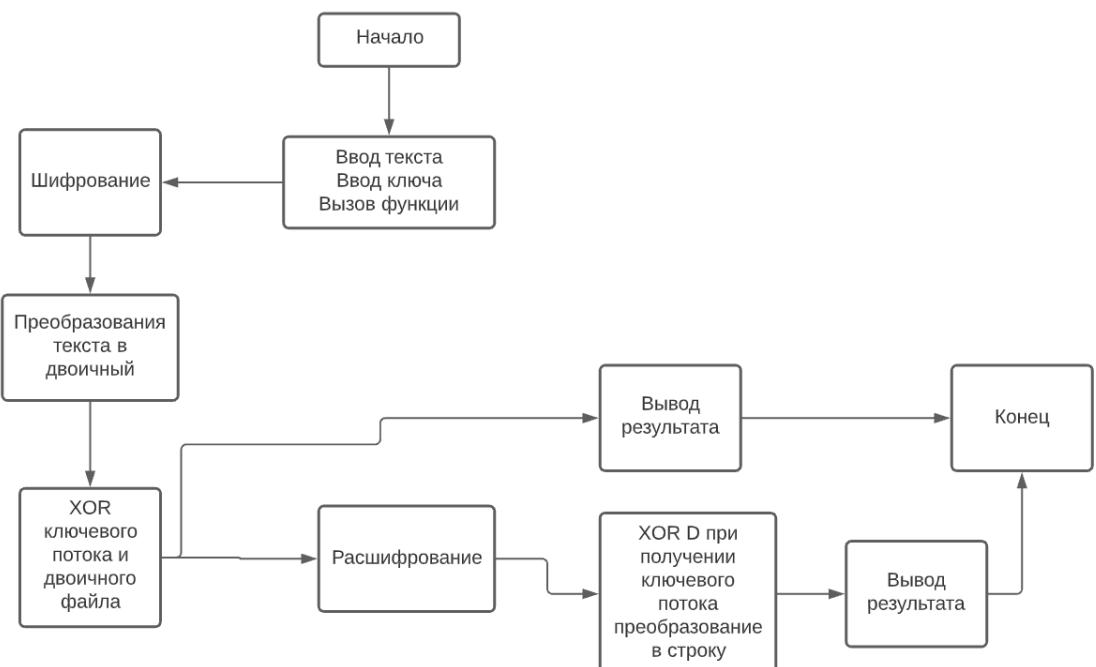
7.1 Шифр A5/1 (15)

1) Описание

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.

2) Блок-схема программы



3) Код программы

```
# -*- coding:utf-8 -*-
from demo import alphabet, input_for_cipher_short, 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 = []

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

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

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

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

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)

def encode(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

def decode(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))

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

# 010100100001101011000111000110010010100100000011011111010110111

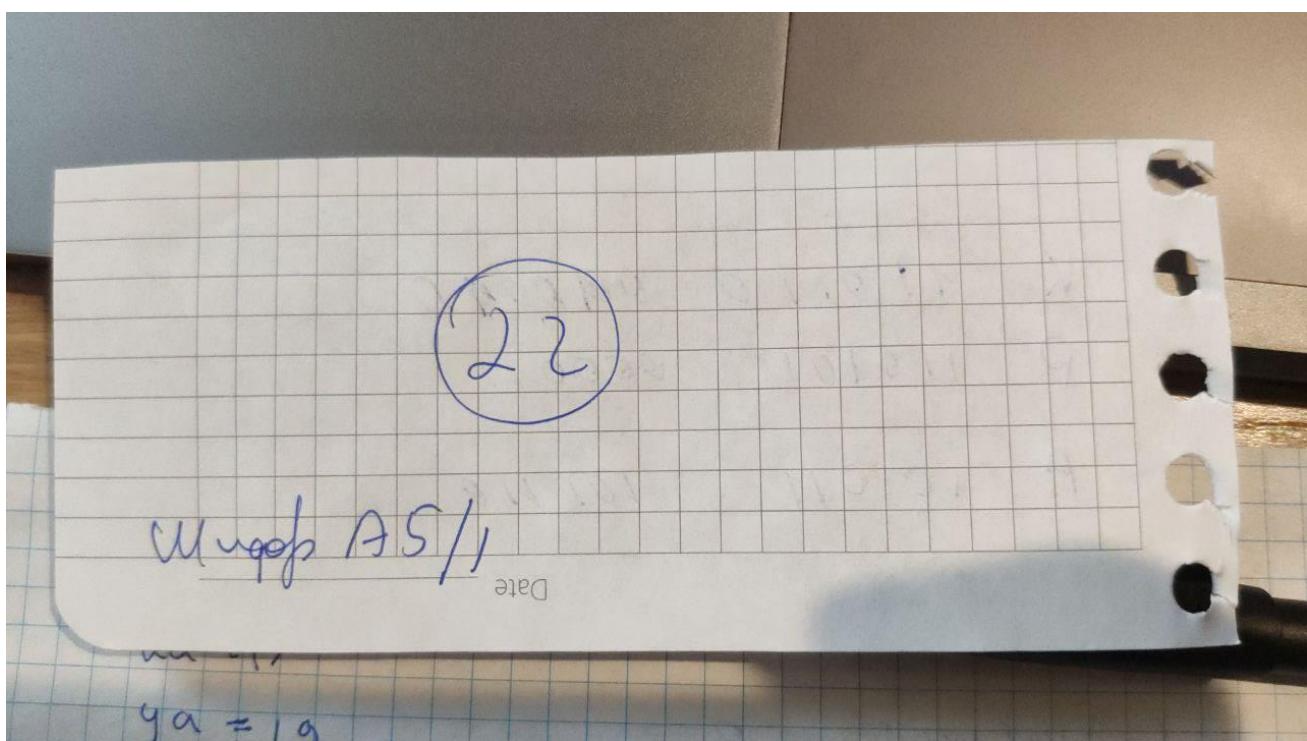
key = str(user_input_key())

```

```
set_key(key)

print(f'''  
Зашифрованный текст:  
{encode(input_for_cipher_short())}  
Расшифрованный текст:  
{output_from_decrypted(decode(encode(  
    input_for_cipher_short())))}  
''' )
```

4) Тестирование



K	110110	000000
P	110101	000001
A	101011	101110

Date

Yuri - (13)

5) Текст на 1000 символов

```
PS C:\Users\xiaomi\Desktop\cryptography_ciphers\lab_6\a51> python3 .\a51.py
```

Ведите 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 трех абзацев и обычно один заголовок. Но можно и без него. На тысячу символов рекомендуется использовать один или два слова и одну картинку. Текст на тысячу символов это сколько пример текста? Статистика показывает, что если в статье включает в себя сты пятьдесят или две строчки слов средней величины. Но, если злоупотребить предлогами, союзами и другими частями речи на один или два символа, то количество слов неизменно возрастает. В копирайтерской деятельности принято считать тысячами с проблемами или без. У通畅 проблем увеличивается объем текста примерно на сто или двести символов именно столько раз мы разделяем слова свободным пространством. Считать проблемы заказчиков не любят, как это за пусто место. Однако некоторые фирмы и видят спредварительной ставить стоимость за тысячу символов с проблемами, считая последние важным элементом качественного восприятия. Согласитесь, читать слитный текст без единого пропуска, никто не будет. Но большинству нужна цена за тысячу знаков без проблем.

Введите 64-bit ключ: 01010010000110101100011100011001001010010000001101111101011011

Зашифрованный текст:

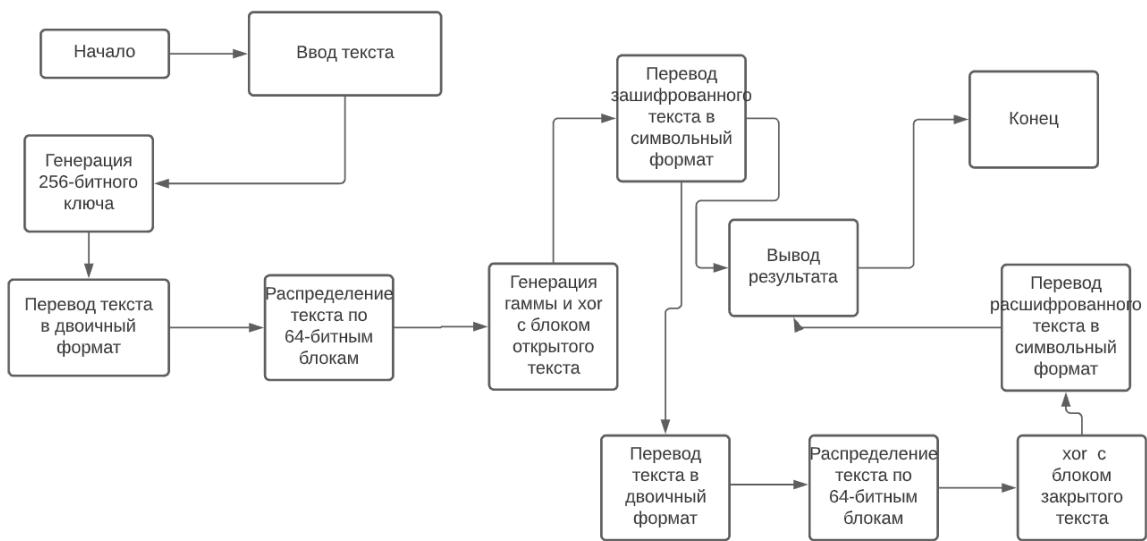
8 БЛОК G: КОМБИНАЦИОННЫЕ ШИФРЫ

8.1 Шифр МАГМА (17)

1) Описание

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

2) Блок-схема программы



3) Код программы

```
# импорт компонентов, необходимых для работы программы
```

```
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

print ('Введите текст')
to_encrypt = input()
```

```

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('ffeeddccbbaa99887766554433221100f0f1f2f3f4f5f6f7f8f9fafbfcdfeff', 16)

i = 0
text_short = to_encrypt
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'))

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

КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{encr_short}

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

main()

```

4) Тестирование

```

PS C:\Users\xiaomi\Desktop\cryptography_ciphers\lab_10\magma> python3 .\magma.py
Введите текст
Красивыми словами пастернак не помаслишь

МАГМА:
КЛЮЧ:
115761816795685524522806652725025505786200410505847444308688553892001406123775

КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
[14104369631423329055, 5054509616596506897, 12898358030550129305, 8068522935311515968, 16835954574122637496
, 15252017840864351372, 18444744497220829627, 933433738022392694, 50240876833766887, 16404698275537332922]

Расшифрованный текст:
Красивыми словами пастернак не помаслишь

PS C:\Users\xiaomi\Desktop\cryptography_ciphers\lab_10\magma>

```

Проверка по примеру из ГОСТ

```

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 rotl1(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('ffeedddccbaa99887766554433221100f0f1f2f3f4f5f6f7f8f9fafbfdf',
    16)
>>>
>>> magma_encrypt(int('fedcba9876543210', 16), key)
5686078090860087869
>>> int('4ee901e5c2d8ca3d', 16)
5686078090860087869
>>> magma_encrypt(int('fedcba9876543210', 16), key) ==
int('4ee901e5c2d8ca3d', 16)
True
```

5) Текст на 1000 символов

```
PS C:\Users\xiaomi\Desktop\cryptography_ciphers\lab_10\magma> python3 .\magma.py
```

Введите текст

Вот пример статьи на тысячу символов. Это достаточно маленький текст, оптимально подходящий для карточек товара 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 стопятьдесят или двести слов средней величины. Но, если злоупотреблять предлогами, сюзами 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 сорок процентов. 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.

МАГМА:

КЛЮЧ:

```
115761816795685524522806652725025505786200410505847444308688553892001406123775
```

КОРОТКИЙ ТЕКСТ:

Зашифрованный текст:

```
[14359360679599805757, 12685277312374858089, 2667118474714810119, 12319862187853716832, 8434050038248085483, 10071726320143396164, 7973995475251728780, 3547755811857389063, 10306886503249707528, 17454835303003071219, 1, 6924640608069374694, 10728741519359495825, 75136223818246669, 13180689774888211027, 16399109579122397444, 9756714696653168197, 15989759834534178049, 9801667378545550247, 8033652681544294730, 18107952748981932832, 11437477797259776203, 14095306192048267951, 10908008121199205063, 9967278521404385427, 11674884899395813190, 11227289996352286978, 12449357967070075325, 13901464993480517897, 6715724306090328274, 16551488750349984268, 17706356638910603080, 10827360545792114073, 6481263531518293115, 638289524755494659, 6758652357409891163, 1148671453306321691, 1, 1537984136415477057, 2550417712328674097, 11489979542620092442, 10600499856183920497, 1052201088673019386, 196912335575192365, 12536652642311017308, 16940342645864470290, 15472794475890367976, 885466372893731385, 10030140827279509611, 9691611694228521613, 15989759834534178049, 2899710365106271551, 13925191362318038386, 8076971601452144580, 8692441145595796326, 4972669465044858795, 9516537257148218061, 6758652357409891163, 3652614910132069273, 17823328546674048794, 14078999342056236256, 5273349685581107575, 5661438598169425038, 10863542892839543007, 7691049094682026909, 11437477797259776203, 16846916086007178313, 10554767423452825611, 10335474781029874525, 9879362418680151726, 5346967328490250768, 4057460703240969295, 13603411684891916210, 6360214449928167986, 9323086031911144658, 12715322944275189718, 1999264582017319410, 15289260373132081671, 10571009493083298815, 2346699712756694809, 10356924971308547229, 13990735642799502546, 6744761926711849017, 1952802480756416988, 6440143827400847284, 16878998971460208088, 6709615258459324581, 9864733468803932691, 9570481040452527541, 9878283213105084583, 13702105077703398490, 18264522114039397278, 11674884899395813190, 9891176404663018964, 4489306346917749030, 17379455870797960787, 2385892422708808321, 10305629430321463988, 12459858634362501357, 15091718010068719845,
```

6395947983, 7341533476182735460, 9753782918686210124, 668136742483333737, 6351355738272738612, 9156667591012550207, 13355264069952555938, 4474228379503693609, 12319862187853716832, 10071726320143396164, 11191877693159917519, 11180192821688693318, 7008462056628757749, 12304329932945990526, 10827360545792114073, 10374705805708783166, 2074796141061574560, 4080061714197831274, 7008462056628757749, 17427187758387558516, 17621673446572863963, 2730770862280929304, 1469107236998940534, 13825198160625090606, 252830099724767561, 1710706201369668323, 12685277312374858089, 2279299398805778860, 7229439830178305392, 7164090608319664395, 10827360545792114073, 2629297776519959089, 5099790616575193503, 6303045106133971510, 4466834888765277740, 4526494865159228242, 11330473927990969077, 9611853757892321121, 9650596426697186821, 1770971406522899636, 6372867096749245854, 17456146687817481297, 21725403387781760, 14127475102588350875, 17753793563406711008, 911664405894625009, 14290410647544626319, 11585766494367366624, 3396266981658896502, 138113508643393643, 4732411609144997917, 9897237904625538597, 3022080911037776780, 9677308884789179142, 16791044065858101251, 6006230909502237940, 8049089039016717789, 13241556452589613439, 16434858711434884630, 6050897895934808029, 1639276206729584883, 18158818095382714754, 178802707364353536, 2950821783567146896, 9460071816568651390, 3497628771392180522, 17493544587936903430, 12504901142051728415, 6475594396565797561, 9702187444171019729, 13959416895886208183, 13557180599640349798, 6736667299586546706, 8969087408330052577, 8063233049346207470, 1329481302222072647, 8427371148233375398, 2108407461501659947, 15377781857344184992, 15681193391117646300, 3050679451140470283, 44864024590894814120, 6142067374720541917, 488708437418147638, 7074162665920146770, 12627403521130439796, 6303045106133971510, 4466834888765277740, 11180192821688693318, 7008462056628757749, 11443714318012531150, 4474228379503693609, 4963456860800277821, 13491036623931188842, 726668423444149593, 5602618129461396309, 15247628652972161342, 18319889900469560529, 8706283879408684716, 7493767877350118609, 8299242569789435168, 6766061707386831762, 8971528550958227067, 7614577409788887631, 7141890055490050634, 7054316115548163699, 15785024483951499481, 8756880839685289505, 1840145209499476946, 3610837602476238650, 12319862187853716832, 8339410501177107321, 2731161190011158263, 252830099724767561, 879019830939153030, 11052333602287131332, 13787079903392782925, 13030258086912371517, 12601379616160262740, 14883478330485766993, 5958175528000152398, 3096824004660835781, 18069994757436189101, 723729663059138711, 7027878945512211340, 2550417712328674097, 739886078468727629, 7785376985192249536, 3176906383886696402, 1123139232973421551, 8889045086590960651, 2385892422708808321, 10305629430321463988, 3007830894932254811, 7080646299977939500, 1530364400755997631, 14738061815484779755, 540299310016161264, 1456421044545289451]

Расшифрованный текст:

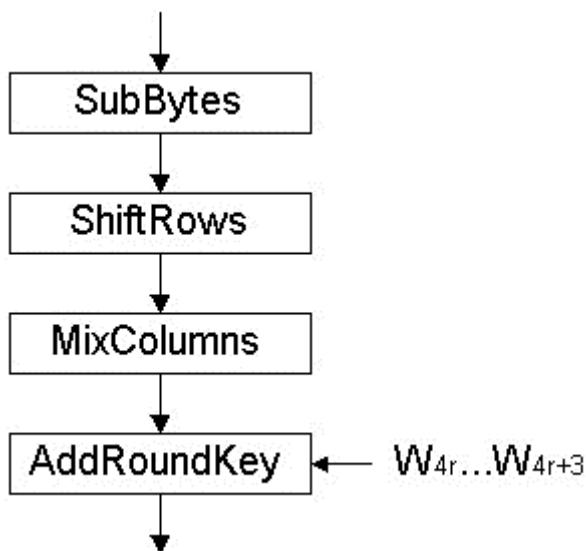
Вот пример статьи на тысячу символов. Это достаточно маленький текст, оптимально подходящий для карточек товаров в интернет или магазинах или для небольших информационных публикаций. В таком тексте редко бывает более двух или трёх абзацев и обычно один подзаголовок. Но можно и без него. На тысячу символов рекомендовано использовать один или два ключа и одну картину. Текст на тысячу символов это сколько примерно слов? Статистика показывает, что тысяча включает в себя стопятьдесят 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 копирайте рской 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 с линейный текст без единого пропуска, никто не будет. Но большинству нужна цена за тысячу знаков без пробелов.

8.2 Шифр AES (19)

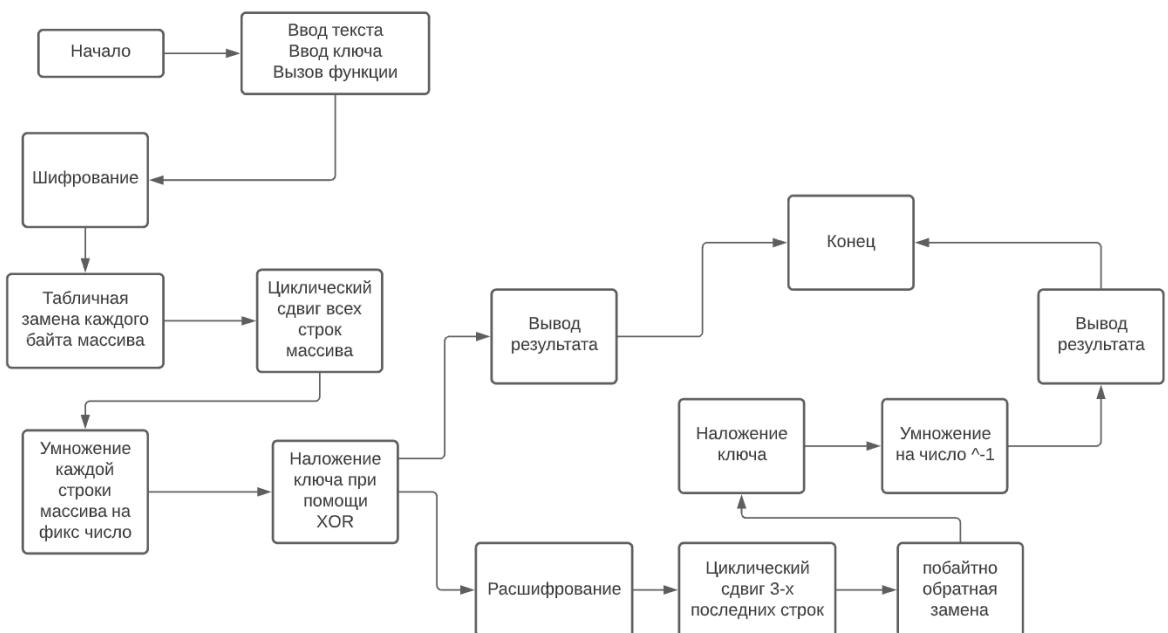
1) Описание

Американский стандарт, опубликованный в 2001 году. В современный криптографических продуктах, наверно, не найдется таких, которые бы не использовали AES. Используется в WI-FI, WinRAR, PGP. DES-предшественник AES.

AES - симметричный алгоритм блочного шифрования (размер блока 128 бит, ключ 128/192/256 бит), принятый в качестве стандарта шифрования правительством США по результатам конкурса AES.



2) Блок-схема программы



3) Код программы

```

if __name__ == '__main__':
    import os
    import time

    import aes128

    print('Шаг 1:')

```

```
while True:
    print('Нажмите 1 для шифрования, 2 для дешифрования')
    way = input()
    if way not in ['1', '2']:
        print('Неверный ввод')
        continue
    else:
        break
print()

print('Шаг 2:')
while True:
    print('Введите полное имя файла')
    input_path = os.path.abspath(input())

    if os.path.isfile(input_path):
        break
    else:
        print('Неверный ввод')
        continue
print()

print('Шаг 3:')
while True:
    print('Введите секретное слово(ключ) меньше 16 символов')
    key = input()

    if len(key) > 16:
        print('Слишком длинный ключ, введите еще раз')
        continue

    for symbol in key:
        if ord(symbol) > 0xff:
            print('Используйте только латинские буквы и цифры')
            continue

    break

time_before = time.time()
```

```
# Input data
with open(input_path, 'rb') as f:
    data = f.read()

if way == '1':
    crypted_data = []
    temp = []
    for byte in data:
        temp.append(byte)
        if len(temp) == 16:
            crypted_part = aes128.encrypt(temp, key)
            crypted_data.extend(cryptd_part)
            del temp[:]
    else:
        #padding v1
        # cryptd_data.extend(temp)

        # padding v2
        if 0 < len(temp) < 16:
            empty_spaces = 16 - len(temp)
            for i in range(empty_spaces - 1):
                temp.append(0)
            temp.append(1)
            cryptd_part = aes128.encrypt(temp, key)
            cryptd_data.extend(cryptd_part)

    out_path = os.path.join(os.path.dirname(input_path) , 'cryptd_' +
os.path.basename(input_path))

# Ounput data
with open(out_path, 'xb') as ff:
    ff.write(bytes(cryptd_data))

else: # if way == '2'
    decrypted_data = []
    temp = []
    for byte in data:
        temp.append(byte)
```

```
if len(temp) == 16:
    decrypted_part = aes128.decrypt(temp, key)
    decrypted_data.extend(decrypted_part)
    del temp[:]

else:
    #padding v1
    # decrypted_data.extend(temp)

    # padding v2
    if 0 < len(temp) < 16:
        empty_spaces = 16 - len(temp)
        for i in range(empty_spaces - 1):
            temp.append(0)
        temp.append(1)
        decrypted_part = aes128.encrypt(temp, key)
        decrypted_data.extend(crypted_part)

    out_path = os.path.join(os.path.dirname(input_path) , 'decrypted_' +
os.path.basename(input_path))

    # Output data
    with open(out_path, 'xb') as ff:
        ff.write(bytes(decrypted_data))

time_after = time.time()
```

4) Тестирование

```
PS C:\Users\xiaomi\Desktop\cryptography_ciphers\lab_7\aes> python3 demo.py
Шаг 1:
Нажмите 1 для шифрования, 2 для дешифрования

Неверный ввод
Нажмите 1 для шифрования, 2 для дешифрования

Неверный ввод
Нажмите 1 для шифрования, 2 для дешифрования

Неверный ввод
Нажмите 1 для шифрования, 2 для дешифрования
1
Шаг 2:
Введите полное имя файла
text.py
Неверный ввод
Введите полное имя файла
text.txt
Неверный ввод
Введите полное имя файла
test.txt

Шаг 3:
Введите секретное слово(ключ) меньше 16 символов
qwerty
```

Создался файл с зашифрованной информацией

```
lab_7 > aes >  crypted_test.txt
1  "♦_[_♦♦g♦m♦♦Z♦♦♦  ♦Ч♦ВС♦5♦b♦k♦U♦U♦$♦X♦♦q8♦  ♦з♦FY♦з♦"♦y♦7o♦"
```

```
PS C:\Users\xiaomi\Desktop\cryptography_ciphers\lab_7\aes> python3 demo.py
Шаг 1:
Нажмите 1 для шифрования, 2 для дешифрования
2

Шаг 2:
Введите полное имя файла
crypted_test.txt

Шаг 3:
Введите секретное слово(ключ) меньше 16 символов
qwerty
PS C:\Users\xiaomi\Desktop\cryptography_ciphers\lab_7\aes>
```

Создался расшифрованный файл

```
lab_7 > aes > _decryptd_cryptetd_test.txt
1      Красивыми словами пастернак не помастишь???????
```

5) Текст на 1000 символов

```
PS C:\Users\xiaomi\Desktop\cryptography_ciphers\lab_7\aes> python3 demo.py
Шаг 1:
Нажмите 1 для шифрования, 2 для дешифрования
1

Шаг 2:
Введите полное имя файла
test.txt

Шаг 3:
Введите секретное слово(ключ) меньше 16 символов
qwerty
PS C:\Users\xiaomi\Desktop\cryptography_ciphers\lab_7\aes>
```



A large block of binary data, consisting of approximately 1000 characters, primarily zeros and ones, representing the encrypted file content.

Расшифрованный текст на 1000 символов

```
lab_7 > aes > decrypted_cryptedit_test.txt
1   Вот пример статьи на тысячу символов. Это достаточно маленький текст, оптимально подходящий для карточек товаров в интернет или магазинах.
```

9 БЛОК Н: АССИМЕТРИЧНЫЕ ШИФРЫ

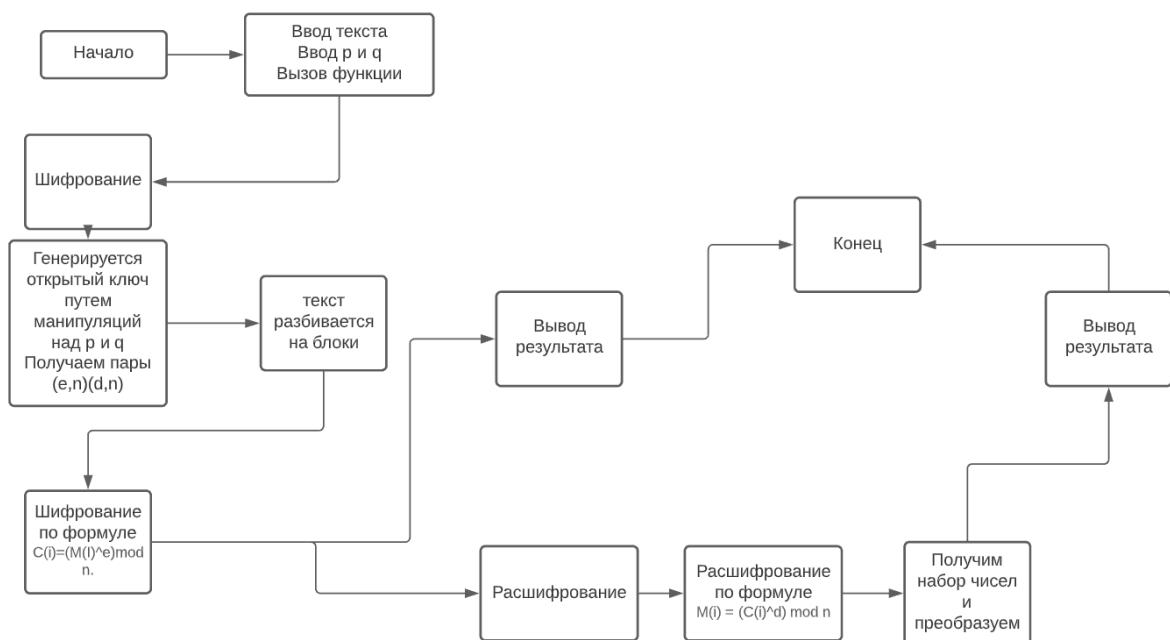
9.1 Шифр RSA (21)

1) Описание

На данный момент асимметричное шифрование на основе открытого ключа RSA (расшифровывается, как Rivest, Shamir and Aldeman - создатели алгоритма) использует большинство продуктов на рынке информационной безопасности.

Его криптостойкость основывается на сложности разложения на множители больших чисел, а именно - на исключительной трудности задачи определить секретный ключ на основании открытого, так как для этого потребуется решить задачу о существовании делителей целого числа. Наиболее криптостойкие системы используют 1024-битовые и большие числа.

2) Блок-схема программы



3) Код программы

```
# -*- coding: utf-8 -*-
import random
```

```

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

def multiplicative_inverse(e,r):
    for i in range(r):
        if((e*i)%r == 1):
            return i

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

def generate_keypair(p, q):
    if not (is_prime(p) and is_prime(q)):
        raise ValueError('Both numbers must be prime.')
    elif p == q:
        raise ValueError('p and q cannot be equal')
    #n = pq
    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))

```

```

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

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

if __name__ == '__main__':
    ...

    Detect if the script is being run directly by the user
    ...

    print("RSA")
    p = int(input("Введите p: "))
    q = int(input("Введите q: "))
    public, private = generate_keypair(p, q)
    print("Публичный ключ: ", public, "Секретный ключ: ", private)
    message = input("Введите сообщение: ")
    encrypted_msg = encrypt(private, message)
    print("Зашифрованное сообщение: ")
    print(''.join([str(x) for x in encrypted_msg]))
    print("Расшифрованное сообщение: ")
    print(decrypt(public, encrypted_msg))

```

4) Тестирование

```

PS C:\Users\xiaomi\Desktop\cryptography_ciphers\lab_8\rsa> python3 .\rsa.py
RSA
Введите p: 107
Введите q: 109
Публичный ключ: (6787, 11663) Секретный ключ: (2827, 11663)
Введите сообщение: Красивыми словами пастернак не помаслишь
Зашифрованное сообщение:
3326599796363814208528181875110612085563638147424868281896361106120855636291796363814534121115997182399636197056361023921115636291786811061963638147424208545261621
Расшифрованное сообщение:
Красивыми словами пастернак не помаслишь

```

Цифр RSA

(22)

Оригинальный номер: (1046787, 11663) Запись номер: (2327, 11663)

88975 43539 65351 73166 91090
71246 51738 18865 31011 ...

Паренесбург 191-351

5) Текст на 1000 символов

Расшифрованное сообщение:
вот пример статьи на тысячу символов. это достаточно маленький текст, оптимально подходящий для карточек товаров в интернет или магазинах или для небольших информации оных 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 один или два клюка и одну картинку. текст на тысячу символов это сколько примерно слов, статистика показывает, что тысяча включает в себя стоятьдесят или двести с слов средней величины, но, если злоупотреблять предлогами, союзами и другими частями речи на один или два символа, то количество слов неизменно возрастает. в колпак ерской деятельности принято считать тысячи с проблемами или без, учет проблем увеличивает объем текста примерно на сто или двести символов именно столько раз мы разделяем слова свободным пространством, считать проблемы заказчики не любят, так как это пустое место, однако некоторые фирмы и бирки видят справедливым ставить стоимость за тысячу символов с проблемами, считая последние важным элементом качественного восприятия. согласитесь, читать сплошной текст без единого пропуска, никто не будет. но большинству нужна цена за тысячу знаков без проблем.

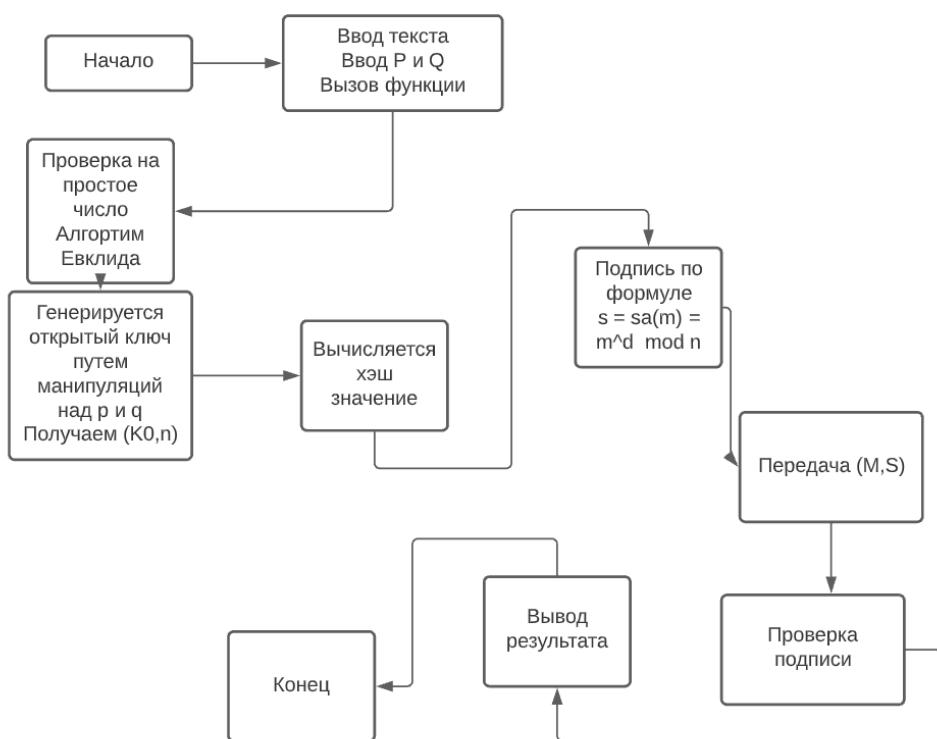
10 БЛОК I: АЛГОРИТМЫ ЦИФРОВЫХ ПОДПИСЕЙ

10.1 ЭЦП RSA (24)

1) Описание

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.

2) Блок-схема программы



3) Код программы

```
from math import gcd

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

```

'ю':30, 'я':31, ' ':32, ",":33, ".":34
}

#проверка на простое число
def IsPrime(n):
    d = 2
    while n % d != 0:
        d += 1
    return d == n

#расширенный алгоритм Евклида или (e**-1) mod fe
def modInverse(e,el):
    e = e % el
    for x in range(1,el):
        if ((e * x) % el == 1):
            return x
    return 1

#инициализация p,q,e,n
p = int(input("Введите p: "))
print(IsPrime(p))
q = int(input("Введите q: "))
print(IsPrime(q))
n = p * q
print("N =",n)
el = (p-1) * (q-1)
print("El =",el)
e = 257
print("E =",e)
if gcd(e,el) == 1:
    print(gcd(e,el),"E подходит")
else:
    print(gcd(e,el),"False")

#нахождение секретной экспоненты D
d = modInverse(e,el)
print("D =",d)
print("Открытый ключ e={} n={}".format(e,n))
print("Секретный ключ d={} n={}".format(d,n))

#хэширование сообщения
msg = input("Введите сообщение:")

```

```

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)))
def hash_value(n,alpha_code):
    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

hash_code_msg = hash_value(n, alpha_code_msg)
print("Хэш сообщения", hash_code_msg)
#подпись сообщения s=Sa(m) = m^d mod n
def signature_msg(hash_code,n,d):
    sign = (hash_code**d)%n
    return sign

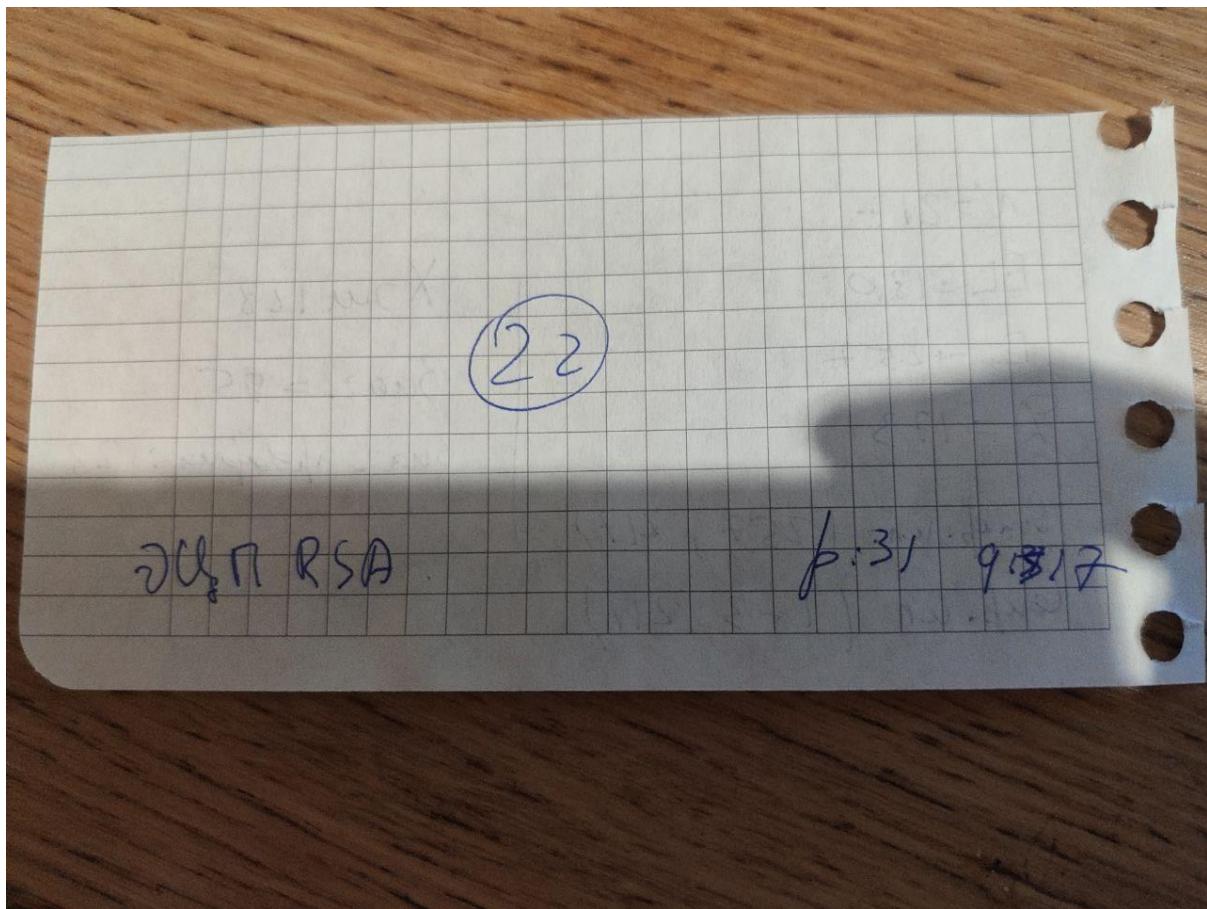
sign_msg = signature_msg(hash_code_msg,n,d)
print("Значение подписи: {}".format(sign_msg))
#передаём пару m,s
def check_signature(sign_msg, n,e):
    check = (sign_msg**e) % n
    return check

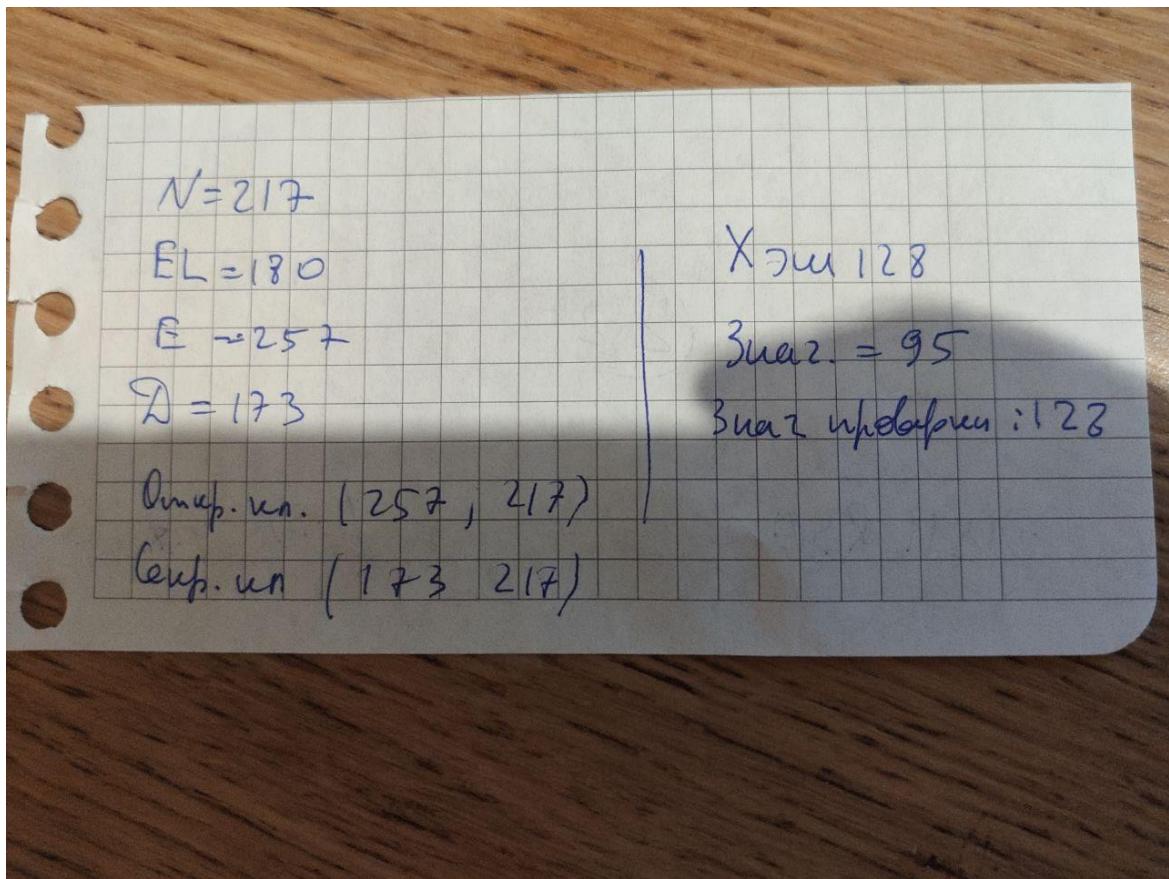
check_sign = check_signature(sign_msg,n,e)
print("Значение проверки подписи = {}".format(check_sign))

```

4) Тестирование

```
PS C:\Users\xiaomi\Desktop\cryptography_ciphers\lab_8\rsa> python3 main.py
Введите p: 31
True
Введите q: 7
True
N = 217
E1 = 180
E = 257
1 E подходит
D = 173
Открытый ключ e=257 n=217
Секретный ключ d=173 n=217
Введите сообщение:красивыми словами пастернак не помаслишь
Длина исходного сообщения 40 символов
Хэш сообщения 128
Значение подписи: 95
Значение проверки подписи = 128
```





5) Текст на 1000 символов

```
PS C:\Users\xiaomi\Desktop\cryptography_ciphers\lab_8\rsa> python3 main.py
Введите p: 31
True
Введите q: 7
True
N = 217
El = 180
E = 257
1 Е подходит
D = 173
Открытый ключ e=257 n=217
Секретный ключ d=173 n=217
Введите сообщение: 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 трех абзацев и обычно один подзаголовок, но можно и без него. на тысячу символов рекомендуется использовать один или два ключа и одну картину. текст на тысячу символов это сколько примерно слов. статистика показывает, что тысяча включает в себя стопятьдесят 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, а не без них. учет пробелов увеличивает объем текста примерно на сто или двести символов именно с только раз мы разделяем слова свободным пространством. считать пробелы заказчики не любят, так как это пустое место. однако некоторые фирмы и биржи видят справедливымставить стоимость за тысячу символов с пробелами, считая последние важным элементом качественного восприятия. согласитесь, читать слитный текст без единого пропуска никоим образом не будет, но большинству нужна цена за тысячу знаков без пробелов.
Длина исходного сообщения 1212 символов
Хэш сообщения 102
Значение подписи: 121
Значение проверки подписи = 102
PS C:\Users\xiaomi\Desktop\cryptography_ciphers\lab_8\rsa>
```

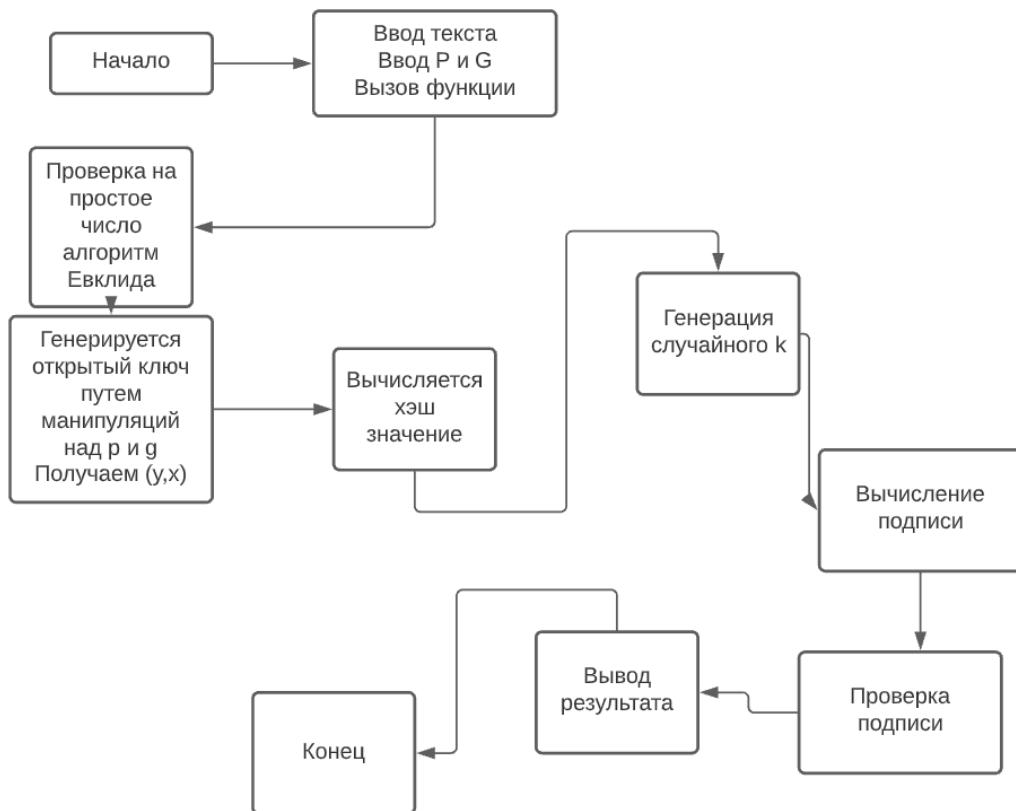
10.2 ЭЦП Elgamal (25)

1) Описание

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.

2) Блок-схема программы



3) Код программы

```

from math import gcd
import random

#инициализация алфавита
alphavit = {'а':0, 'б':1, 'в':2, 'г':3, 'д':4,
            'е':5, 'ж':6, 'з':7, 'и':8, 'й':9,
            'к':10, 'л':11, 'м':12, 'н':13, 'օ':14,
            'պ':15, 'ր':16, 'ս':17, 'տ':18, 'յ':19,
            'ֆ':20, 'խ':21, 'ց':22, 'չ':23, 'շ':24,
            'պ':25, 'ն':26, 'ա':27, 'ե':28, 'է':29,
            'յ':30, 'յ':31, ' ':32, ",":33, ".":34
            }

#проверка на простое число
def IsPrime(n):
    d = 2
    
```

```

while n % d != 0:
    d += 1
return d == n

#расширенный алгоритм Евклида или (e**-1) mod fe
def modInverse(e,el):
    e = e % el
    for x in range(1,el):
        if ((e * x) % el == 1):
            return x
    return 1

#выбор простого целого P, выбор целого числа G,G<P
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

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

p = gen_prime(10)
print("P =",p)
print()
g = random.randint(2,p-1)
print("G =",g)
print()
#отправитель выбирает случайное целое число X,1<x<(p-1)
x = random.randint(2,p-2)

```

```

y = (g**x)%p
print("Открытый ключ(Y)={}, Секретный ключ(X)={}{}".format(y,x))
print()
#хэшируем сообщение
msg = input("Введите сообщение:")
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)))
print()

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

hash_code_msg = hash_value(p, alpha_code_msg)
print("Хэш сообщения:= {}".format(hash_code_msg))
print()
#генерация случайное целое число K
k = 1
while True:
    k = random.randint(1,p-2)
    if gcd(k,p-1) == 1:
        print("K =",k)
        break

#отправитель вычисляет число целое число a
a = (g**k)%p
#вычисляем b
b = modInverse(k,p-1) * ((hash_code_msg - (x * a))%(p-1))
#b = modInverse((int(hash_code_msg) - int(x)*int(a)),p-1)
print("Значение подписи:S={},{}".format(a,b))
print()

```

```
#проверка подписи (передвём m, a,b)
check_hash_value = hash_value(p, alpha_code_msg)
a_1 = ((y**a) * (a**b)) % p
print("A1={}".format(a_1))
print()
a_2 = (g**check_hash_value)%p
print("A2={}".format(a_2))
print()
if a_1 == a_2:
    print("Подпись верна")
else:
    print("Подпись неверна")
```

4) Тестирование

```
PS C:\Users\xiaomi\Desktop\cryptography_ciphers\lab_8\elgamal> python3 main.py
P = 727
G = 257
Открытый ключ(Y)=563, Секретный ключ(X)=718
Введите сообщение:красивыми словами пастернак не помаслишь
Длина исходного сообщения 40 символов
Хэш сообщения:= 482
K = 155
Значение подписи:S=715,35066
A1=451
A2=451
Подпись верна
```

(22)

2411 2462

P=777
G=257

Онл.книги = 563

Закр. книги = 719

Хочу : 482

K=155

S=715,35066

A₁=451 A₂=451 => Всеса

5) Текст на 1000 символов

```
PS C:\Users\xiaomi\Desktop\cryptography_ciphers\lab_8\elgamal> python3 main.py
```

```
P = 863
```

```
G = 268
```

```
Открытый ключ(Y)=27, Секретный ключ(X)=240
```

Ведите сообщение: 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 трех абзацев и обычно один подзаголовок, но можно и без него. на тысячу символов рекомендуется использовать один или два ключа и одну картину. текст на тысячу символов это сколько примерно слов. статистика показывает, что тысяча включает в себя стопятьдесят 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 с только 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.

```
Длина исходного сообщения 1212 символов
```

```
Хэш сообщения:= 639
```

```
K = 743
```

```
Значение подписи:S=184,249165
```

```
A1=427
```

```
A2=427
```

```
Подпись верна
```

11 БЛОК Ј: СТАНДАРТЫ ЦИФРОВЫХ ПОДПИСЕЙ

11.1 Стандарт ГОСТ Р 34.10-94 (26)

1) Описание

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 $aq \bmod p = 1$;

x - некоторое число, меньшее q ;

$y = ax \bmod p$.

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

2) Блок-схема программы



3) Код программы

```
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, 'օ': 15,
            'պ': 16, 'ր': 17, 'ս': 18, 'տ': 19, 'յ': 20,
```

```

        'Ѡ': 21, 'Ӯ': 22, 'Ӵ': 23, 'ӵ': 24, 'ӷ': 25,
        'ӹ': 26, 'Ӱ': 27, 'Ӳ': 28, 'ӱ': 29, 'Ӹ': 30,
        'ӻ': 31, 'Ӽ': 32
    }

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")

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

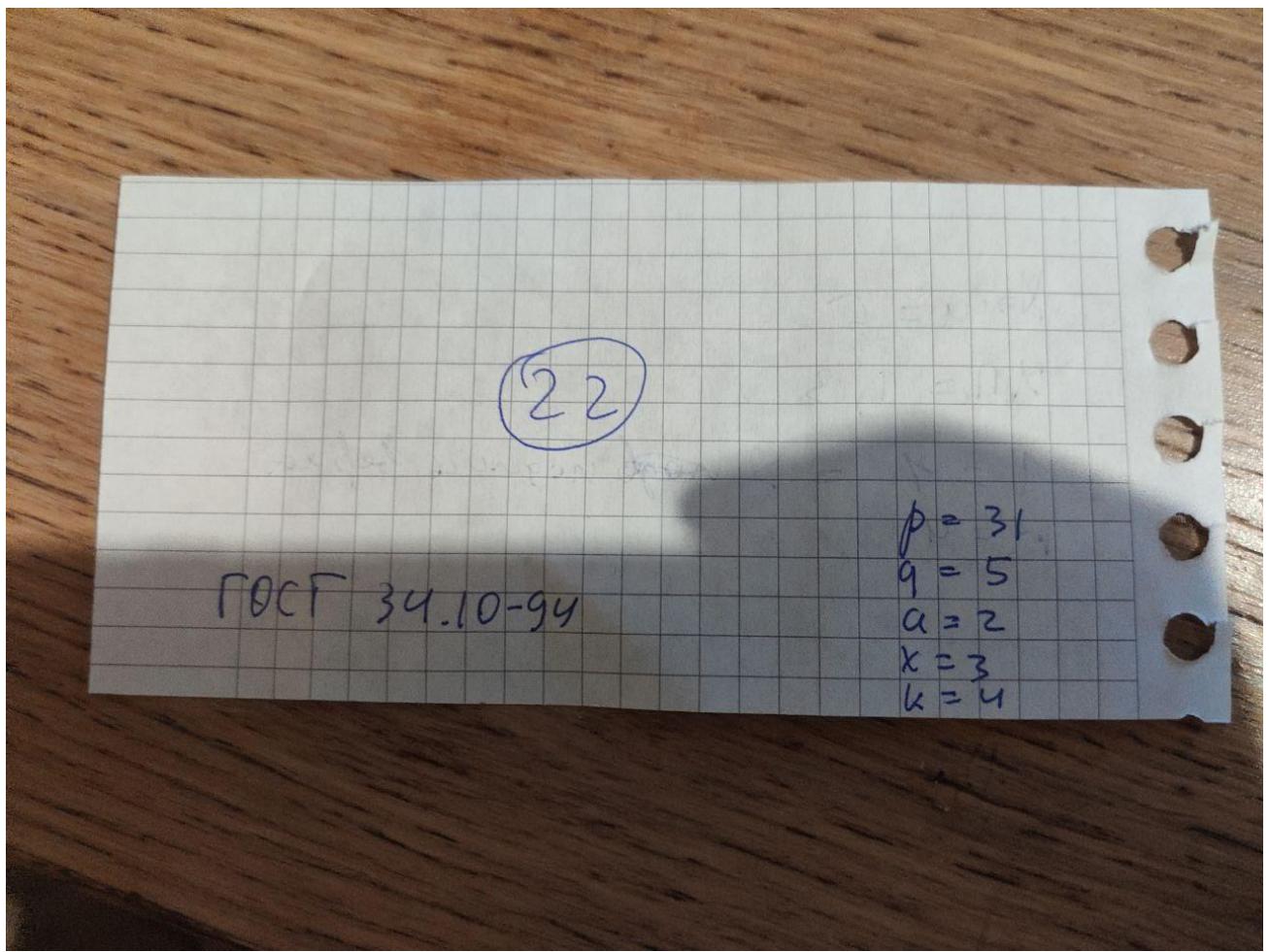
def main():
    print('ГОСТ Р 34.10-94:')
    message = input("Введите сообщение: ")
    ciphergostd(message)

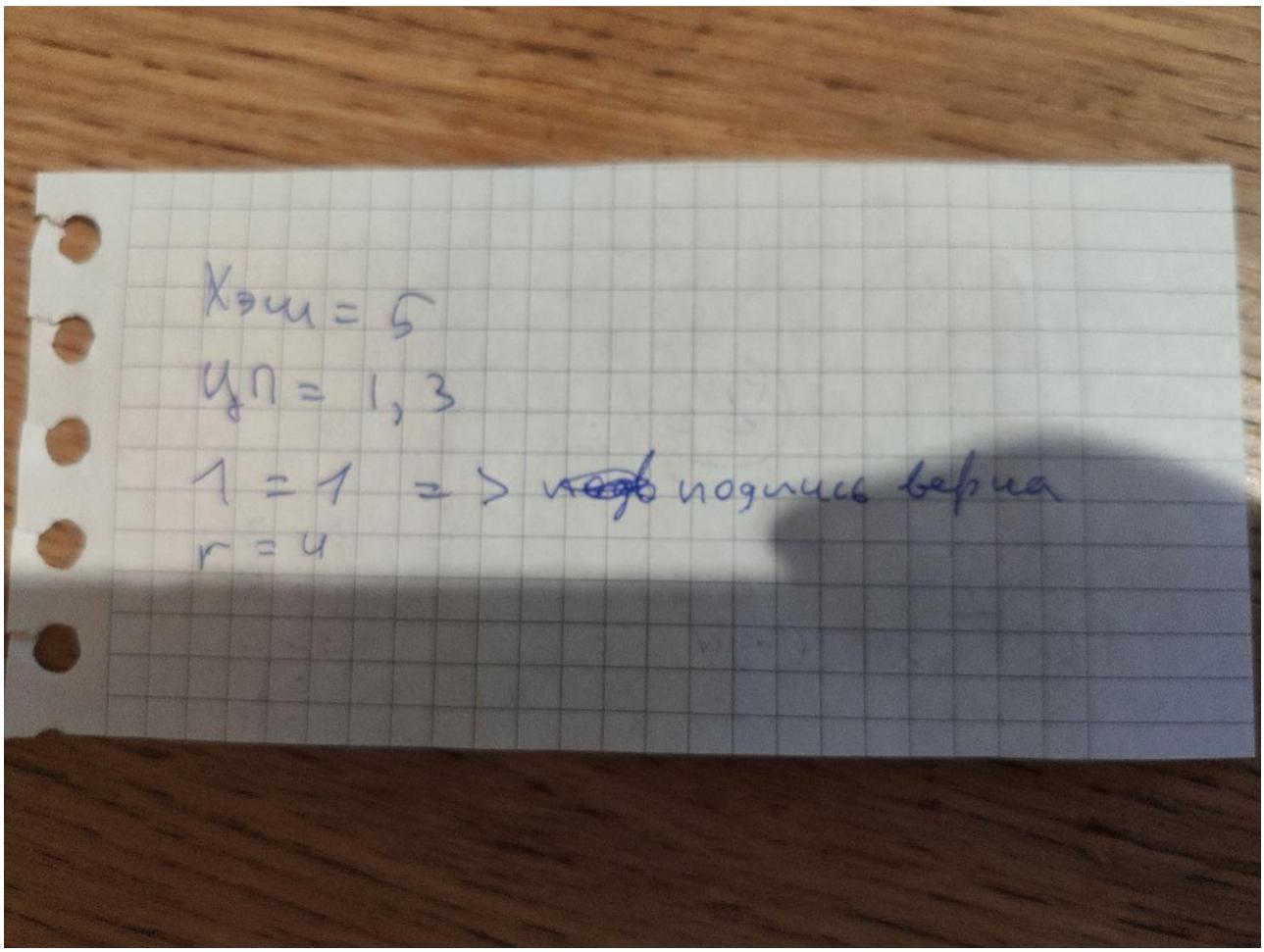
```

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

4) Тестирование

```
PS C:\Users\xiaomi\Desktop\cryptography_ciphers\lab_10\gost_94> python3 .\gost.py
ГОСТ Р 34.10-94:
Ведите сообщение: красивыемисловамиастернакнепомаслишь
p = 31
q = 5
a = 2
x = 3
k = 4
Длина исходного сообщения 36 символов
Хэш сообщения:= 5
Цифровая подпись = 1 , 3
1 = 1
r = u, следовательно:
Подпись верна
```





5) Текст на 1000 символов

```
PS C:\Users\xiaomi\Desktop\cryptography_ciphers\lab_10\gost_94> python3 .\demo.py
Введите 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 слова и одну картинку. Текст на тысячу символов это сколько примерно слов? Статистика показывает, что тысяча включает в себя стопятьдесят 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 стоимостью 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.
p = 31
q = 5
a = 2
x = 3
k = 4
Длина исходного сообщения 1087 символов
Хаш сообщения= 20
Цифровая подпись = 1 , 3
1 = 1
r = u, следовательно:
Подпись верна
```

11.2 Стандарт Р 34.10-2012 (27)

1) Описание

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

$x < q$, и вычисляет точку на кривой $Y_u = [x]G$ (открытый ключ).

Параметры кривой и список открытых ключей передаются всем пользователям.

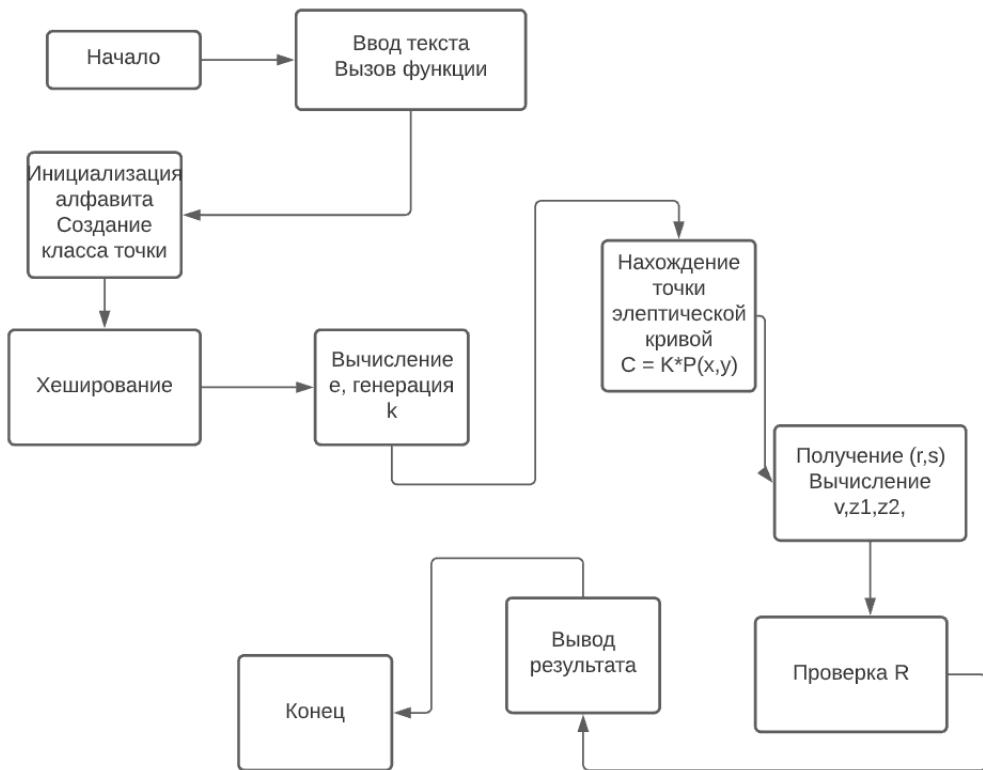
Чтобы подписать сообщение , пользователь А делает следующее:

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 $s = (kh + rxa) \bmod q$ (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.

2) Блок-схема программы



3) Код программы

```

import random
import collections

alphabet_lower = {'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, 'օ': 15,
                  'ռ': 16, 'պ': 17, 'ս': 18, 'տ': 19, 'յ': 20,
                  'Փ': 21, 'Խ': 22, 'Ը': 23, 'Ծ': 24, 'Շ': 25,
                  'թ': 26, 'Ե': 27, 'Յ': 28, '՚': 29, '՚՛': 30,
                  '՚Ո': 31, '՚Յ': 32
}

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

EllipticCurve = collections.namedtuple(
    'EllipticCurve', 'name p q_mod a b q g n h')
curve = EllipticCurve(
    'secp256k1',
    p=0xfffffffffffffffffffffffffffffffffffffefffffc2f,
    q_mod=0xfffffffffffffeffffffffcffffffffffffcfffffffffffffefffffc2f,

    a=7,
    b=11,

    g=(0x79be667ef9dcb bac55a06295ce870b07029bfcd b2dce28d959f2815b16f81798,
        0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8),
    q=(0xA0434D9E47F3C86235477C7B1AE6AE5D3442D49B1943C2B752A68E2A47E247C7,
        0x893ABA425419BC27A3B6C7E693A24C696F794C2ED877A1593CBEE53B037368D7),

    n=0xfffffffffffffffffffffebaaedce6af48a03bbfd25e8cd0364141,
    h=1,
)

def ciphergostd(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!")

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


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


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


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


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


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

def main():
    print('ГОСТ Р 34.10-2012:')

if __name__ == "__main__":
    main()

```

4) Тестирование

```

PS C:\Users\xiaomi\Desktop\cryptography_ciphers\lab_10\gost_2012> python3 .\demo.py
Введите текст
Красивыми словами пастернак не помаслишь
Длина исходного сообщения 36 символов
Q mod 1157920892372108883131902140479076077470404524942491262870694982560773809634351
P mod 11579208923731619542357985008687907853269984665640564039457584007908834671663
Хэш сообщения:=41734968009033504843595667972719937160002463522949733803928547400276299376185
E=41734968009933504843595667972719937160002463522949733803928547400276299376185
K=54101491378372682162188286106143732389464295481696833657503317461199861084485
D=10
Point_C=( x=33546030276856489284313896481160760174851777093607859709967439619313428716642, y=369451525768681177800000903791635480779893439661162303426012330480200971
24338)
R=33546030276856489284313896481160760174851777093607859709967439619313428716642
S=33911622774720931660243986213563325266950198836687131218874451305856193136938
V=4854799953503456083513215788152760512194084907292994747664768790559022707003
Point_Q=( x=10980558621116626629432866892583231117554510260596600142888290125507993067118, y=5124383235504058321191534323736250822297443681753984114121156474938550
647252 )
R_new=33546030276856489284313896481160760174851777093607859709967439619313428716642
Подпись прошла проверку!

```

Проверка по примеру из Гост

```

type 'help' or 'copyright' or 'credits' or 'license' for more information
>>> from publickey.ec import ECPoint
>>> from publickey.gost import DSGOST
>>> 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)
...     assert sign == expected
...     print(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 = 57520216126176808443614050338071176630104906313632182896741342206604859403
...     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) == True)
...
>>> test_gost_sign()
True
>>> test_gost_verify()
True
>>> █

```

5) Текст на 1000 символов

```

PS C:\Users\xiaomi\Desktop\cryptography_ciphers\lab_10\gost_2012> python .\demo.py
Введите 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 стопятьдесят 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 копиях терпкой 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 проблы заказчики не любят, так как это пустое место. Однако некоторые фирмы и биржи видят справедливым ставить стоимостью 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.
Длина исходного сообщения 1087 символов
Q mod 115792089237210883131902140479076077470404524942491262870694982560773809634351
R mod 11579208923731619542357098500868790785326998466564054039457584007908834671663
Хэш сообщения:=34910327282876669981090751617174695655616842716415000931628825868664952301091
E=34910327282876669981090751617174695655616842716415000931628825868664952301091
K=51928834512219181248301315739408102453486096158365035370298806285380244488479
D=10
Point_C=( x-8509270981536802167904415151341490324671440174964840286300895809991344314208, y=80034903046514627328106970716047346016623799727680274313024615692514952
20177)
R=8509270981536802167904415151341490324671440174964840286300895809991344314208
S=655429833996348299120148268203820056293982042304588932737216706138149533788
V=80771434365083562859973299123302845370380616728395900081312955467152482216320
Point_Q=( x-1098055862116620662943286689258323117554510260596600142888290125507993067118, y=51243083235504058321191534323736250822297443681753984114121156474938550
647252 )
R_new=8509270981536802167904415151341490324671440174964840286300895809991344314208
Полиссы прошли проверку!

```

12 БЛОК К: ОБМЕН КЛЮЧАМИ

12.1 ОБМЕН КЛЮЧАМИ ПО ДИФФИ-ХЕЛЛМАНУ

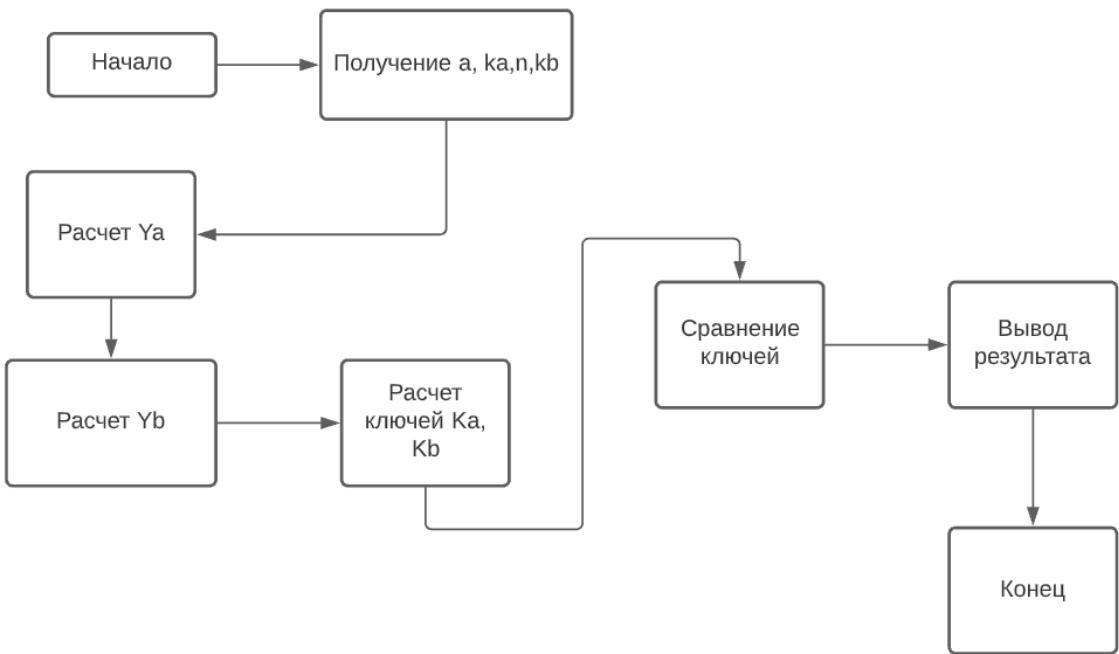
1) Описание

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 : $Y = aK \bmod n$
4. Обменяться ключами Y_A и Y_B по открытому каналу связи.
5. Независимо определить общий секретный ключ K : $K_A = Y_B K \bmod n$
 $K_B = Y_A K \bmod n$.

$$K_A = K_B = K$$

2) Блок-схема программы



3) Код программы

```

def main():
    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)

if __name__ == "__main__":
    main()

```

4) Тестирование

```
PS C:\Users\xiaomi\Desktop\cryptography_ciphers\lab_11> cd ..\dh
PS C:\Users\xiaomi\Desktop\cryptography_ciphers\lab_11\dh> python3 .\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
PS C:\Users\xiaomi\Desktop\cryptography_ciphers\lab_11\dh> python3 .\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
```

Handwritten calculations for the Diffie-Hellman key exchange:

- Initial values:
 $a: 5, n: 23, ka: 5$
- Intermediate calculation:
 $a^k \equiv 15625$
- Calculation of Y_a :
 $Y_a = (a^k) \bmod n = 8$
- Value of Y_b :
 $Y_b = 19$
- Intermediate calculation:
 $Y_a * Y_b \equiv 152$
- Final calculation:
 $a^k * (Y_a * Y_b) \bmod n = 5^k * 152 \bmod 23 = 12$

$$a : 5$$

$$n : 23$$

$$ka : 15$$

$$ya = 19$$

$$yb = 8$$

Общий : 12

5) Проверка