ГОСТ Р 34.10-94

Код:

import random

import math

def is\_prime(n):

    if n <= 1:

        return False

    if n <= 3:

        return True

    if n % 2 == 0 or n % 3 == 0:

        return False

    i = 5

    while i \* i <= n:

        if n % i == 0 or n % (i + 2) == 0:

            return False

        i += 6

    return True

def pred(s):

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

    s = s.lower().replace(' ', '')

    for sim in s:

        if sim not in llst:

            if sim == '.':

                s = s.replace('.', 'тчк')

            elif sim == ' ':

                s = s.replace(' ', 'пробел')

            elif sim == '?':

                s = s.replace('?', 'впрс')

            elif sim == '!':

                s = s.replace('!', 'всклзн')

            elif sim == ':':

                s = s.replace(':', 'двтч')

            elif sim == '–':

                s = s.replace('–', 'минус')

            elif sim == ',':

                s = s.replace(',', 'зпт')

            elif sim == '-':

                s = s.replace('-', 'тире')

            elif sim == 'ё':

                s = s.replace('ё', 'е')

            elif sim == ',':

                s = s.replace(',', 'зпт')

            elif sim == '-':

                s = s.replace('-', 'тире')

            elif sim == 'ё':

                s = s.replace('ё', 'е')

            elif sim == '0':

                s = s.replace('0', 'ноль')

            elif sim == '1':

                s = s.replace('1', 'один')

            elif sim == '2':

                s = s.replace('2', 'два')

            elif sim == '3':

                s = s.replace('3', 'три')

            elif sim == '4':

                s = s.replace('4', 'четыре')

            elif sim == '5':

                s = s.replace('5', 'пять')

            elif sim == '1':

                s = s.replace('6', 'шесть')

            elif sim == '2':

                s = s.replace('7', 'семь')

            elif sim == '3':

                s = s.replace('8', 'восемь')

            elif sim == '4':

                s = s.replace('9', 'девять')

            else:

                s = s.replace(sim, '')

    return s

def hesh(str,p,i):

    alp = " АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ"

    if i==0:

        q=(alp.index(str[i])\*\*2)%p

        return q

    else:

        q=((hesh(str,p,i-1)+alp.index(str[i]))\*\*2)%p

        return q

### ГОСТ Р 34.10-94 подпись ###

def decgost94(s, P, Q, A, X):

    llst = ['а', 'б', 'в', 'г', 'д', 'е', 'ж', 'з', 'и', 'й', 'к', 'л', 'м', 'н', 'о', 'п', 'р', 'с', 'т', 'у', 'ф',

            'х', 'ц', 'ч', 'ш', 'щ', 'ъ', 'ы', 'ь', 'э', 'ю', 'я']

    s = pred(s)

    Y = A \*\* X % P

    print("Y = ", Y)

    # K = random.randint(2,Q)

    K = 4

    h = 0

    for x in s:

        x = llst.index(x)

        h = ((h + x) \*\* 2) % 32

    print("h0 = ", h)

    R = ((A \*\* K) % P) % Q

    if R == 0:

        while R == 0:

            # K = random.randint(2,Q)

            R = ((A \*\* K) % P) % Q

    print("k = ", K)

    print("R = ", R)

    S = ((X \* R) + (K \* h)) % Q

    print("S = ", S)

    return R, S, Q, A, Y, P, s

def checkgost94(s, key):

    s = pred(s)

    R = int(key[0])

    S = int(key[1])

    Q = int(key[2])

    A = int(key[3])

    Y = int(key[4])

    P = int(key[5])

    llst = ['а', 'б', 'в', 'г', 'д', 'е', 'ж', 'з', 'и', 'й', 'к', 'л', 'м', 'н', 'о', 'п', 'р', 'с', 'т', 'у', 'ф',

            'х', 'ц', 'ч', 'ш', 'щ', 'ъ', 'ы', 'ь', 'э', 'ю', 'я']

    h = 0

    for x in s:

        x = llst.index(x)

        h = ((h + x) \*\* 2) % 32

    print("h = ", h)

    V = h \*\* (Q - 2) % Q

    Z1 = (S \* V) % Q

    print("Z1 = ", Z1)

    Z2 = ((Q - R) \* V) % Q

    print("Z2 = ", Z2)

    U = ((A \*\* Z1 \* Y \*\* Z2) % P) % Q

    print("U = ", U)

    if U == R:

        print("U = ", U, "равен", "R = ", R)

        print('Цифровая подпись подтверждена')

        result = 'Цифровая подпись подтверждена'

    else:

        print("U = ", U, "не равен", "R = ", R)

        print('Цифровая подпись не подтверждена')

        result = 'Цифровая подпись не подтверждена'

    return result

def is\_prime\_factor(q,n):

    Ans = []

    d = 2

    while d \* d <= n:

        if n % d == 0:

            Ans.append(d)

            n //= d

        else:

            d += 1

    if n > 1:

        Ans.append(n)

    for i in Ans:

        if i == q:

            return True

    return False

# # Гост 94 подпись

def x(Q,P):

    res = []

    for i in range(2,P):

        if (i\*\*Q) % P == 1 :

            res.append(i)

    return res

print("ГОСТ 94")

P = int(input("P = "))

if is\_prime(P):

    Q = int(input("q = "))

    if is\_prime\_factor(Q, P-1):

        # print(x(Q,P))

        A = int(input("a = "))

        if (A\*\*Q) % P == 1 and (A != 1):

            X = int(input("x = "))

            if X < Q and X > 1:

                text = str(input("Вводите текст: "))

                res = decgost94(text, P, Q, A, X)

                checkgost94(res[-1],res[:-1])

            else:

                print("X не может быть больше Q")

        else:

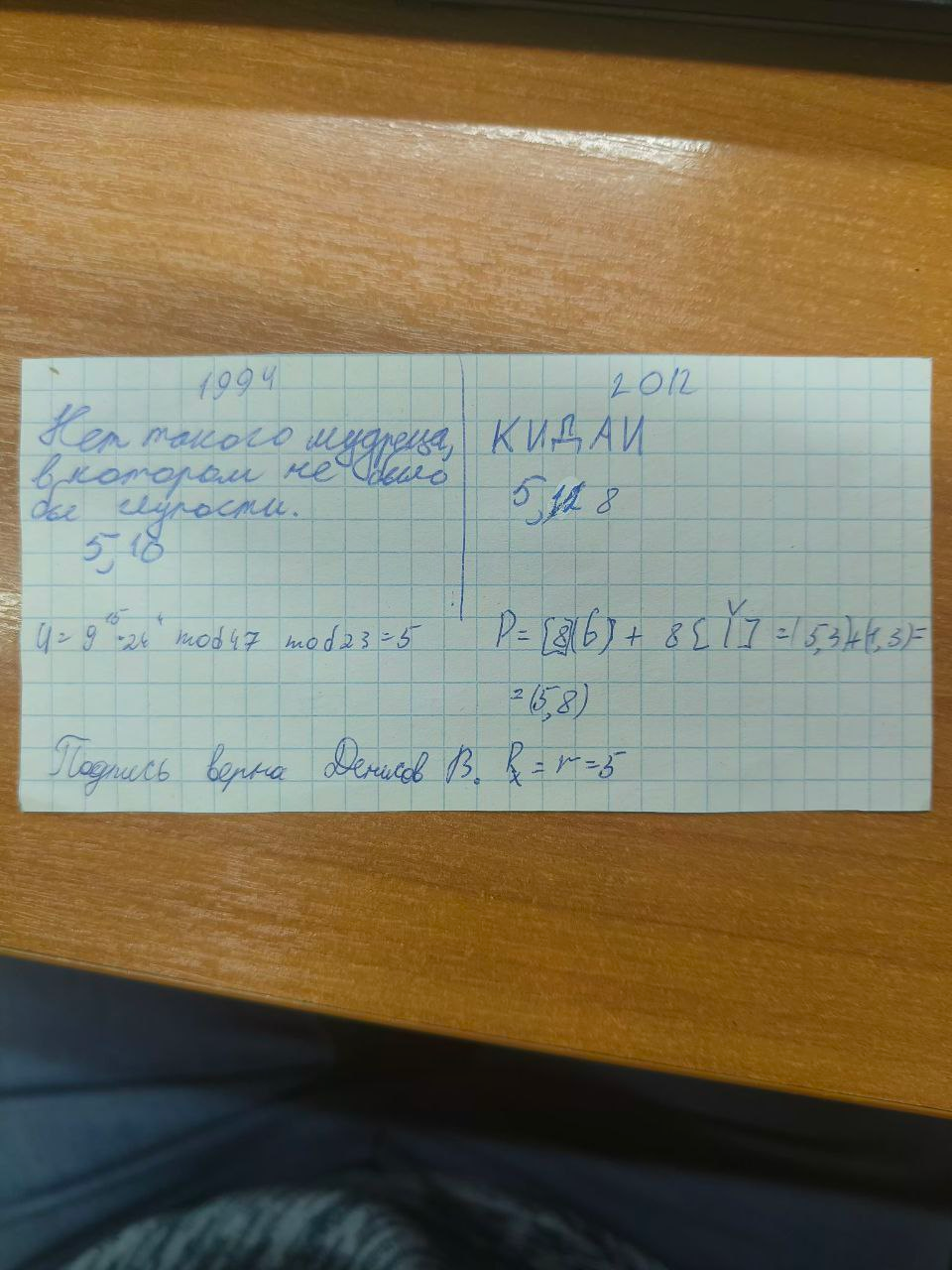
            print("A не удовлетворяет уровнения")

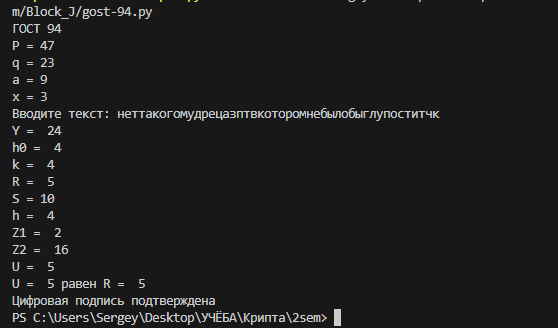
    else:

        print("q не простой сомножитель p-1")

else:

    print("Число P не простой")





ГОСТ Р 34.10-2012

Код:

import random

import math

def is\_prime(n):

    if n <= 1:

        return False

    if n <= 3:

        return True

    if n % 2 == 0 or n % 3 == 0:

        return False

    i = 5

    while i \* i <= n:

        if n % i == 0 or n % (i + 2) == 0:

            return False

        i += 6

    return True

def pred(s):

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

    s = s.lower().replace(' ', '')

    for sim in s:

        if sim not in llst:

            if sim == '.':

                s = s.replace('.', 'тчк')

            elif sim == ' ':

                s = s.replace(' ', 'пробел')

            elif sim == '?':

                s = s.replace('?', 'впрс')

            elif sim == '!':

                s = s.replace('!', 'всклзн')

            elif sim == ':':

                s = s.replace(':', 'двтч')

            elif sim == '–':

                s = s.replace('–', 'минус')

            elif sim == ',':

                s = s.replace(',', 'зпт')

            elif sim == '-':

                s = s.replace('-', 'тире')

            elif sim == 'ё':

                s = s.replace('ё', 'е')

            elif sim == ',':

                s = s.replace(',', 'зпт')

            elif sim == '-':

                s = s.replace('-', 'тире')

            elif sim == 'ё':

                s = s.replace('ё', 'е')

            elif sim == '0':

                s = s.replace('0', 'ноль')

            elif sim == '1':

                s = s.replace('1', 'один')

            elif sim == '2':

                s = s.replace('2', 'два')

            elif sim == '3':

                s = s.replace('3', 'три')

            elif sim == '4':

                s = s.replace('4', 'четыре')

            elif sim == '5':

                s = s.replace('5', 'пять')

            elif sim == '1':

                s = s.replace('6', 'шесть')

            elif sim == '2':

                s = s.replace('7', 'семь')

            elif sim == '3':

                s = s.replace('8', 'восемь')

            elif sim == '4':

                s = s.replace('9', 'девять')

            else:

                s = s.replace(sim, '')

    return s

llst = ['а', 'б', 'в', 'г', 'д', 'е', 'ё', 'ж', 'з', 'и', 'к', 'л', 'м', 'н', 'о', 'п', 'р', 'с', 'т', 'у', 'ф',

            'х', 'ц', 'ч', 'ш', 'щ', 'ъ', 'ы', 'ь', 'э', 'ю', 'я']

def hash(a):

    h = 0

    for x in a:

        x = llst.index(x)+1

        h = (h + x) \*\* 2 % 32

        # print(h)

    return h

mod1 = 11

def truemod(a):

    flag=False

    for i in range(1,10000):

        if (a\*i)%mod1==1:

            flag=True

            return i

    if flag==False:

        return 0

def solve\_R(k, g):

        x1, y1 = g[0], g[1]

        x2, y2 = x1, y1

        mod = 11

        for i in range(2,k+1):

            if x1 == x2 and y1 == y2:

                lambd = ((3 \* (x1 \*\* 2) + a) % mod \* truemod(2 \* y1)) % mod

                # if lambd==0:

                #     # print("0")

                #     break

                # else:

                x3 = (lambd \*\* 2 - 2 \* x1) % mod

                y3 = (lambd \* (x1 - x3) - y1) % mod

                # print("(" + str(x3) + ";" + str(y3)+")")

                x2 = x3

                y2 = y3

            else:

                lambd = (((y2 - y1) % mod) \* truemod(x2 - x1)) % mod

                if x2 - x1 == 0:

                    # print("0")

                    break

                x3 = (lambd \*\* 2 - x1 - x2) % mod

                y3 = (lambd \* (x1 - x3) - y1) % mod

                # print("(" + str(x3) + ";" + str(y3) + ")")

                x2 = x3

                y2 = y3

            # print("k= " + str(i))

        return x2, y2

def dota\_plus(dot1, dot2):

    x1, y1 = dot1

    x2, y2 = dot2

    mod = 11

    if x1 == x2 and y1 == y2:

        lambd = ((3 \* (x1 \*\* 2) + a) % mod \* truemod(2 \* y1)) % mod

        # if lambd==0:

        #     # print("0")

        #     break

        # else:

        x3 = (lambd \*\* 2 - 2 \* x1) % mod

        y3 = (lambd \* (x1 - x3) - y1) % mod

        # print("(" + str(x3) + ";" + str(y3)+")")

        x2 = x3

        y2 = y3

    else:

        lambd = (((y2 - y1) % mod) \* truemod(x2 - x1)) % mod

        if x2 - x1 == 0:

            return 0, 0

        x3 = (lambd \*\* 2 - x1 - x2) % mod

        y3 = (lambd \* (x1 - x3) - y1) % mod

        # print("(" + str(x3) + ";" + str(y3) + ")")

        x2 = x3

        y2 = y3

    return x3, y3

q = 13

x = 0

k = 0

text = input("Введите текст для подписи: ")

while True:

    x = int(input("Введите секрентый ключ x: "))

    if x >= q or x < 1:

        print("Х должен быть меньше q и больше 0")

    else:

        break

while True:

    k = int(input("Введите  k: "))

    if k >= q or k < 1:

        print("k должен быть меньше q и больше 0")

    else:

        break

print("Введите парамарты эллептической кривой:")

a = int(input("a = "))

b = int(input("b = "))

modul = int(input("modul = "))

open\_key = 0

G = (1, 8)

def decgost(x, mes, a, b, modul, k = 0):

    s = pred(mes)

    q=13

    m = hash(s)

    print(m)

    open\_key = solve\_R(x, G)

    print('Открытый ключ ', open\_key)

    # mod = 32# mod > длина алфавит

    x1, y1= solve\_R (k, G)

    P = [x1,y1]

    r = P[0] % q

    if r == 0:

        return "Неподходящее k"

    s = (k\*m + r \* x) % q

    return r, s, open\_key

def check\_gost(sign, open\_key, mes, G, mode):

    s = pred(mes)

    q=13

    # if mode == 1:

    #     print("Хэш при проверке 27")

    #     m = 16

    # else:

    #     m = hash(s)

    #     print("Хэш при проверке ", m)

    m = hash(s)

    print("Хэш при проверке ", m)

    r, s = sign

    print(r, s)

    if r > 0 and r < q and s > 0 and s < q:

        u1 = (s \* truemod(m)) % q

        u2 = (-r \* truemod(m)) % q

        print("u1 ", u1)

        print("u2 ", u2)

        P1 = solve\_R(u1, G)

        P2 = solve\_R(u2, open\_key)

        print("P1 ", P1)

        print("P2 ", P2)

        res = dota\_plus(P1, P2)

        print(res[0] % q)

        if res[0] % q == r:

            return "подпись верна ", res[0] % q

    return "Подпись неверна"

mode = int(input("1 - карточка, 2 - текст: "))

res\_gost = decgost(x, text, a, b, modul, k)

open\_key = res\_gost[2]

res\_gost = res\_gost[0], res\_gost[1]

print("Подпись ",res\_gost)

print(open\_key)

print(check\_gost(res\_gost, open\_key, text, G, mode))

