МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КУБГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**Отчет**

**по практическому заданию №6**

**по курсу**

**«КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ»**

Работу выполнил

Студенты 46 группы

Нагалевский А.М.

Преподаватель:

Крамаренко А.А.

Краснодар 2024

**Постановка задачи.**

В данной работе реализован алгоритм шифрования, основанный на использовании эллиптических кривых. Эллиптическая криптография является одним из самых мощных инструментов в области современной криптографии, обеспечивающим высокий уровень безопасности при относительно малых размерах ключей по сравнению с классическими методами.

Эллиптические кривые в криптографии представляют собой множество точек, удовлетворяющих уравнению вида в конечном поле. Основные операции, выполняемые с точками на кривой, включают сложение точек и умножение точек на число.

**Описание алгоритма**

Алгоритм шифрования на эллиптических кривых включает в себя следующие шаги:

1. Выбор параметров кривой и генерирующей точки G. В нашем случае использовалась кривая с параметрами и модулем , генерирующая точка .
2. Генерация пары ключей. Секретный ключ является случайным большим числом. Открытый ключ вычисляется как .
3. Шифрование сообщения. Каждый символ сообщения сопоставляется с точкой на кривой. Для шифрования символа используется пара точек , где — точка, соответствующая символу, k — случайное число для каждого символа.
4. Дешифрование сообщения. Получатель может восстановить исходную точку , используя свой секретный ключ nb.

**Реализация**

Реализация алгоритма включает определение функций для арифметических операций на эллиптических кривых (сложение точек, удвоение точек, умножение точки на число) и функций для шифрования/дешифрования сообщений. Также реализована функция для генерации пары ключей. Результаты шифрования и дешифрования приведены на рисунке 1.

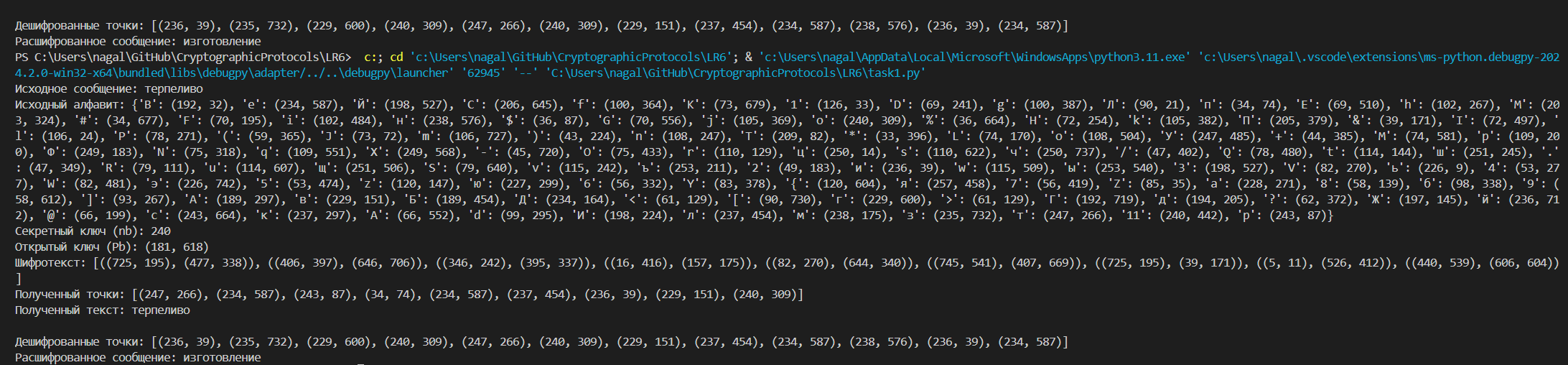


Рисунок 1 – результаты работы программы.

**Текст программы:**

**Файл LR6.py:**

import random

# task 1

# Параметры эллиптической кривой и модуля

a = -1

b = 1

p = 751

G = (0, 1)

# Модульная инверсия

def modinv(a, p):

    return pow(a, p - 2, p)

# Сложение двух точек P и Q на эллиптической кривой

def point\_add(P, Q):

    if P == (0, 0):

        return Q

    if Q == (0, 0):

        return P

    if P == Q:

        return point\_double(P)

    xp, yp = P

    xq, yq = Q

    if xp == xq and yp == p - yq:

        return (0, 0)

    lam = (yq - yp) \* modinv(xq - xp, p) % p

    x = (lam \*\* 2 - xp - xq) % p

    y = (lam \* (xp - x) - yp) % p

    return (x, y)

# Удвоение точки P на кривой

def point\_double(P):

    if P == (0, 0):

        return P

    xp, yp = P

    lam = (3 \* xp \*\* 2 + a) \* modinv(2 \* yp, p) % p

    x = (lam \*\* 2 - 2 \* xp) % p

    y = (lam \* (xp - x) - yp) % p

    return (x, y)

# Умножение точки P на скаляр n

def point\_multiply(P, n):

    R = (0, 0)

    N = P

    while n:

        if n & 1:

            R = point\_add(R, N)

        N = point\_double(N)

        n >>= 1

    return R

# Генерация ключей

def generate\_keys():

    nb = random.randint(1, p - 1)

    Pb = point\_multiply(G, nb)

    return nb, Pb

# Шифрование

def encrypt(Pm, k, Pb):

    C1 = point\_multiply(G, k)

    C2 = point\_add(Pm, point\_multiply(Pb, k))

    return C1, C2

# Функция для инвертирования точки

def point\_negate(P):

    x, y = P

    return (x, p - y) if y != 0 else P

# Дешифрование

def decrypt(C1, C2, nb):

    return point\_add(C2, point\_negate(point\_multiply(C1, nb)))

def decrypt\_to\_text(decrypted\_points, alphabet):

    # Инвертирование алфавита для поиска символов по точкам

    points\_to\_char = {point: char for char, point in alphabet.items()}

    # Сопоставление каждой точки с символом

    return ''.join(points\_to\_char.get(point, '?') for point in decrypted\_points)

alphabet = {

    'В': (67, 84), 'е': (99, 456), 'Й': (198, 527), 'С': (67, 667), 'f': (100, 364), 'К': (200, 30), '1': (33, 396), 'D': (69, 241), 'g': (100, 387), 'Л': (200, 721), 'п': (34, 74),

    'Е': (69, 510), 'h': (102, 267), 'M': (203, 324), '#': (34, 677), 'F': (70, 195), 'i': (102, 484), 'н': (203, 427), '$': (36, 87), 'G': (70, 556), 'j': (105, 369), 'о': (205, 372),

    '%': (36, 664), 'Н': (72, 254), 'k': (105, 382), 'П': (205, 379), '&': (39, 171), 'I': (72, 497), 'l': (106, 24), 'Р': (206, 106), '(': (39, 580), 'J': (73, 72), 'm': (106, 727),

    'С': (206, 645), ')': (43, 224), 'К': (73, 679), 'n': (108, 247), 'Т': (209, 82), '\*': (33, 396), 'L': (74, 170), 'o': (108, 504), 'У': (209, 669), '+': (44, 385), 'М': (74, 581),

    'p': (109, 200), 'Ф': (210, 31), '(': (44, 366), 'N': (75, 318), 'q': (109, 551), 'X': (210, 720), '-': (45, 720), 'О': (75, 433), 'r': (110, 129), 'ц': (215, 247), '(': (45, 31),

    'Р': (78, 271), 's': (110, 622), 'ч': (215, 504), '/': (47, 402), 'Q': (78, 480), 't': (114, 144), 'ш': (218, 150), '.': (47, 349), 'R': (79, 111), 'u': (114, 607), 'щ': (218, 601),

    '1': (48, 702), 'S': (79, 640), 'v': (115, 242), 'ъ': (221, 138), '2': (49, 183), 'и': (236, 39), 'w': (115, 509), 'ы': (221, 613), '3': (49, 568), 'V': (82, 270), 'X': (116, 92),

    'ь': (226, 9), '4': (53, 277), 'W': (82, 481), 'У': (116, 659), 'э': (226, 742), '5': (53, 474), 'X': (83, 373), 'z': (120, 147), 'ю': (227, 299), '6': (56, 332), 'Y': (83, 378),

    '{': (120, 604), 'я': (227, 452), '7': (56, 419), 'Z': (85, 35), '1': (125, 292), 'а': (228, 271), '8': (58, 139), '1': (85, 716), '1': (126, 33), 'б': (228, 480), '9': (58, 612),

    ']': (86, 726), 'A': (189, 297), 'в': (229, 151), '(': (59, 365), 'Л': (90, 21), 'Б': (189, 454), 'Д': (234, 164), 'е': (234, 587), '<': (61, 129), '[': (90, 730),

    'В': (192, 32), 'г': (229, 600), '>': (61, 129), ']': (93, 267), 'Г': (192, 719), 'д': (194, 205), '?': (62, 372), 'б': (98, 338), 'Ж': (197, 145), 'й': (236, 712),

    '@': (66, 199), 'с': (98, 413), '3': (197, 606), 'к': (237, 297), 'А': (66, 552), 'd': (99, 295), 'И': (198, 224), 'л': (237, 454), 'А': (66, 552), 'с': (99, 413),

    '3': (198, 527), 'м': (238, 175), 'с': (243, 664), 'ц': (250, 14), 'з': (235, 732), 'ы': (253, 540), 'н': (238, 576), 'т': (247, 266), 'ч': (250, 737), 'о': (240, 309), 'У': (247, 485),

    'ш': (251, 245), '11': (240, 442), 'Ф': (249, 183), 'щ': (251, 506), 'р': (243, 87), 'X': (249, 568), 'ъ': (253, 211), 'я': (257, 458)

}

message = 'терпеливо'

nb, Pb = generate\_keys()

k\_values = [random.randint(1, p - 1) for \_ in range(len(message))]

ciphertext = [encrypt(alphabet[char], k, Pb) for char, k in zip(message, k\_values)]

decrypted\_points = [decrypt(C1, C2, nb) for C1, C2 in ciphertext]

decrypted\_message = decrypt\_to\_text(decrypted\_points, alphabet)

print("Исходное сообщение:", message)

print("Исходный алфавит:", alphabet)

print("Секретный ключ (nb):", nb)

print("Открытый ключ (Pb):", Pb)

print("Шифротекст:", ciphertext)

print("Полученный точки:", decrypted\_points)

print("Полученный текст:", decrypted\_message)

# task2

# Обновленная генерирующая точка

G = (-1, 1)

# Секретный ключ

nb = 12

# Шифротекст

ciphertext = [

    ((16, 416), (128, 672)),

    ((56, 419), (59, 386)),

    ((425, 663), (106, 24)),

    ((568, 355), (145, 608)),

    ((188, 93), (279, 398)),

    ((425, 663), (99, 295)),

    ((179, 275), (269, 187)),

    ((188, 93), (395, 337)),

    ((188, 93), (311, 68)),

    ((135, 82), (556, 484)),

    ((56, 419), (106, 727)),

    ((16, 416), (307, 693))

]

# Дешифрование сообщения

decrypted\_points = [decrypt(C1, C2, nb) for C1, C2 in ciphertext]

print("Дешифрованные точки:", decrypted\_points)

print("Расшифрованное сообщение:", decrypt\_to\_text(decrypted\_points, alphabet))