ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет программной инженерии и компьютерной техники

Дисциплина «Информационная безопасность»

Лабораторная работа №2.5

«Шифрование открытого текста на основе эллиптических кривых»

Вариант: 4

Учебно-методическое пособие: Криптографические системы с секретным и открытым ключом: учебное пособие. / А.А. Ожиганов; УНИВЕРСИТЕТ ИТМО. — Санкт-Петербург, 2015

Автор: Калинин Даниил Дмитриевич

Группа: P34141

Преподаватель: Маркина Татьяна Анатольевна

г. Санкт-Петербург

2024

Содержание

Цель работы

Зашифровать открытый текст, используя алфавит, приведенный в [4], в подразделе «Задачи к лабораторным работам по криптографии на эллиптических кривых (используется кривая E751(-1,1) − и генерирующая точка G = (0, 1))».

Порядок выполнения работы

- Ознакомьтесь с теорией в учебном пособии «Криптография», а также в учебно-методическом пособии к выполнению лабораторного практикума по дисциплине «Криптография;
- Получите вариант задания у преподавателя;
- Зашифруйте открытый текст;
- Результаты и промежуточные вычисления оформить в виде отчета.

Вариант

№ варианта	**Открытый текст**	**Открытый ключ *B***	**Значения случайных чисел *k* для букв открытого текста**
4	симметрия	(179, 275)	11, 17, 18, 19, 16, 6, 12, 8, 2

Выполнение работы

Код

```
import csv
def read_alphabet():
    alphabet = []
    with open("../resources/alphabet.csv", encoding='utf-8') as r_file:
        file_reader = csv.reader(r_file, delimiter=",")
        count = 0
        for row in file_reader:
            if count != 0:
                alphabet.append([int(row[0]), row[1], int(row[2]), int(row[3])])
            count += 1
    return alphabet

def find_symbol_point_in_alphabet(alphabet, symbol):
    for row in alphabet:
        if row[1] == symbol:
            return row[2], row[3]
    print(f'Ошибка: символ {symbol} не найден в исходном алфовите')
    exit(1)

def sum_points_elliptic_curve(p, E, P, Q):
    # Вычисляем лямбда
    l = 0
    if P[0] == Q[0] and P[1] == Q[1]:
        l_top = (3*P[0]**2 + E[0]) % p
        l_bottom = (2*P[1]) % p
        for res in range(p+1):
            if (res == p):
                print("Ошибка: невозможно найти модуль от деления в процессе вычисления
lambda")
                exit(1)
            if ((l_bottom * res) % p) == (l_top % p):
                l = res
                break
    else:
        l_top = (Q[1] - P[1]) % p
        l_bottom = (Q[0] - P[0]) % p
        for res in range(p + 1):
            if (res == p):
                print("Ошибка: невозможно найти модуль от деления в процессе вычисления
lambda")
                exit(1)
            if ((l_bottom * res) % p) == (l_top % p):
                l = res
                break

    if (l % 1 == 0):
        l = int(l)
    else:
        print(f'Ошибка: lambda = {l}, не целое цисло!')
        exit(1)
    #print(f'lambda = {l}')

    # Вычисляем координаты
    x = (pow(l, 2, p) - P[0] - Q[0]) % p
    y = (l*(P[0] - x) - P[1]) % p
```

```
    return [x, y]

def calc_k_multiply_point(p, E, k, Point, point_name):
    iPoint = Point
    for i in range(2, k+1):
        iPoint = sum_points_elliptic_curve(p, E, iPoint, Point)
        #print(f'{i}{point_name} = {iPoint}')
    return iPoint


if __name__ == '__main__':
    # Константы
    p = 751
    E = [-1, 1]
    G = [0, 1]
    alphabet = read_alphabet()

    # Описание варианта
    open_message = "симметрия"
    Pb = [179, 275]
    k_array_A = [11, 17, 18, 19, 16, 6, 12, 8, 2]

    print("-- Исходные данные --")
    print(f'Открытый текст = {open_message}')
    print(f'Открытый ключ B = {Pb}')
    print(f'Значения случайных чисел k для букв открытого текста = {k_array_A}')
    print()

    # Проверяем корректность введенных данных
    if len(open_message) != len(k_array_A):
        print("Ошибка: длина массива k и открытого текста должна быть равной")
        exit(1)

    # Шифруем сообщение
    print("-- Шифрование --")
    close_message = []
    for i in range(len(open_message)):
        symbol = open_message[i]
        k = k_array_A[i]
        print(f'Шифруем символ \'{symbol}\', k = {k}')

        # Код символа из алфавита
        Pm = find_symbol_point_in_alphabet(alphabet, symbol)
        print(f'Pm = {Pm}')

        # Открытый ключ A
        kG = calc_k_multiply_point(p, E, k, G, "G")
        print(f'{k}G = {kG}')

        # Шифрование символа
        kPb = calc_k_multiply_point(p, E, k, Pb, "Pb")
        print(f'{k}Pb = {kPb}')

        Cm = sum_points_elliptic_curve(p, E, Pm, kPb)
        print(f'Cm = Pm + kPb = {Pm} + {kPb} = {Cm}')
        print()
        close_message.append((kG, Cm))

    print("-- Результат --")
    for i in range(len(open_message)):
        symbol = open_message[i]
        k = k_array_A[i]
```

```
        tmp = "{" + str(close_message[i][0]) + ", " + str(close_message[i][1]) + "}"
        print(f'Символ {symbol}, k = {k} ----> Cm = {tmp}')
```

Результаты работы программы

```
 2 -- Исходные данные --
 3 Открытый текст = симметрия
 4 Открытый ключ B = [179, 275]
 5 Значения случайных чисел k для букв открытого текста = [11, 17, 18, 19, 16, 6, 12, 8, 2]
 6
 7 -- Шифрование --
 8 Шифруем символ 'с', k = 11
 9 Pm = (243, 664)
10 11G = [179, 275]
11 11Pb = [1, 1]
12 Cm = Pm + kPb = (243, 664) + [1, 1] = [383, 411]
13
14 Шифруем символ 'и', k = 17
15 Pm = (236, 39)
16 17G = [440, 539]
17 17Pb = [425, 663]
18 Cm = Pm + kPb = (236, 39) + [425, 663] = [229, 151]
19
20 Шифруем символ 'м', k = 18
21 Pm = (238, 175)
22 18G = [618, 206]
23 18Pb = [72, 254]
24 Cm = Pm + kPb = (238, 175) + [72, 254] = [466, 214]
25
26 Шифруем символ 'м', k = 19
27 Pm = (238, 175)
28 19G = [568, 355]
29 19Pb = [82, 481]
30 Cm = Pm + kPb = (238, 175) + [82, 481] = [156, 704]
31
32 Шифруем символ 'е', k = 16
33 Pm = (234, 587)
34 16G = [72, 254]
35 16Pb = [725, 556]
36 Cm = Pm + kPb = (234, 587) + [725, 556] = [564, 38]
37
38 Шифруем символ 'т', k = 6
39 Pm = (247, 266)
40 6G = [725, 195]
41 6Pb = [74, 581]
42 Cm = Pm + kPb = (247, 266) + [74, 581] = [145, 143]
43
44 Шифруем символ 'р', k = 12
45 Pm = (243, 87)
46 12G = [286, 136]
47 12Pb = [116, 92]
48 Cm = Pm + kPb = (243, 87) + [116, 92] = [176, 413]
49
50 Шифруем символ 'и', k = 8
51 Pm = (236, 39)
52 8G = [346, 242]
53 8Pb = [56, 332]
54 Cm = Pm + kPb = (236, 39) + [56, 332] = [12, 314]
55
```

```
Шифруем символ 'я', k = 2
Pm = (257, 458)
2G = [188, 93]
2Pb = [702, 225]
Cm = Pm + kPb = (257, 458) + [702, 225] = [275, 456]

-- Результат --
Символ с, k = 11 ----> Cm = {[179, 275], [383, 411]}
Символ и, k = 17 ----> Cm = {[440, 539], [229, 151]}
Символ м, k = 18 ----> Cm = {[618, 206], [466, 214]}
Символ м, k = 19 ----> Cm = {[568, 355], [156, 704]}
Символ е, k = 16 ----> Cm = {[72, 254], [564, 38]}
Символ т, k = 6 ----> Cm = {[725, 195], [145, 143]}
Символ р, k = 12 ----> Cm = {[286, 136], [176, 413]}
Символ и, k = 8 ----> Cm = {[346, 242], [12, 314]}
Символ я, k = 2 ----> Cm = {[188, 93], [275, 456]}

Process finished with exit code 0

```

Вывод

В ходе лабораторной работы был зашифрован открытый текст, согласно алгоритму, приведенному в [4], в подразделе «Задачи к лабораторным работам по криптографии на эллиптических кривых (используется кривая E751(-1,1) − и генерирующая точка G = (0, 1))». Также в процессе работы был изучен метод шифрования текста на основе эллиптических кривых.