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

Факультет программной инженерии и компьютерной техники

Дисциплина «Информационная безопасность»

Лабораторная работа №2.6

«Расшифрование криптограммы 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

Дан шифртекст, используя алфавит, приведенный в [4], в подразделе «Задачи к лабораторным работам по криптографии на эллиптических кривых (используется кривая E_{751}(-1,1) − и генерирующая точка G = (0,1))» и зная секретный ключ n_b, найти открытый текст

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;
- Найдите открытый текст;
- Результаты и промежуточные вычисления оформите в виде отчета

Вариант

№ варианта	**Секретный ключ** n_b	**Шифртекст**
4	34	{(618, 206), (426, 662)}; {(72, 254), (67, 667)}; {(286, 136), (739, 574)}; {(16, 416), (143, 602)}; {(618, 206), (313, 203)}; {(618, 206), (114, 607)}; {(618, 206), (438, 711)}; {(188, 93), (573, 168)}

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_by_point_in_alphabet(alphabet, point):
    for row in alphabet:
        if int(row[2]) == point[0] and int(row[3]) == point[1]:
            return row[1]
    print(f'Ошибка: точка {point} не найдена 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 число!')
        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
    close_message_text = """
                    {(618, 206), (426, 662)}; {(72, 254), (67, 667)};
                    {(286, 136), (739, 574)}; {(16, 416), (143, 602)};
                    {(618, 206), (313, 203)}; {(618, 206), (114, 607)};
                    {(618, 206), (438, 711)}; {(188, 93), (573, 168)}
                    """
    nb = 34

    # Обработка шифртекста
    close_message = []
    for part in close_message_text.split(";"):
        part = part.strip().replace("{", "").replace("}","").replace("(","").replace(")","")
        part_split = part.split(", ")
        close_message.append(([int(part_split[0]), int(part_split[1])], [int(part_split[2]),
int(part_split[3])]))

    print("-- Исходные данные --")
    print(f'Шифртекст = {close_message}')
    print(f'Закрытый ключ B = {nb}')
    print()

    #Дешифрование сообщения
    print("-- Дешифрование --")
    open_message = ""
    Pm_array = []
    for kG, Cm in close_message:
        print(f'Дешифруем часть шифртекста {Cm}, kG = {kG}')

        nbkG = calc_k_multiply_point(p, E, nb, kG, "*kG")
        print(f'{nb}*kG = {nbkG}')

        nbkG[1] = p - nbkG[1]
        print(f'-{nb}*kG = {nbkG}')

        Pm = sum_points_elliptic_curve(p, E, Cm, nbkG)
        print(f'Pm = Cm - nb(kG) = {Cm} + {nbkG} = {Pm}')

        symbol = find_symbol_by_point_in_alphabet(alphabet, Pm)
        print(f'Символ {Pm} = \'{symbol}\'')

        open_message += symbol
        Pm_array.append(Pm)
        print()

    print("-- Результат --")
    for i in range(len(close_message)):
        kG = close_message[i][0]
        Cm = close_message[i][1]
        Pm = Pm_array[i]
        symbol = open_message[i]

        print(f'kG = {kG}, Cm = {Cm} ----> Pm = {Pm} -----> \'{symbol}\'')
    print(f"Дешифрованное сообщение = \'{open_message}\'")
```

Результаты работы программы

```
-- Исходные данные --
Шифртекст = [([618, 206], [426, 662]), ([72, 254], [67, 667]), ([286, 136], [739, 574]), ([16, 416], [143, 602]), ([618
, 206], [313, 203]), ([618, 206], [114, 607]), ([618, 206], [438, 711]), ([188, 93], [573, 168])]
Закрытый ключ B = 34

-- Дешифрование --
Дешифруем часть шифртекста [426, 662], kG = [618, 206]
34*kG = [74, 581]
-34*kG = [74, 170]
Pm = Cm - nb(kG) = [426, 662] + [74, 170] = [229, 151]
Символ [229, 151] = 'в'

Дешифруем часть шифртекста [67, 667], kG = [72, 254]
34*kG = [188, 658]
-34*kG = [188, 93]
Pm = Cm - nb(kG) = [67, 667] + [188, 93] = [235, 732]
Символ [235, 732] = 'з'

Дешифруем часть шифртекста [739, 574], kG = [286, 136]
34*kG = [175, 192]
-34*kG = [175, 559]
Pm = Cm - nb(kG) = [739, 574] + [175, 559] = [237, 454]
Символ [237, 454] = 'л'

Дешифруем часть шифртекста [143, 602], kG = [16, 416]
34*kG = [406, 397]
-34*kG = [406, 354]
Pm = Cm - nb(kG) = [143, 602] + [406, 354] = [240, 309]
Символ [240, 309] = 'о'

Дешифруем часть шифртекста [313, 203], kG = [618, 206]
34*kG = [74, 581]
-34*kG = [74, 170]
Pm = Cm - nb(kG) = [313, 203] + [74, 170] = [238, 175]
Символ [238, 175] = 'м'

Дешифруем часть шифртекста [114, 607], kG = [618, 206]
34*kG = [74, 581]
-34*kG = [74, 170]
Pm = Cm - nb(kG) = [114, 607] + [74, 170] = [251, 506]
Символ [251, 506] = 'щ'

Дешифруем часть шифртекста [438, 711], kG = [618, 206]
34*kG = [74, 581]
-34*kG = [74, 170]
Pm = Cm - nb(kG) = [438, 711] + [74, 170] = [236, 39]
Символ [236, 39] = 'и'

Дешифруем часть шифртекста [573, 168], kG = [188, 93]
34*kG = [591, 196]
-34*kG = [591, 555]
Pm = Cm - nb(kG) = [573, 168] + [591, 555] = [237, 297]
Символ [237, 297] = 'к'

-- Результат --
kG = [618, 206], Cm = [426, 662] ----> Pm = [229, 151] -----> 'в'
kG = [72, 254], Cm = [67, 667] ----> Pm = [235, 732] -----> 'з'
kG = [286, 136], Cm = [739, 574] ----> Pm = [237, 454] -----> 'л'
kG = [16, 416], Cm = [143, 602] ----> Pm = [240, 309] -----> 'о'
kG = [618, 206], Cm = [313, 203] ----> Pm = [238, 175] -----> 'м'
kG = [618, 206], Cm = [114, 607] ----> Pm = [251, 506] -----> 'щ'
kG = [618, 206], Cm = [438, 711] ----> Pm = [236, 39] -----> 'и'
kG = [188, 93], Cm = [573, 168] ----> Pm = [237, 297] -----> 'к'
Дешифрованное сообщение = 'взломщик'

```

Вывод

В ходе лабораторной работы был дешифрован открытый текст по известному шифртексту и секретному ключу n_b согласно алфавиту, приведенному в [4], в подразделе «Задачи к лабораторным работам по криптографии на эллиптических кривых (используется кривая $E_{751}(-1,1) -$ и генерирующая точка G = (0, 1))». Также в процессе работы был изучен метод дешифрования открытого текста из шифртекста и закрытого ключа на основе эллиптических кривых.