Отчет по лабораторной работе №7

Дисциплина: Математические основы защиты информации и информационной безопасности

Выполнила Дяченко Злата Константиновна, НПМмд-02-22

Содержание

List of Figures

1 Цель работы

Ознакомится и реализовать алгоритм, реализующий p-метод Полларда для задач дискретного логарифмирования.

2 Задание

Реализовать программно алгоритм, реализующий p-метод Полларда для задач дискретного логарифмирования.

3 Теоретическое введение

Алгоритм, реализующий p-метод Полларда. *Вход*. Простое число p,число a порядка r по модулю p, целое число b, 1<b<p; отображение f обладающее сжимающими свойствами и сохраняющее вычислимость логарифма. *Выход*. Показатель x для которого $a^x \equiv b(modp)$ если такой показатель существует. 1. Выбрать произвольные целые числа u, v и положить $c \leftarrow a^u b^v(modp), d \leftarrow c$. 2. Выполнять $c \leftarrow f(c)(modp), d \leftarrow f(f(d))(modp)$, вычисляя при этом логарифмы для c и d как линейные функции от x по модулю r, до получения $c \equiv d(modp)$. 3. Приравняв логарифмы для c и d, вычислить логарифм x решением сравнения по модулю r. Результат: x или “Решений нет”

4 Выполнение лабораторной работы

4.1 Шаг 1

Ознакомилась с предоставленными теоретическими данными. Для выполнения задания решила использовать язык Python. Написала функцию, реализующую 1-2 шаг алгоритма p-метода Полларда для задач дискретного логарифмирования. Код функции и результат ее использования представлен на Рисунке 1 (рис. - fig. 4.1) и Рисунке 2 (рис. - fig. 4.2). Функция принимает на вход число a, b, p, u, v и число c. Пример работы алгоритма для числа из представленых для лабораторной работы материалов также представлен на рисунке.

```
In [64]: def polard (a, b, p, u, v):
             log_c=[]
             log_d=[]
             c=a**u*b**v%p
             c_107=a**u*b**v
             d=c
             d_107=c_107
             log_c.append([u, v])
             log_d.append([u, v])
             u_c=u
             u_d=u
             v_c=v
             v_d=v
             k=0
             i=0
             while (i==0):
                 if (c<53):
                     u_c+=1
                     c=(c*a)%p
                     log_c.append([u_c, v_c])
                 else:
                     v_c+=1
                     c=(c*b)%p
                     log_c.append([u_c, v_c])
                 if (d<53):
                     u_d+=1
                     d=(d*a)%p
                     if (d<53):
                         u_d+=1
                         d=(d*a)%p
                         log_d.append([u_d, v_d])
                     else:
                         v_d+=1
                         d=(d*b)%p
                         log_d.append([u_d, v_d])
```

Figure 4.1: Реализация шагов 1 и 2 алгоритма, реализующего p-метод Полларда для задач дискретного логарифмирования

```
        else:
            v_d+=1
            d=(d*b)%p
            if (d<53):
                u_d+=1
                d=(d*a)%p
                log_d.append([u_d, v_d])
            else:
                v_d+=1
                d=(d*b)%p
                log_d.append([u_d, v_d])
        k+=1
        print(c, log_c[k], d, log_d[k])
        if (c==d):
            return(log_c[k], log_d[k])
```

```
In [65]: log_c, log_d= polard(10, 64, 107, 2, 2)
```

```
40 [3, 2] 79 [4, 2]
79 [4, 2] 56 [5, 3]
27 [4, 3] 75 [5, 5]
56 [5, 3] 3 [5, 7]
53 [5, 4] 86 [7, 7]
75 [5, 5] 42 [8, 8]
92 [5, 6] 23 [9, 9]
3 [5, 7] 53 [11, 9]
30 [6, 7] 92 [11, 11]
86 [7, 7] 30 [12, 12]
47 [7, 8] 47 [13, 13]
```

Figure 4.2: Реализация шагов 1 и 2 алгоритма, реализующего p-метод Полларда для задач дискретного логарифмирования

4.2 Шаг 2

Реализовала 3 шаг алгоритма, написав функцию *poisk*. Код функции и результат ее использования представлен на Рисунке 3 (рис. - fig. 4.3). Полученный ответ совпадает с ответом, представленным в теоретических материалах.

```
In [66]: log_c
Out[66]: [7, 8]

In [67]: log_d
Out[67]: [13, 13]

In [109]: koef_1=log_c[0]-log_d[0]

In [110]: koef_2=log_c[1]-log_d[1]

In [111]: def poisk(c, k, dell):
              for i in range(1, 100):
                  b=dell*i
                  for j in range (1, 100):
                      a=c+k*j
                      if (a==b):
                          return j
                  b=dell*(-i)
                  for j in range (1, 100):
                      a=c+k*j
                      if (a==b):
                          return j

In [112]: q=poisk(koef_1, koef_2, 53)

In [113]: q
Out[113]: 20
```

Figure 4.3: Реализация 3 шага алгоритма

5 Выводы

Я ознакомилась с алгоритмом, реализующем p-метод Полларда для задач дискретного логарифмирования, и реализовала его программно. Результаты работы находятся в репозитории на GitHub, а также есть скринкаст выполнения лабораторной работы.