РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ
Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

Отчёт по лабораторной работе №7

Дисциплина: *Математические основы защиты информации и информационной безопасности*

Студент: Майорова О.А., НФИмд-02-21
Преподаватель: д.ф.-м.н. Кулябов Д.С.

Москва 2021

Содержание

List of Figures

List of Tables

1 Цель работы

Цель: Ознакомиться с задачей дискретного логарифмирования в конечном поле.

2 Задание

Программно реализовать ρ-метод Полларда для задач дискретного логарифмирования.

3 Теоретическое введение

Дискретное логарифмирование (DLOG) — задача обращения функции g^x в некоторой конечной мультипликативной группе G. Наиболее часто задачу дискретного логарифмирования рассматривают в мультипликативной группе кольца вычетов или конечного поля, а также в группе точек эллиптической кривой над конечным полем. Эффективные алгоритмы для решения задачи дискретного логарифмирования в общем случае неизвестны. Для заданных g и a решение x уравнения $g^x = a$ называется дискретным логарифмом элемента a по основанию g. В случае, когда G является мультипликативной группой кольца вычетов по модулю m, решение называют также индексом числа a по основанию g. Индекс числа a по основанию g гарантированно существует, если g является первообразным корнем по модулю m. Решение задачи дискретного логарифмирования состоит в нахождении некоторого целого неотрицательного числа x, удовлетворяющего уравнению $g^x = a$. Если оно разрешимо, у него должно быть хотя бы одно натуральное решение, не превышающее порядок группы. Это сразу даёт грубую оценку сложности алгоритма поиска решений сверху — алгоритм полного перебора нашёл бы решение за число шагов не выше порядка данной группы. В кольце вычетов по простому модулю одним из алгоритмов решения задачи с экспоненциальной сложностью является ρ-метод Полларда [1].

На вход алгоритму подаются ростое число p, число a порядка r по модулю p, целое число $b : 1 < b < p$ и f - отображение, обл-ее сжимающими св-ми и сохраняющее вычислимость логарифма. Для дискретного логарифмирования в качестве случайного отображения f чаще всего используются ветвящиеся отображения, например:

$$f(c) = \begin{cases} ac & c < p/2 \\ bc & c > p/2 \end{cases}$$

При $c < p/2$ имеем $\log_a f(c) = \log_a c + 1$, при $c > p/2$ имеем $\log_a f(c) = \log_a c + x$.

1. Выбрать произвольные целые числа u, v и положить $c \leftarrow a^u b^v (\mod p), d \leftarrow c$.

2. Выполнять $c \leftarrow f(c)(\mod p), d \leftarrow f(f(d))(\mod p)$, вычисляя при этом логарифмы для c и d как линейные функции от x по модулю r, до получения равенства $c \equiv d(\mod p)$.

3. Приравняв логарифмы для c и d, вычислить логарифм x решением сравнения по модулю r. Результат: x или “Решений нет”.

Задача дискретного логарифмирования, как и задача разложения на множители, применяется во многих алгоритмах криптографии с открытым ключом. Предложенная в 1976 году У. Диффи и М. Хеллманом для установления сеансового ключа, эта задача послужила основой для создания протоколов шифрования и цифровой подписи, доказательств с нулевым разглашением и других криптографических протоколов [2].

4 Выполнение лабораторной работы

Для выполнения лабораторной работы был выбран язык Python. Далее реализуем представленный алгоритм в виде функции в соответствии с описанием из задания к лабораторной работе.

Сначала реализуем функцию вычисления ветвящегося отображения и этом логарифмов для c и d:

```
def f(c, u, v):
    if c < 53:
        return 10*c % 107, u+1, v
    else:
        return 64*c % 107, u, v+1
```

Также, для вычисления наибольшего общего делителя, используем функцию, реализующую расширенный алгоритм Евклида, из лабораторной работы 4:

```
def ExtEuclid(a, b):
    rp = a
    rc = b
    xp, xc = 1, 0
    yp, yc = 0, 1
    rn = rp % rc
    d = rc
    while rn != 0:
        rn = rp % rc
```

```
        q = (rp - rn)/rc
        d, x, y = rc, xc, yc

        rp = rc
        rc = rn

        xc = xp - q*xc
        xp = x

        yc = yp - q*yc
        yp = y

    return d, x, y
```

Наконец, реализуем ρ-метод Полларда для задач дискретного логарифмирования:

```
def PollardLog(p, a, r, b, u, v):
    c = a**u * b**v % p
    d = c
    uc, vc = u, v
    ud, vd = u, v

    c, uc, vc = f(c, uc, vc)
    c %= p
    d, ud, vd = f(*f(d, ud, vd))
    d %= p

    while c%p != d%p:
        c, uc, vc = f(c, uc, vc)
```

```
        c %= p
        d, ud, vd = f(*f(d, ud, vd))
        d %= p

    v = vc - vd
    u = ud - uc

    d, x, y = ExtEuclid(v, r)

    while d != 1:
        v /= d
        u /= d
        r /= d
        d, x, y = ExtEuclid(v, r)

    return x*u % r
```

Результатом запуска функции будет рис. 4.1.

```
PollardLog(107, 10, 53, 64, 2, 2)
```

```
20.0
```

Figure 4.1: Проверка функции

Можно видеть, что был получен верный результат, и функция работает корректно.

5 Выводы

Таким образом, была достигнута цель, поставленная в начале лабораторной работы. Было осуществлено знакомство с задачей дискретного логарифмирования в конечном поле. Также была получена реализация на языке Python ρ-метода Полларда для задач дискретного логарифмирования.

Список литературы

1. Дискретное логарифмирование [Электронный ресурс]. Википедия: Свободная энциклопедия, 2021. URL: https://ru.wikipedia.org/w/index.php?title=Дискретное_логарифмирование&oldid=118793513.
2. Бубнов С.А. Лабораторный практикум по основам криптографии. 2012.