Дискретное логарифмирование в конечном поле

Дисциплина: Математические основы защиты информации и информационной безопасности

Студент: Гонсалес Ананина Луис Антонио, 1032175329
Группа: НФИмд-02-21
Преподаватель: д-р.ф.-м.н., проф. Кулябов Дмитрий Сергеевич

25 декабря, 2021, Москва

Цели и задачи

Цель лабораторной работы

Цель данной лабораторной работы- изучить теорию и реализовать рассмотренный алгоритм программно.

Выполнение лабораторной работы

Дискретное логарифмирование в конечном поле

Пусть p — небольшое простое число, п> l,q = p’ GF(q) — конечное поле из q элементов, g — примитивный элемент поля GF(q), h е GF(q) Требуется найти решение уравнения g* = h относительно x при известных g и h. Решение данной задачи существенно зависит от способа представления элементов поля. В данном параграфе мы познакомимся с алгоритмами логарифмирования в GF(q), использующими изученные в курсе алгебры способы задания конечных полей.

Поле GF(q) может быть задано в виде GF(p)[y]/f(y), где f(y) — неприводимый многочлен над GF(p) степени п (см. [ГЕН2, утверждение 17, с. 181]). Поэтому можно считать, что поле GF(q) состоит из многочленов над GF(p) степени не более п -

Алгоритм полларда для дискретного логарифмирования

р-метод Полларда для дискретного логарифмирования (р-метод) — алгоритм дискретного логарифмирования в кольце вычетов по простому модулю, имеющий экспоненциальную сложность. Предложен британским математиком Джоном Поллардом (англ. John Pollard) в 1978 году, основные идеи алгоритма очень похожи на идеи ро-алгоритма Полларда для факторизации чисел. Данный метод рассматривается для группы ненулевых вычетов по модулю p, где p — простое число, большее 3.

Для заданного простого числа p и двух целых чисел a и b требуется найти целое число x, удовлетворяющее сравнению:

Результат выполнения работы 1

```
In [1]: def exteuclid(a, b):
            if b == 0:
                return a, 1, 0
            else:
                d, xx, yy = exteuclid(b, a % b)
                x = yy
                y = xx - (a // b) * yy
                return d, x, y


        def inverse(a, n):
            return exteuclid(a, n)[1]
```

```
In [2]: def xab(x, a, b, xxx_change):

            (G, H, P, Q) = xxx_change
            sub = x % 3 # Subsets

            if sub == 0:
                x = x*xxx_change[0] % xxx_change[2]
                a = (a+1) % Q

            if sub == 1:
                x = x * xxx_change[1] % xxx_change[2]
                b = (b + 1) % xxx_change[2]

            if sub == 2:
                x = x*x % xxx_change[2]
                a = a*2 % xxx_change[3]
                b = b*2 % xxx_change[3]

            return x, a, b
```

Figure 2: Выполнение1

Результат выполнения работы 2

```
In [3]: def pollard(G, H, P):


            Q = int((P - 1) // 2)


            x = G*H
            a = 1
            b = 1

            X = x
            A = a
            B = b

            for i in range(1, P):

                x, a, b = xab(x, a, b, (G, H, P, Q))

                X, A, B = xab(X, A, B, (G, H, P, Q))
                X, A, B = xab(X, A, B, (G, H, P, Q))

                if x == X:
                    break


            nom = a-A
            denom = B-b


            res = (inverse(denom, Q) * nom) % Q

            if verify(G, H, P, res):
                return res

            return res + Q
```

Figure 3: Выполнение2

Результат выполнения работы 3

```
In [4]: def verify(g, h, p, x):
            return pow(g, x, p) == h

        args = [
            (10, 64, 107),
        ]

        for arg in args:
            res = pollard(*arg)
            print(arg, ': ', res)
            print("Validates: ", verify(arg[0], arg[1], arg[2], res))
            print()

(10, 64, 107) :  20
Validates:  True
```

Figure 4: Выполнение3

Выводы

В ходе данной лабораторной работы была изучена теория и реализован рассмотренный алгоритм программно.