Лабораторная работа №7

Математические основы защиты информации и информационной безопасности

Минов К.В, НПМмд-02-239

декабря 2023

Российский университет дружбы народов

Москва, Россия

Цель лабораторной работы

Реализовать на языке программирования p-метод Полларда для дискретного логарифмирования

Обозначим $F_p = Z/pZ$, p - простое целое число и назовем конечным полем из p элементов. Задача дискретного логарифмирования в конечном поле F_p формулируется так: для данных целых чисел a и b, $a > 1, b > p$, найти логарифм - такое целое число x, что $a^x \equiv b (mod\ p)$ (если такое число существует). По аналогии с вещественными числами используется обозначение $x = log_a b$.

Теоретическое введение

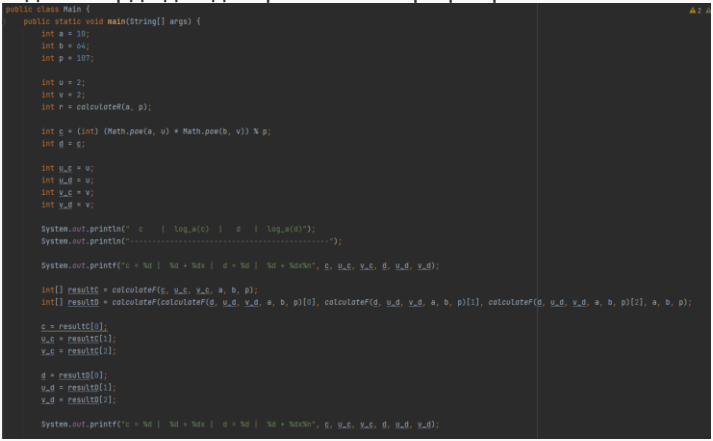
Безопасность соответствующих криптосистем основана на том, что зная числа a, x, p вычислить $a^x \pmod p$ легко, а решить задачу дискретного логарифмирования трудно. Рассмотрим p-метод Полларда, который можно применить и для задач дискретного логарифмирования. При этом случайное отображение f должно обладать не только сжимающими свойствами, но и вычислимостью логарифма (логарифм числа $f(c)$ можно выразить через неизвестный логарифм x и $\log_a f(c)$). Для дискретного логарифмирования в качестве случайного отображения f чаще всего используются ветвящиеся отображения

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

При $c < \frac{p}{2}$: $\log_a f(c) = \log_a c + 1$, при $c > \frac{p}{2}$: $\log_a f(c) = \log_a c + x$.

Ход выполнения лабораторной работы

- Реализуем p-метод Полларда для дискретного логарифмирования



```
public class Main {
    public static void main(String[] args) {
        int a = 10;
        int b = 64;
        int p = 107;

        int u = 2;
        int v = 2;
        int r = calculateR(a, p);

        int c = (int) (Math.pow(a, u) * Math.pow(b, v)) % p;
        int d = c;

        int u_c = u;
        int u_d = u;
        int v_c = v;
        int v_d = v;

        System.out.println("  c    |  log_a(c)  |   d    |  log_a(d)");
        System.out.println("-------------------------------------------------");

        System.out.printf("c = %d |  %d + %dx |  d = %d |  %d + %dx%n", c, u_c, v_c, d, u_d, v_d);

        int[] resultC = calculateF(c, u_c, v_c, a, b, p);
        int[] resultD = calculateF(calculateF(d, u_d, v_d, a, b, p)[0], calculateF(d, u_d, v_d, a, b, p)[1], calculateF(d, u_d, v_d, a, b, p)[2], a, b, p);

        c = resultC[0];
        u_c = resultC[1];
        v_c = resultC[2];

        d = resultD[0];
        u_d = resultD[1];
        v_d = resultD[2];

        System.out.printf("c = %d |  %d + %dx |  d = %d |  %d + %dx%n", c, u_c, v_c, d, u_d, v_d);
```

Figure 1: Рис.1: p-метод Полларда для дискретного логарифмирования

Ход выполнения лабораторной работы

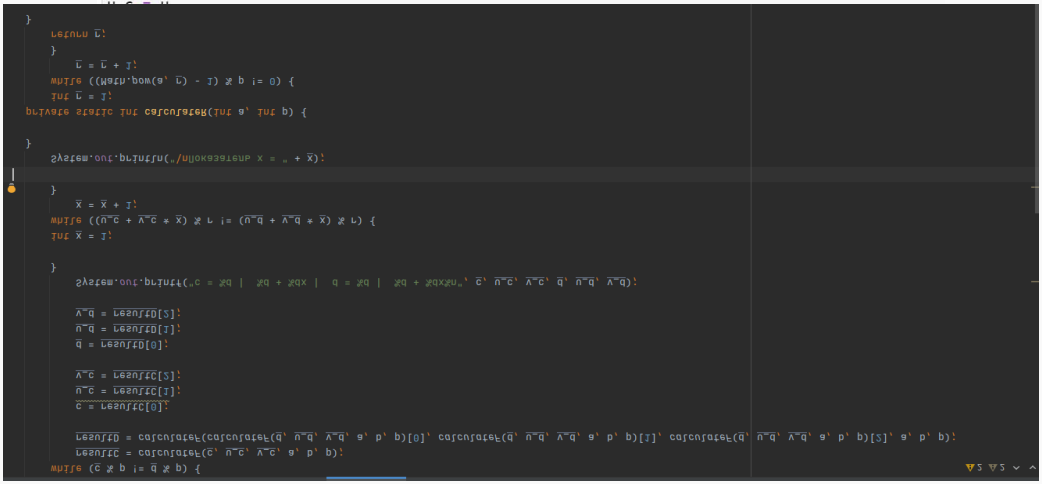


Figure 2: Рис.2: p-метод Полларда для дискретного логарифмирования

Ход выполнения лабораторной работы

```
    private static int calculateR(int a, int p) {
        int r = 1;
        while ((Math.pow(a, r) - 1) % p != 0) {
            r = r + 1;
        }
        return r;
    }

    private static int[] calculateF(int x, int u, int v, int a, int b, int p) {
        int[] result = new int[3];
        int r = calculateR(a, p);

        if (x < r) {
            result[0] = (int) ((a * x) % p);
            result[1] = u + 1;
            result[2] = v;
        } else {
            result[0] = (int) ((b * x) % p);
            result[1] = u;
            result[2] = v + 1;
        }

        return result;
    }
}
```

Figure 3: Рис.3: p-метод Полларда для дискретного логарифмирования

Вывод

- В ходе выполнения данной лабораторной работы был реализован p-метод Полларда для дискретного логарифмирования