ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

ЛАБОРАТОРНАЯ РАБОТА № 2.5

ПО ДИСЦИПЛИНЕ «ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ»

Шифрование открытого текста на основе эллиптических кривых

Выполнил: Давыдов Иван Денисович

Группа: Р3400

Вариант 5

Санкт-Петербург
2020/2021

Цель работы

Зашифровать открытый текст, используя алфавит, приведенный в задании к лабораторной (используется кривая $E_{751}(-1, 1)$: $y^2 = x^3 - 1x + 1 \pmod{751}$ и генерирующая точки G(0,1)).

Задание

№ варианта	**Открытый текст**	**Открытый ключ *B***	**Значения случайных чисел *k* для букв открытого текста**
5	уверовать	(425, 663)	6, 14, 5, 7, 12, 11, 4, 9, 19

Листинг

Point.java

```
package core;

public class Point {
    private int x;
    private int y;


    Point(int x, int y){
        this.x = x;
        this.y = y;
    }

    int getX() {
        return x;
    }

    int getY() {
        return y;
    }

    @Override
    public boolean equals(Object obj){
        if (this == obj)
            return true;
        if (obj == null || getClass() != obj.getClass())
            return false;

        Point p = ((Point) obj);
        return this.x == p.getX() && this.y == p.getY();
    }

    @Override
    public String toString() {
        return "(" + x + ", " + y + ")";
    }
}
```

EllepticCurve.java

```
package core;

class EllepticCurve {
    private int a, b, p;

    EllepticCurve(int a, int b, int p){
        this.a = a;
        this.b = b;
        this.p = p;
    }

    // Умножаем по быстрому алгоритму удвоения-сложения
    // 41 == 0b101001 => 41P = 2^5P + 2^3P + 2^0P
    Point mul(Point p, int n){
        int tmp = 1, power = -1;
        Point resP = null, tmpP = new Point(p.getX(), p.getY());;

        while (tmp <= n) {
            power++;
            tmp = tmp << 1;
        }

        tmp = n;
        boolean first = true;
        for (int i = 0; i <= power; i++){
            if((tmp & 1) == 1) {
                if(first) {
                    resP = new Point(tmpP.getX(), tmpP.getY());
                    first = false;
                }
                else
                    resP = this.sum(resP, tmpP);

            }

            tmpP = this.sum(tmpP, tmpP);
            tmp = tmp >>> 1;
        }

        return resP;
    }

    // Метод расчета суммы двух точек на эллиптической кривой
    Point sum(Point p1, Point p2) {

        /*
            x3 = λ^2 - x1 - x2(mod p)
            y3 = λ(x1 - x3) - y1(mod p)
        */
        int lambda = calcLambda(p1, p2);
        int tmp = lambda * lambda - p1.getX() - p2.getX();
        int newX = tmp >= 0 ? tmp % this.p : this.p + (tmp % this.p);

        tmp = lambda * (p1.getX() -newX) - p1.getY();
        int newY = tmp >= 0 ? tmp % this.p : this.p + (tmp % this.p) ;

        Point res = new Point(newX, newY);


        return new Point(newX, newY);
    }
```

```
    // метод расчета наклона прямой, проходящей через две точки
    private int calcLambda(Point p1, Point p2 ) {
        int numerator, denominator;

        // p1 == p2: λ = (3x1^2 + a) / 2y1
        // p1 != p2: λ = (y2 - y1) / (x2 - x1)
        if (p1.equals(p2)) {
            numerator = 3 * p1.getX() * p1.getX() + this.a;
            denominator = 2 * p1.getY();
        }
        else {
            numerator = p2.getY() - p1.getY();
            denominator = p2.getX() - p1.getX();
        }

        // a / b = a * b^(-1); Ищем обратную величину по модулю
        denominator = this.invMod(denominator);

        // если вычисленное значение получится отрицательным, приводим к положительному
        return (numerator * denominator < 0) ?
                this.p + (numerator * denominator) % this.p :
                (numerator * denominator) % this.p;
    }


    //Возвращает обратную величину n по модулю p
    //такое целое число m, при котором (n*m)%p == 1
    //Применяется расширенный алгоритм Евклида
    private int invMod(int n){
        int s = 0, oldS = 1;
        int r = this.p, oldR = n;

        while (r != 0){
            int quotient = (int)Math.floorDiv(oldR , r);
            int tmp = r;
            r = oldR - (quotient * r);
            oldR = tmp;

            tmp = s;
            s = oldS - quotient * s;
            oldS = tmp;

        }

        //учитываем отрицательный результат
        return oldS + (oldS < 0 ? this.p : 0);
    }
}
```

Main.java

```
package core;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

public class Main {

    public static void main(String[] args) {
        EllepticCurve curve;
        String text;
        Point pB, g;
        List<Integer> ks;

        // считываем алфавит и вариант из файлов
       HashMap<String, Point> alphabet = TaskData.getAlphabet();
        HashMap<String, Object> var = TaskData.getVar();


        text = (String) var.get("Text");
        pB = new Point(
                Integer.parseInt((String)var.get("Bx")),
                Integer.parseInt((String)var.get("By"))
        );

        ks = (ArrayList) var.get("k");

        curve = new EllepticCurve(-1, 1, 751); // E(-1, 1) mod 751
        g = new Point(0, 1); // G = (0, 1)

        // шифруем текст
        List<Point> result = new ArrayList<>();

        for(int i = 0; i < text.length(); i++){
            Point pm = alphabet.get(text.charAt(i)+"");
            int char_k = ks.get(i);

            // Cm = {kG, Pm + kPb}
            Point kG = curve.mul(g,char_k);
            Point kPb = curve.mul(pB,char_k);
            Point pmkpb = curve.sum(kPb, pm); // Pm + kPb

            System.out.println("-----------------------------------");
            System.out.println(text.charAt(i)+": k= "+ char_k+"; kG= "+kG.toString()+"; Pm=
"+pm.toString()+"; kPb= "+ kPb.toString());
            System.out.println("Cm = (kg, Pm+kpB) = "+kG.toString()+" "+pmkpb.toString());

            result.add(kG);
            result.add(pmkpb);
        }


        System.out.println("\nResult: ");
        for(int i = 0; i < result.size(); i=i+2) {
            System.out.print(text.charAt(i/2)+": [");
            for (int j = 0; j < 2; j++)
                System.out.print(result.get(i+j).toString());
            System.out.println("];");
        }
    }
}
```

Результаты

```
у: k= 6; kG= (725, 195); Pm= (247, 485); kPb= (1, 1)
Cm = (kg, Pm+kpB) = (725, 195) (620, 680)
-----------------------------------
в: k= 14; kG= (596, 433); Pm= (229, 151); kPb= (455, 368)
Cm = (kg, Pm+kpB) = (596, 433) (39, 171)
-----------------------------------
е: k= 5; kG= (425, 663); Pm= (234, 587); kPb= (74, 170)
Cm = (kg, Pm+kpB) = (425, 663) (654, 102)
-----------------------------------
р: k= 7; kG= (135, 82); Pm= (243, 87); kPb= (384, 276)
Cm = (kg, Pm+kpB) = (135, 82) (85, 716)
-----------------------------------
о: k= 12; kG= (286, 136); Pm= (240, 309); kPb= (750, 1)
Cm = (kg, Pm+kpB) = (286, 136) (99, 295)
-----------------------------------
в: k= 11; kG= (179, 275); Pm= (229, 151); kPb= (499, 595)
Cm = (kg, Pm+kpB) = (179, 275) (526, 412)
-----------------------------------
а: k= 4; kG= (16, 416); Pm= (228, 271); kPb= (657, 285)
Cm = (kg, Pm+kpB) = (16, 416) (458, 490)
-----------------------------------
т: k= 9; kG= (489, 468); Pm= (247, 266); kPb= (406, 397)
Cm = (kg, Pm+kpB) = (489, 468) (140, 115)
-----------------------------------
ь: k= 19; kG= (568, 355); Pm= (256, 121); kPb= (16, 416)
Cm = (kg, Pm+kpB) = (568, 355) (400, 56)
-----------------------------------

Result:
у: [(725, 195)(620, 680)];
в: [(596, 433)(39, 171)];
е: [(425, 663)(654, 102)];
р: [(135, 82)(85, 716)];
о: [(286, 136)(99, 295)];
в: [(179, 275)(526, 412)];
а: [(16, 416)(458, 490)];
т: [(489, 468)(140, 115)];
ь: [(568, 355)(400, 56)];
```

Выводы

В ходе данной лабораторной работы был изучен алгоритм шифрования на основании эллиптических кривых. Была написана программа, реализующая этот алгоритм и позволяющая зашифровать открытый текст, используя приведенный в задании алфавит на основе кривой $E_{751}(-1, 1)$: $y^2 = x^3 – 1x + 1 \pmod{751}$ и генерирующей точки G(0,1).