МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

ЛАБОРАТОРНАЯ РАБОТА №2.6

“Расшифрование криптограммы на основе эллиптических кривых”

по дисциплине

“Информационная безопасность”

Студент:

Алексеев Даниил Иннокентьевич

Группа P34302

Преподаватель:

Рыбаков Степан Дмитриевич

Санкт-Петербург

2023

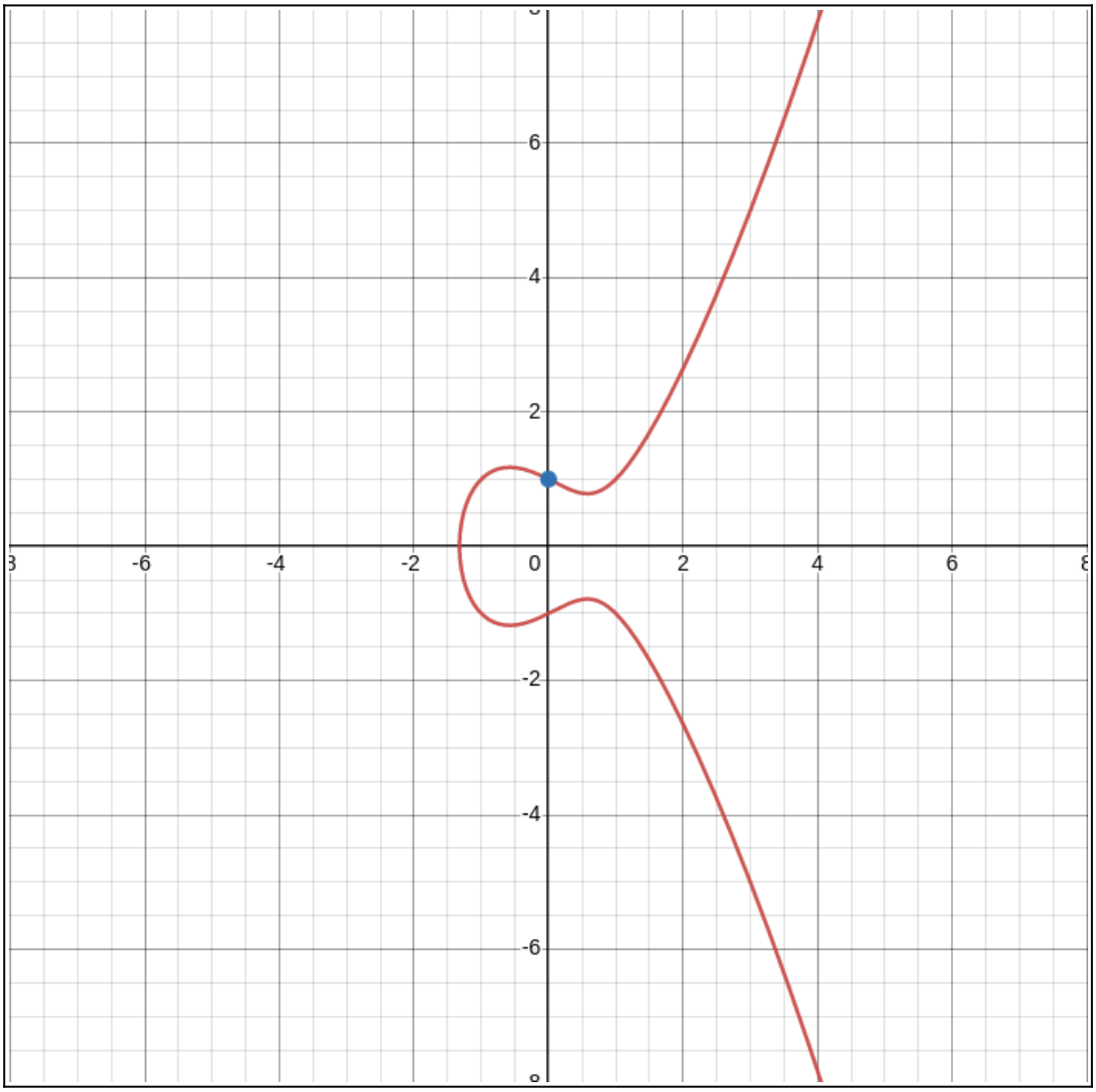
Цели работы:

- Получить базовые знания об эллиптических кривых
- Изучить основные принципы шифрования при помощи механизма эллиптических кривых
- Найти открытый текст шифротекста

вариант	**секретный ключ n_b**	**шифротекст**
6	44	{(377, 456), (367, 360)}; {(425, 663), (715, 398)}; {(188, 93), (279, 353)}; {(179, 275), (128, 79)}; {(568, 355), (515, 67)}; {(568, 355), (482, 230)}; {(377, 456), (206, 645)}; {(188, 93), (300, 455)}; {(489, 468), (362, 446)}; {(16, 416), (69, 510)}; {(425, 663), (218, 601)}

символа	kG	P_m+kP_B
1	(377, 456)	(367, 360)
2	(425, 663)	(715, 398)
3	(188, 93)	(279, 353)
4	(179, 275)	(128, 79)
5	(568, 355)	(515, 67)
6	(568, 355)	(482, 230)
7	(377, 456)	(206, 645)
8	(188, 93)	(300, 455)
9	(489, 468)	(362, 446)
10	(16, 416)	(69, 510)
11	(425, 663)	(218, 601)

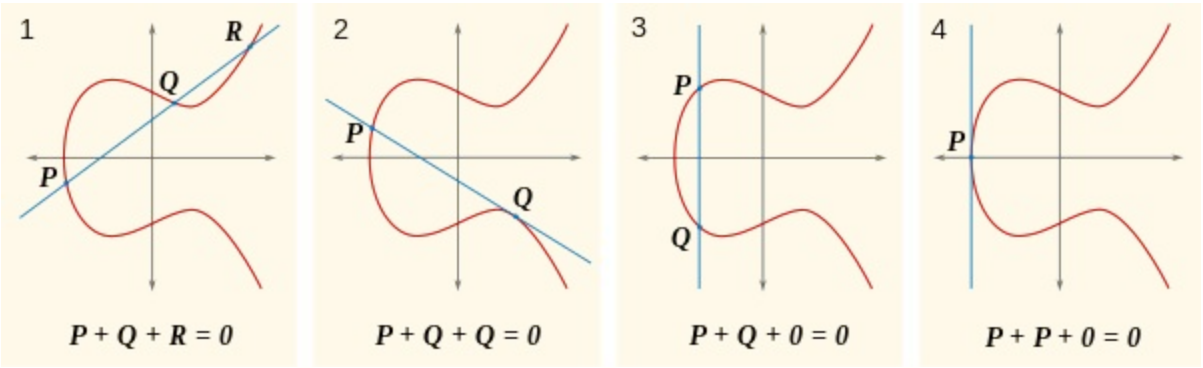
G = (0,1) – открытый ключ; точка на эллиптической кривой.



Введем класс точки эллиптической кривой. Точка может являться точкой в бесконечности.

```
data class EllipticCurvePoint (val x: Long, var y:Long, val pointAtInfinity: Boolean){
    fun isEq(p: EllipticCurvePoint): Boolean{
        return p.x == x && p.y == y
    }
}
```

Введем операцию сложения точек на эллиптической кривой:



Над полем целых чисел по модулю p:

```
fun pointAddition(
    p1: EllipticCurvePoint,
    p2: EllipticCurvePoint,
    primeNumber: Long,
    a: Long
): EllipticCurvePoint {
    if (p2.pointAtInfinity) {
        return p1
    }
    if (p1.pointAtInfinity) {
        return p2
    }
    val lambda: Long
    if (mod(p1.x - p2.x, primeNumber) == 0L) {
        if (mod(p1.y - p2.y, primeNumber) == 0L) {
            lambda = (3 * (p1.x) * (p1.x) + a) * invMod(2 * p1.y, primeNumber)
        } else {
            return EllipticCurvePoint(0, 0, true)
        }
    } else {
        lambda = mod((p2.y - p1.y), primeNumber) * invMod(p2.x - p1.x, primeNumber)
    }

    val x3 = mod(lambda * lambda - p1.x - p2.x, primeNumber)
    val y3 = mod((lambda * (p1.x - x3) - p1.y), primeNumber)
    return EllipticCurvePoint(x3, y3, false)
```

Операция умножения на скалярное n вводится как выполнение сложения точки самой с собой n раз. Оптимизированным способом выполнить умножения является удвоение-сложение:

```
private fun multiplyPoint(point: EllipticCurvePoint, m: Long): EllipticCurvePoint {
    var p = EllipticCurvePoint(0, 0, true)
    var bits = Integer.toBinaryString(m.toInt())
    var i = bits.length
    while (i > 0) {
        p = pointAddition(p, p, primeNumber, a)
        if (bits[bits.length - i] == '1') {
            p = pointAddition(point, p, primeNumber, a)
        }
        i--
    }
    return p
}
```

Вычитание вводится как сложение с инвертированной точкой:

```
private fun subPoints(p1: EllipticCurvePoint, p2: EllipticCurvePoint): EllipticCurvePoint {
    return pointSub(p1, p2, primeNumber, a)
}
```

Теперь можно выполнять дешифрование точки ({P1.x, P1.y},{P2.x, P2.y}):

Найдем P1 * secretKey

```
val np = multiplyPoint(p1, secretKey)
```

Произведем вычитание точек P2 - (P1*secretKey)

```
val s = subPoints(p2. np)
```

И найдем соответствие точки с символом алфавита:

```
println(String.format("(%3d,%3d) | %s", s.x, s.y, alphabet.getLetterFromCode(Pair(s.x.toInt(), s.y.toInt()))))
```

```
(235,732) | з
(228,271) | а
(240,309) | о
(243,664) | с
(247,266) | т
(243, 87) | р
(234,587) | е
(238,576) | н
(238,576) | н
(253,540) | ы
(236,712) | й
```

Вывод

В ходе выполнения лабораторной работы я ознакомился с основами шифрования при помощи эллиптических кривых. В ходе реализации алгоритма для выполнения операций над точками эллиптических кривых пришлось освежить в памяти модульную арифметику, алгоритм Евклида.

весь код:

https://github.com/danANDla/InfoSec/tree/master/labs/l2.6_elliptic_curves_decryption