МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
(МОСКОВСКИЙ ПОЛИТЕХ)

Факультет информационных технологий
Кафедра «Инфокогнитивные технологии»

ЛАБОРАТОРНАЯ РАБОТА №6

на тему: *«Изучение свойств группы эллиптической кривой»*

Направление подготовки 09.03.03 «Прикладная информатика»
Профиль «Корпоративные информационные системы»
Дисциплина «Защита информации»

Выполнил:

студентка группы 201-361

Саблина Анна Викторовна

Проверил:

Харченко Елена Алексеевна

Москва 2023

Теоретическая часть

Эллиптическая криптография (*ЭК*) – это раздел современной криптографии, основанный на математической теории эллиптических кривых. Она использует свойства эллиптических кривых для создания криптографических алгоритмов, обеспечивающих безопасность и эффективность в области шифрования, подписей и других криптографических применений.

В эллиптической криптографии основным строительным блоком является *эллиптическая кривая*, которая представляет собой множество точек, удовлетворяющих определенному уравнению в координатной плоскости. Криптографические операции, такие как сложение и удваивание точек на эллиптической кривой, образуют алгебраическую структуру, которая используется для выполнения шифрования и других операций.

Преимущества эллиптической криптографии включают высокую стойкость и эффективность по сравнению с классическими криптографическими алгоритмами, такими как RSA. Это достигается благодаря использованию более коротких ключей при достижении той же уровня безопасности. Это делает эллиптическую криптографию особенно полезной в ситуациях, где требуется высокая безопасность при ограниченных вычислительных ресурсах, таких как мобильные устройства или сети с низкой пропускной способностью.

Эллиптическая криптография широко применяется в современных системах безопасности, включая протоколы шифрования данных, электронные цифровые подписи, аутентификацию и другие криптографические приложения.

Эллиптическая кривая – это набор точек, описывающихся уравнением Вейерштрассе:

$$y^2 = x^3 + ax + b$$

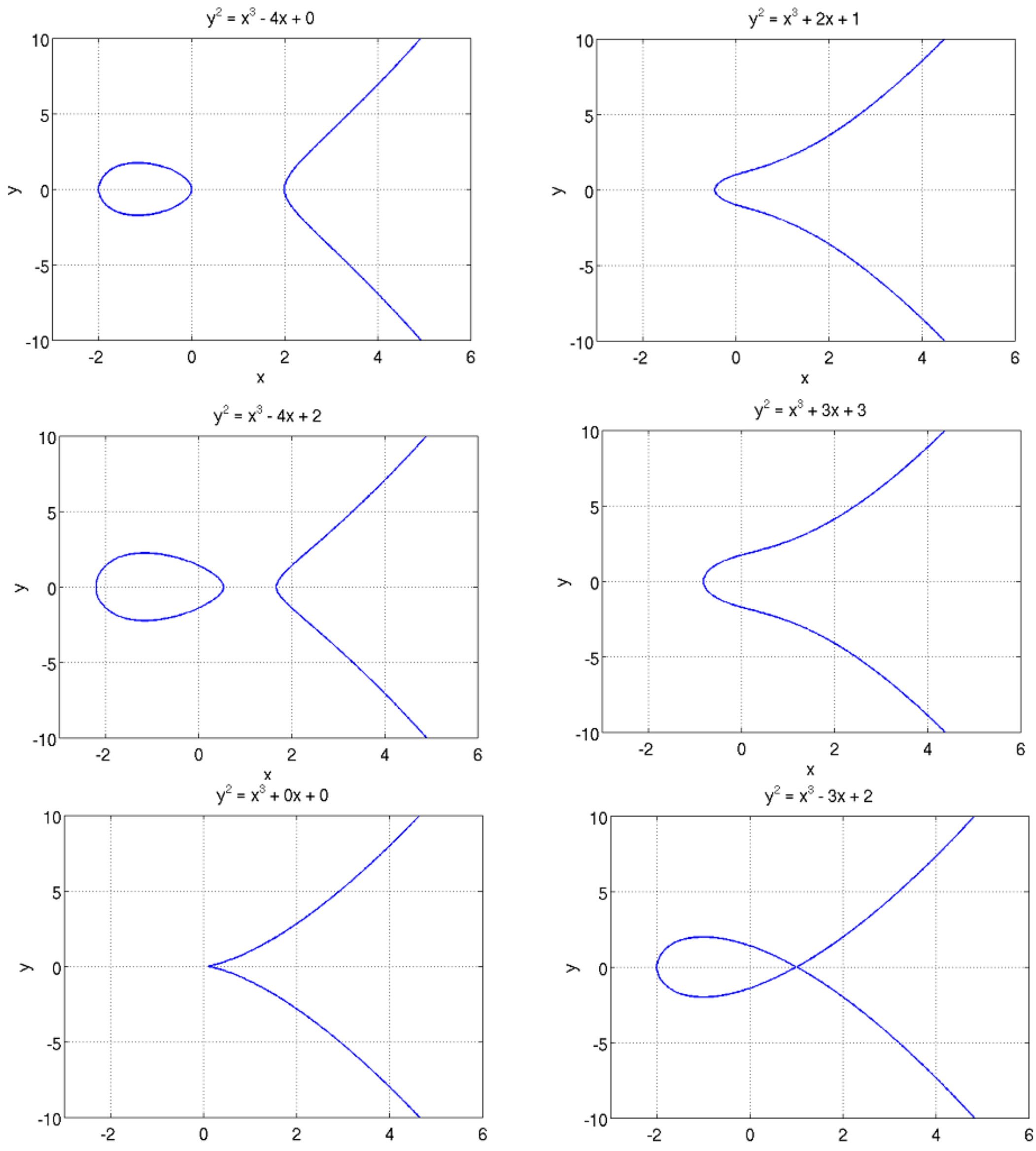

Рисунок 1 – Типичные варианты графиков эллиптических кривых

Эллиптические кривые, представленные на первых 4-х графиках, называются *гладкими*. В то время как две нижние кривые относятся к *сингулярным эллиптическим кривым*.

Для *гладких эллиптических кривых* выполняется следующее неравенство:

$$4a^3 + 27b^2 \neq 0$$

Тогда как для *сингулярных кривых* это условие не выполняется.

Важно отметить, что нельзя использовать в схемах электронной цифровой подписи (далее – ЭЦП) сингулярные кривые: использование сингулярных кривых повышает риск значительного снижения стойкости схемы ЭЦП.

Арифметические операции в эллиптической криптографии производятся над точками кривой. Основной операцией является «*сложение*».

Сложение двух точек легко представить графически:

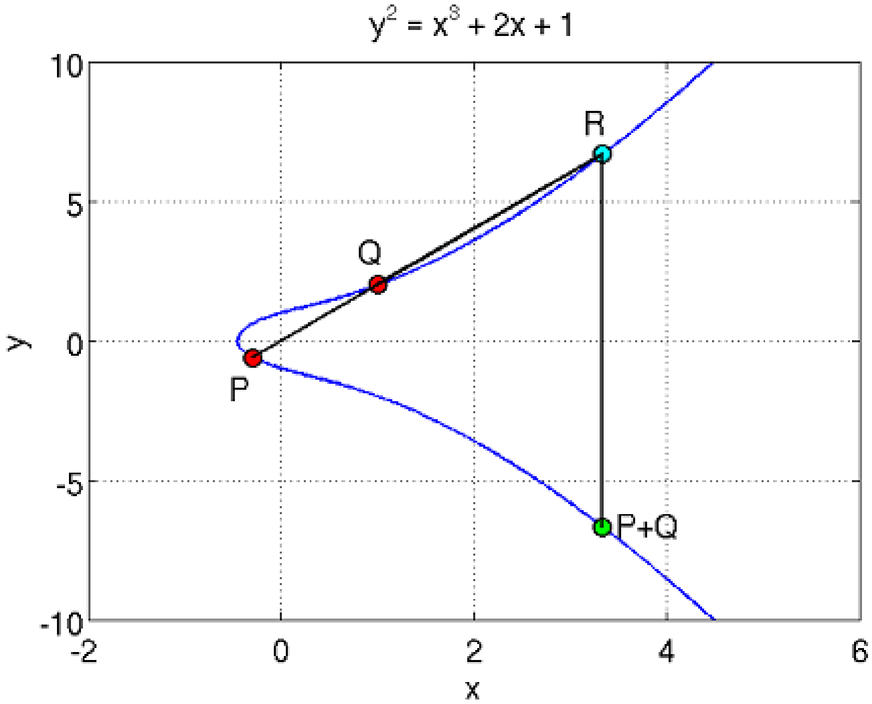

Рисунок 2 – Сложение двух точек

Как видно из рисунка, для сложения точек P и Q, необходимо провести между ними прямую линию, которая обязательно пересечет кривую в какой-либо третьей точке R. Отразим точку R относительно горизонтальной оси координат и получим искомую точку $P+Q$.

Алгебраическое представление «сложения»

Запишем сложение двух точек в виде формулы:

$$P + Q = -R$$

Пусть координатами точки P будут (x_P, y_P), а координатами точки Q соответственно (x_Q, y_Q). Вычислим

$$\alpha = \frac{y_Q - y_P}{x_Q - x_P}$$

и тогда координаты точки $P+Q$ будут равны:

$$x_{P+Q} = \alpha^2 - x_P - x_Q$$

$$y_{P+Q} = -y_P + \alpha\left(x_p - x_R\right)$$

Алгебраическое представление «удваивания»

Пусть точка P имеет координаты $(x_P,\, y_P)$ на кривой, и пусть α представляет собой наклонную линию, касательную к кривой в точке P. Тогда удваивание точки P обозначается как $2P$ и вычисляется следующим образом:

$$\alpha = \frac{3x_P^2 + a}{2y_P}$$

Здесь a представляет коэффициент, связанный с параметрами кривой. После вычисления наклонной линии α, координаты точки $2P$ находятся следующим образом:

$$x_{2P} = \alpha^2 - 2x_P$$

$$y_{2P} = -y_P + \alpha(x_P - x_{2P})$$

Таким образом, удваивание точки P позволяет найти новую точку на кривой, которая является результатом *сложения* точки P *с самой собой*.

Практическая часть

Для реализации программы, генерирующей и визуализирующей все решения уравнения вида $y^2 \equiv x^3 + ax + b \pmod p$, где $a, b \in \mathbb{Z}_p$, где p – простое число, была использована графическая библиотека `TKinter`.

Также были реализованы операции: 1) сложения двух точек кривой; 2) удвоения точки кривой.

Был создан проект на языке Python.

Общий принцип работы программы:

1. Создается кликабельный интерфейс программы.
2. Запускается интро.
3. При заполнении полей, содержащих значения параметров a и b уравнения эллиптической кривой, по нажатию на кнопку «Сгенерировать график» отображается кривая, соответствующая значениям введенных параметров.
4. При заполнении полей, содержащих значения по оси X точек, для которых необходимо совершить операцию сложения/удвоения, по нажатию на кнопку «Рассчитать» производится необходимая операция:
   a) если введены две различные точки, для них производится операция сложения;
   b) если введены две одинаковые точки, для них производится операция удвоения.

```
# Создание окна tkinter
window = tk.Tk()
window.title("dp_Lab6_py")

# Создание объекта Figure для matplotlib
figure = Figure(figsize=(6, 4), dpi=100)
subplot = figure.add_subplot(111)

# Создание холста для отображения графика matplotlib в tkinter
```

```
canvas = FigureCanvasTkAgg(figure, master=window)
canvas.draw()

canvas.get_tk_widget().pack(side=tk.LEFT)

# Создание фрейма для кнопки и полей ввода
controls_frame = tk.Frame(window)
controls_frame.pack(side=tk.RIGHT)

# Создание метки и поля ввода для параметра 'a'
label1 = tk.Label(controls_frame, text="Введите a:")
label1.pack(anchor=tk.W)  # Выравнивание метки слева

entryA = tk.Entry(controls_frame)
entryA.pack()

# Создание метки и поля ввода для параметра 'b'
label2 = tk.Label(controls_frame, text="Введите b:")
label2.pack(anchor=tk.W)  # Выравнивание метки слева

entryB = tk.Entry(controls_frame)
entryB.pack()

# Создание кнопки для генерации графика
button = tk.Button(controls_frame, text="Сгенерировать график", command=generate_graph)
button.pack()

# Создание первого поля ввода
entryX1 = tk.Entry(controls_frame)
entryX1.pack(pady=(50, 0))

# Создание второго поля ввода
entryX2 = tk.Entry(controls_frame)
entryX2.pack()

# Создание кнопки для добавления точек
button = tk.Button(controls_frame, text="Добавить точки", command=calc_points)
button.pack()

a = 15
up = False

# Запуск цикла обработки событий tkinter
update_interface()
tk.mainloop()
```

Листинг 1 – Создание кликабельного интерфейса программы

```
intro = True


# Определение функции для интро
def update_interface():
    if intro:
        global up
        global a
        if not up:
            if a > -40:
                a -= 2
            else:
                up = True
        else:
            if a < 40:
                a += 2
            else:
                up = False
        generate_graphV(a, 0)
```

```
    # Обновление каждые 1000 milliseconds (1 second)
    window.after(50, update_interface)
```

Листинг 2 – Интро программы: динамически изменяющийся график

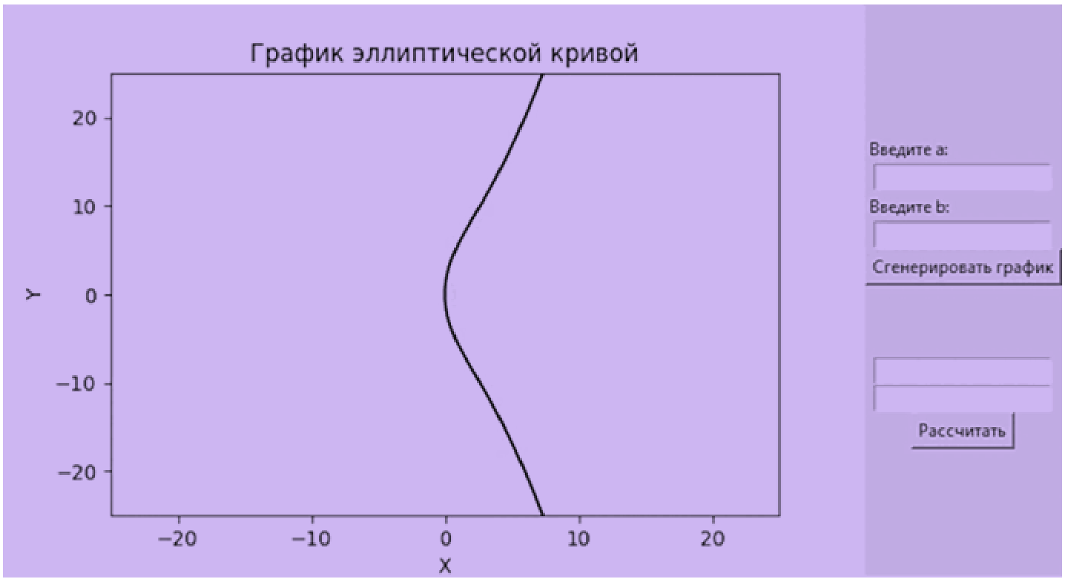

Рисунок 3 – Интро

Функции `f(x, a, b)`, `sum_two_points(x1, y1, x2, y2, a, b)`, `double_point(x, y, a, b)` и `calc_points()` содержат в себе основные математические вычисления, требующиеся по заданию:

- вычисление эллиптической кривой,
- операция удвоения точки,
- операция сложения двух точек,
- определение, какую из двух операций необходимо выполнить.

```
# Определение функции эллиптической кривой
def f(x, a, b):
    return x ** 3 + a * x + b


# Определение функции для сложения двух точек
def sum_two_points(x1, y1, x2, y2, a, b):
    kl = (y2 - y1) / (x2 - x1)
    bl = -x1 * kl + y1  # bl = -x2*kl + y2

    # y^2 = x^3 + ax + b, y = kl * x + bl => [-1, kl^2, 2 * kl * bl, bl^2 - b]
    poly = np.poly1d([-1, kl ** 2, 2 * kl * bl, bl ** 2 - b])

    # Корни уравнения
    x = np.roots(poly)
    y = np.sqrt(f(x, a, b))
    return x, y, kl, bl
```

```python
# Определение функции для удвоения точки
def double_point(x, y, a, b):
    kl = (3 * x ** 2 + a) / (2 * y)
    bl = -x * kl + y

    # y^2 = x^3 + ax + b, y = kl * x + bl => [-1, kl^2, 2 * kl * bl, bl^2 - b]
    poly = np.poly1d([-1, kl ** 2, 2 * kl * bl - a, bl ** 2 - b])

    # Корни уравнения
    x = np.roots(poly)
    y = np.sqrt(f(x, a, b))
    return x, y, kl, bl


# Определение функции для выполнения операций
def calc_points():
    a, b = get_params()

    # Точка 1
    x1 = int(entryX1.get())
    y1 = -np.sqrt(f(x1, a, b))

    # Точка 2
    x2 = int(entryX2.get())
    y2 = np.sqrt(f(x2, a, b))

    # Линия: y = kl * x + bl
    if x1 != x2:
        x, y, kl, bl = sum_two_points(x1, y1, x2, y2, a, b)
    else:
        x, y, kl, bl = double_point(x1, y1, a, b)

    scaling = max(int(max(x) * 2), int(max(y) * 2))
    generate_graph(scaling, scaling)
    subplot.plot(x, y, "o")
    subplot.plot(x, -y, "o")

    x = np.linspace(min(x), max(x))
    subplot.plot(x, kl * x + bl)
    canvas.draw()
```

Листинг 3 – Основные математические вычисления

Функции `get_params()`, `generate_graph(scaleX=25, scaleY=25)` и `generate_graphV(_a, _b)` генерируют график, при необходимости получая на вход параметры a и b.

```python
# Определение функции получения параметров
def get_params():
    a = int(entryA.get())
    b = int(entryB.get())
    return a, b


# Определение функции для генерации графа
def generate_graph(scaleX=25, scaleY=25):
    global intro
    intro = False
    subplot.cla()  # Очистка предыдущего графика

    a, b = get_params()
    Y, X = np.mgrid[-scaleX:scaleX:250j, -scaleY:scaleY:250j]
    print(scaleX, scaleY)
    subplot.contour(X, Y, Y ** 2 - f(X, a, b), levels=[0])
```

```
    subplot.set_xlabel("X")
    subplot.set_ylabel("Y")
    subplot.set_title("График эллиптической кривой")
    canvas.draw()
    return a, b


# Определение функции для обновления графа
def generate_graphV(_a, _b):
    subplot.cla()  # Очистка предыдущего графика
    a = _a
    b = _b

    Y, X = np.mgrid[-25:25:100j, -25:25:100j]
    subplot.contour(X, Y, Y ** 2 - f(X, a, b), levels=[0])

    subplot.set_xlabel("X")
    subplot.set_ylabel("Y")
    subplot.set_title("График эллиптической кривой")
    canvas.draw()
```

Листинг 4 – Генерация графика

Наглядный пример использования программы:

Рисунок 4 – График при введенных параметрах уравнения

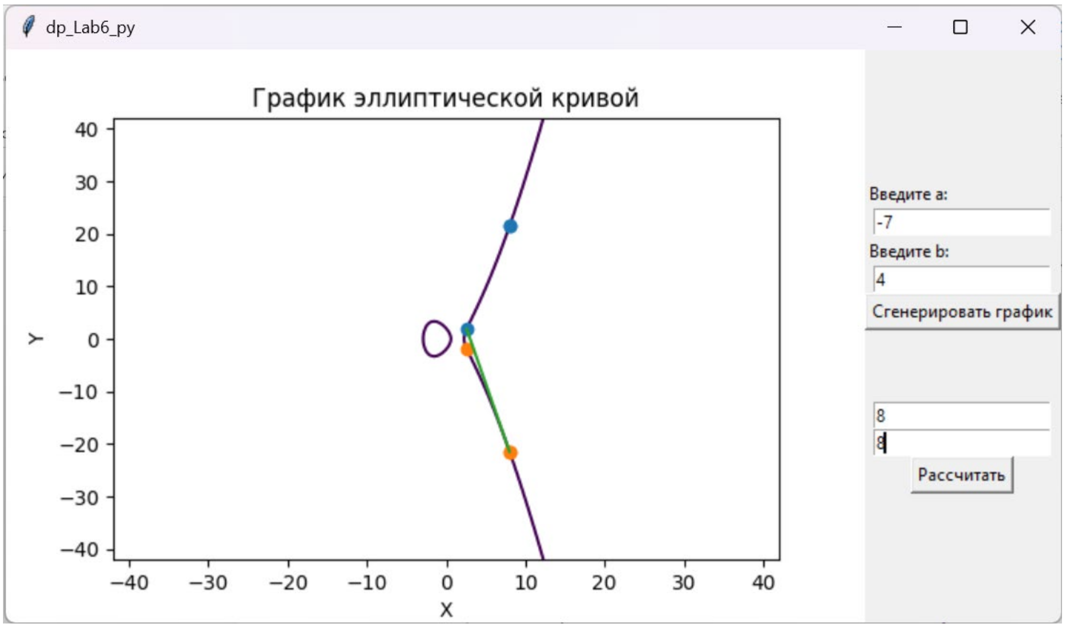

Рисунок 5 – График после выполнения операции по удвоению точки

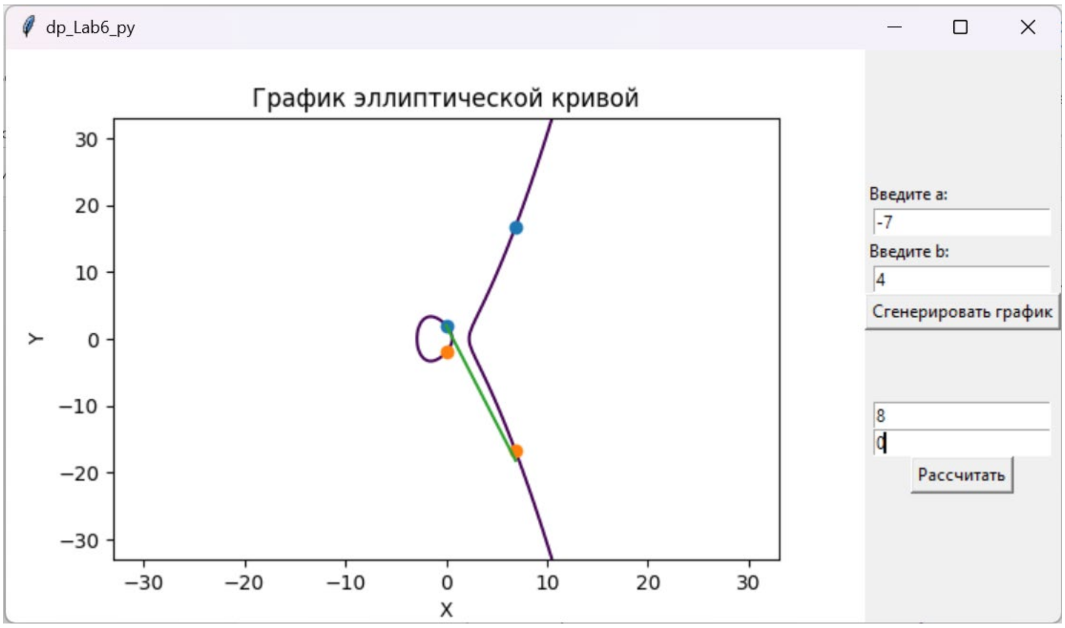

Рисунок 6 – График после выполнения операции по сложению двух точек

Ссылка на проект в репозитории GitHub:

– https://github.com/LazyShAman/dp/tree/main/6.