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

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:

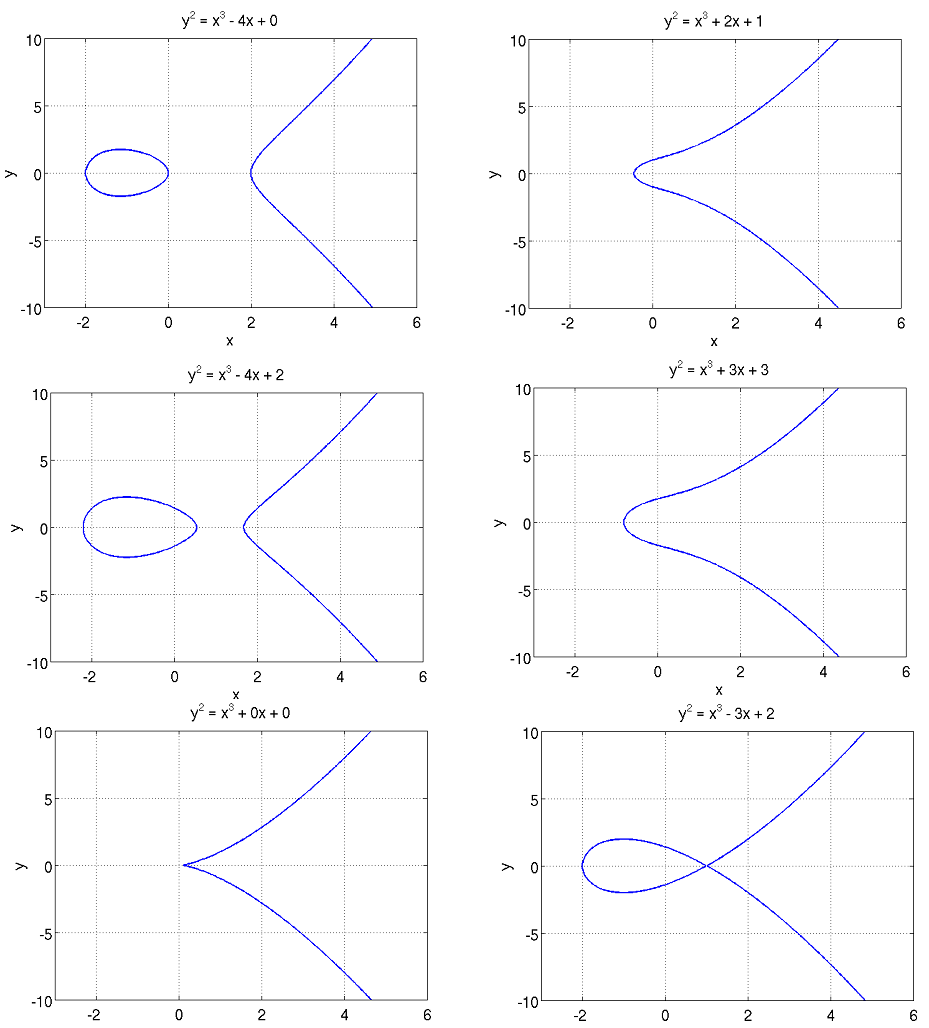


Рисунок 1 – Типичные варианты графиков эллиптических кривых

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:

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:

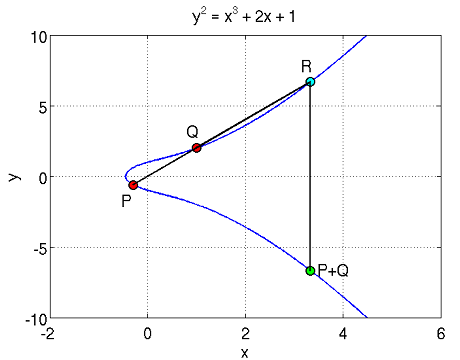


Рисунок 2 – Сложение двух точек

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

**Алгебраическое представление «сложения»**

Запишем сложение двух точек в виде формулы:

Пусть координатами точки будут , а координатами точки соответственно . Вычислим

и тогда координаты точки будут равны:

**Алгебраическое представление «удваивания»**

Пусть точка имеет координаты на кривой, и пусть представляет собой наклонную линию, касательную к кривой в точке . Тогда удваивание точки обозначается как и вычисляется следующим образом:

Здесь представляет коэффициент, связанный с параметрами кривой. После вычисления наклонной линии , координаты точки находятся следующим образом:

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

Практическая часть

Для реализации программы, генерирующей и визуализирующей все решения уравнения вида , где , 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 и уравнения эллиптической кривой, по нажатию на кнопку «Сгенерировать график» отображается кривая, соответствующая значениям введенных параметров.
4. При заполнении полей, содержащих значения по оси точек, для которых необходимо совершить операцию сложения/удвоения, по нажатию на кнопку «Рассчитать» производится необходимая операция:
   1. если введены две различные точки, для них производится операция сложения;
   2. 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

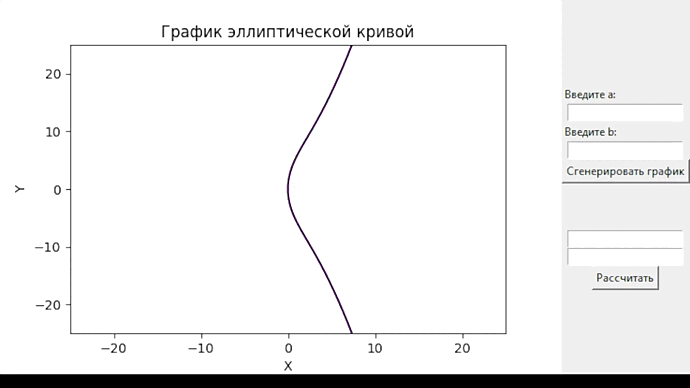


Рисунок 3 – Интро

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  
  
  
# 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 и .

# 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

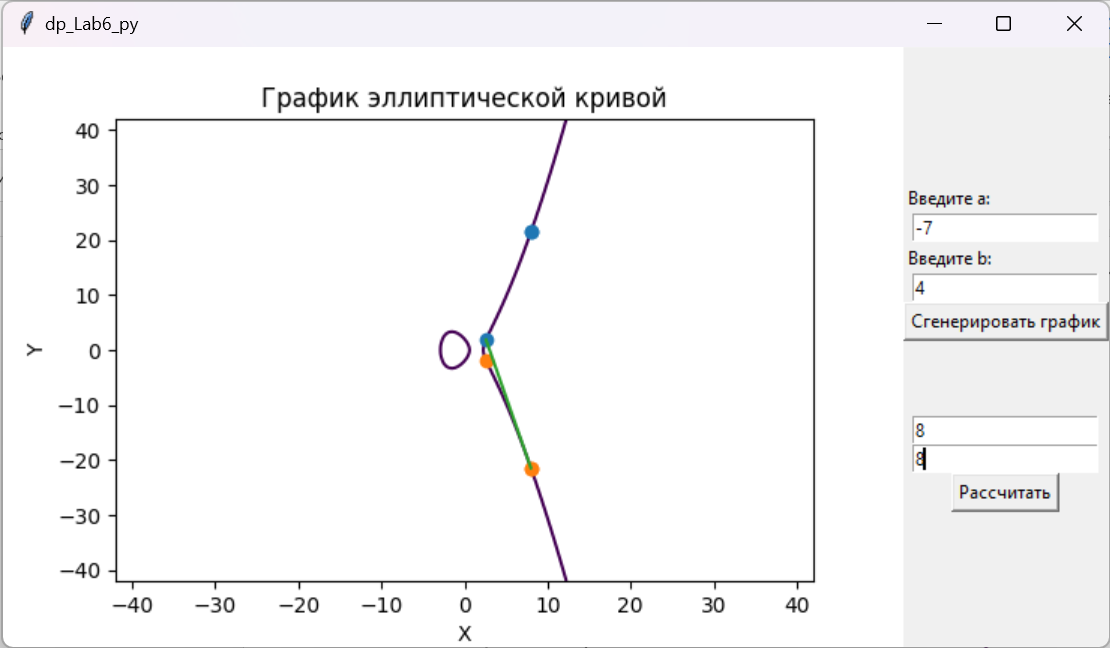


Рисунок 5 – График после выполнения операции по удвоению точки

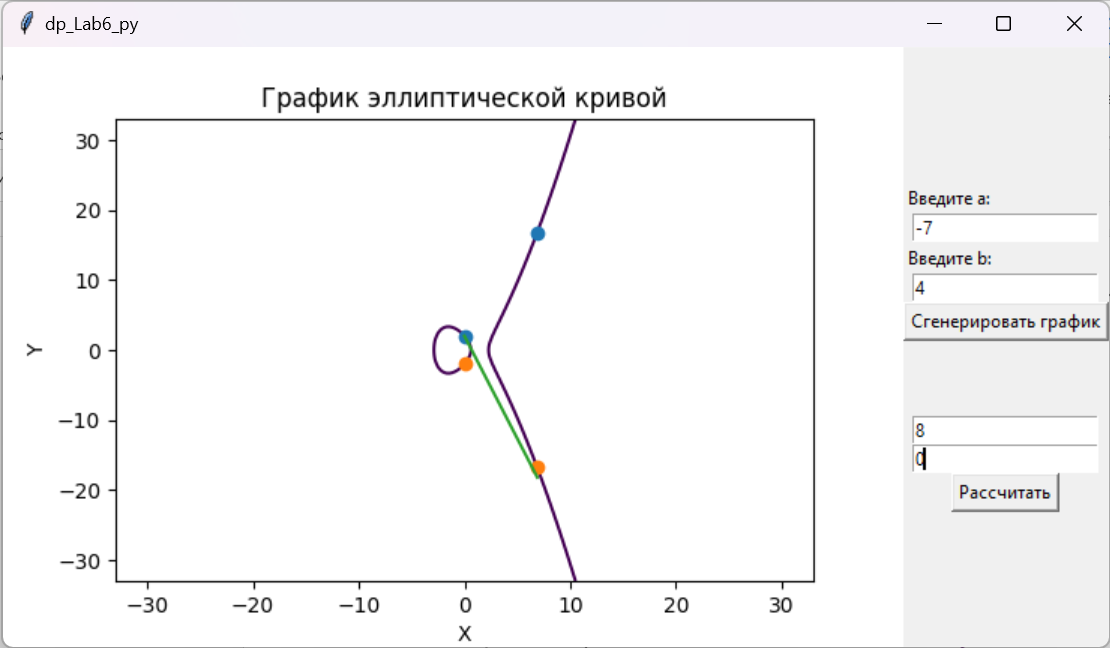


Рисунок 6 – График после выполнения операции по сложению двух точек

Ссылка на проект в репозитории GitHub:

* <https://github.com/LazyShAman/dp/tree/main/6>.