ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Московский Институт Электроники и Математики им. А. М. Тихонова
Факультет прикладной математики

Выпускная квалификационная работа

на тему: **Построение эллиптических кривых с комплексным умножением для криптографических приложений**

по направлению подготовки 10.05.01 «Компьютерная безопасность»

СОГЛАСОВАНО

Научный руководитель,
Доцент кафедры Компьютерной
безопасности НИУ ВШЭ МИЭМ

_____ А.Ю. Нестеренко.
«_____» _____ 2024 г.

ВЫПОЛНИЛ

студент группы СКБ182
образовательной программы
10.05.01 «Компьютерная безопасность»

_____ В. А. Парамонов
«_____» _____ 2024 г.

Рецензент
Руководитель направления разработки,
СКЗИ НТК "Криптонит"
Подставин С.Е.

Москва 2024

Аннотация

Исследование процесса вычисления кратной точки занимает центральное место в сфере криптографии с открытым ключом. Применение комплексного умножения в эллиптических кривых, которое ранее было предметом, в основном теоретического интереса, набирает популярность в последние годы. Настоящая работа посвящена исследованию операционной эффективности алгоритмов вычисления кратных точек с использованием комплексного умножения, применяя для этого инструменты программирования Python и Sage.

Данная работа состоит из 25 страниц, 4 глав, 4 рисунков, 4 таблиц, 6 приложений. Использовано 11 источников.

Ключевые слова: эллиптическая кривая; вычисление кратной точки; эндоморфизм; комплексное умножение.

Abstract

The computation of multiple points remains a cornerstone task in the realm of public-key cryptography. The incorporation of complex multiplication in elliptic curves, once primarily of theoretical interest, has gained significant traction in recent years. This paper delves into an investigation of the operational efficiency in executing multiple point computations through complex multiplication, employing Python and Sage as the programming tools.

The paper contains 25 pages, 4 chapters, 4 figures, 4 tables, 6 appendices. 11 sources are used.

Keywords: elliptic curve; point multiplication; endomorphism; complex multiplication;.

Содержание

Используемые определения и термины

Аддитивная Абелева группа – Аддитивная Абелева группа – это такая группа, где групповая операция (в данном случае сложение) является коммутативной. На множестве точек эллиптической кривой можно определить структуру абелевой группы. [1].

Бесконечно удаленная точка – Бесконечно удаленная точка эллиптической кривой (обозначается O) определяется как точка, векторы которой эквивалентны вектору $(0 : 1 : 0)$. Для каждого такого вектора уравнение (1) выполняется, и последняя координата равна 0. [1].

Изоморфизм эллиптических кривых – Изоморфизмом между двумя эллиптическими кривыми E_1 и E_2 называется биективное гомоморфное отображение $\phi : E_1 \to E_2$, которое сохраняет структуру группы. Другими словами, ϕ сопоставляет каждой точке на E_1 уникальную точку на E_2 таким образом, что сложение точек сохраняется. [1].

Комплексное умножение – Комплексное умножение – это класс эллиптических кривых, для которых существует не только эндоморфизм умножения на число, но и эндоморфизмы, соответствующие умножению на некоторые комплексные числа. Это расширение классической концепции умножения точки кривой на целое число. [2].

Конечное поле – Конечным полем называется поле, содержащее конечное количество элементов. Эллиптические кривые над конечными полями обладают рядом уникальных свойств, делающих их полезными в криптографии. [1].

Конечные поля и порядок группы – Если поле F конечно, то группа точек эллиптической кривой также конечна. Порядок группы точек эллиптической кривой – это количество элементов такой группы, обозначается $|E(F)|$. [1].

Обратный элемент – Для любой точки эллиптической кривой $(x_1 : y_1 : z_1)$ существует обратная точка $(x_1 : -y_1 : z_1)$. [1].

Проективное пространство – Пусть задано произвольное поле F. Множество векторов $P^2(F) = \{(x : y : z), x, y, z \in F\}$ определяем как двумерное проективное пространство над полем F. В данном пространстве вводится отношение эквивалентности: $(x : y : z) \sim (x_1 : y_1 : z_1)$, если существует $\lambda \in F$, $\lambda \neq 0$, такое что $x_1 = \lambda x$, $y_1 = \lambda y$, $z_1 = \lambda z$. [1].

Ранг эллиптической кривой – Ранг эллиптической кривой над конечным полем определяется как максимальное число линейно независимых точек бесконечного порядка на этой кривой. Ранг является важным параметром при анализе структуры группы точек на кривой. [2].

Скалярное умножение – Скалярное умножение точки P эллиптической кривой на натуральное число n определяется как $[n]P = P + P + \cdots + P$ (n раз). [1].

Торсионные точки – Торсионными точками эллиптической кривой называются такие точки, которые при умножении на некоторое натуральное число n дают нулевую точку (точку на бесконечности) кривой. Множество всех торсионных точек кривой образует торсионную подгруппу. [1].

Точка проективного пространства – Множество векторов, эквивалентных в рамках определенного выше отношения, представляет собой точку введенного проективного пространства. [1].

Уравнения над кольцом вычетов – Уравнения эллиптических кривых можно рассматривать над произвольным кольцом вычетов Z_n. В этом случае уравнение принимает вид: $y^2 z \equiv x^3 + axz^2 + bz^3 \pmod{n}$, $a, b, x, y, z \in Z_n$. [1].

Характеристика поля – Характеристикой поля F является наименьшее неотрицательное целое число p, для которого выполняется равенство $p \cdot a = 0$ для любого элемента $a \in F$. [1].

Эллиптическая кривая – При характеристике поля F больше 3, рассмотрим однородное уравнение эллиптической кривой вида $y^2z = x^3 + axz^2 + bz^3$ (уравнение 1), где $4a^3 + 27b^2 \neq 0$, $a, b \in F$. Множество точек проективного пространства $P^2(F)$, удовлетворяющих данному уравнению, называется эллиптической кривой над полем F. [1].

Эллиптической кривая в аффинной форме – Множество решений уравнения (1) в аффинной форме вместе с бесконечно удаленной точкой O образует эллиптическую кривую над полем F. Аффинная точка эллиптической кривой представляет собой пару (x, y), удовлетворяющую уравнению $y^2 = x^3 + ax + b$ (уравнение 2). [1].

Эндоморфизм – Эндоморфизмом эллиптической кривой называется такое гомоморфное отображение кривой в себя, которое сохраняет структуру группы. То есть, для эндоморфизма ϕ и любых точек P, Q на эллиптической кривой E, выполняется условие $\phi(P + Q) = \phi(P) + \phi(Q)$. [2].

Введение

Целью данной выпускной квалификационной работы является исследование эффективности использования комплексного умножения для вычисления кратных точек на эллиптических кривых. Исследование включает в себя построение эллиптической кривой с комплексным умножением, сравнение форм эллиптических кривых, анализ количества действий, а также разработку программной реализации.

Актуальность данного исследования обусловлена широким применением эллиптических кривых в современной криптографии, включая криптографические системы с открытым ключом. Эффективность форм эллиптических кривых для вычисления кратных точек напрямую влияет на скорость выполнения криптографических операций и, как следствие, на общую производительность криптографических систем.

Новизна данной работы заключается в отсутствии полноценного сравнения методов комплексного умножения с точки зрения количества необходимых операций для их выполнения, несмотря на растущий интерес к этой тематике в академическом сообществе. Дипломная работа состоит из введения, четырех основных глав, заключения, библиографического списка и приложения. В первой главе излагаются основные формы, используемые в работе. Вторая глава посвящена описанию эндоморфизма, используемого в работе. Третья глава фокусируется на определении основных алгоритмов вычисления кратной точки, используемых в вычислениях. Четвертая глава состоит из построения и сравнения результатов при вычислении кратной точки. В приложении представлен программный код, реализованный на языках программирования Python + Sage.

Глава 1. Сравниваемые формы эллиптических кривых

Для анализа и сравнения было выбрано 4 формы эллиптических кривых.

$W_{a,b}$	$y_w^2 \equiv x_w^3 + ax_w + b(\mathrm{mod}p)$	короткая форма Вейерштрасса
$M_{\mu,\gamma}$	$\gamma y_m^2 \equiv x_m^3 + \lambda x_m^2 + x_m(\mathrm{mod}p)$	форма Монтгомери
$E_{e,d}$	$ex_e^2 + y_e^2 \equiv 1 + dx_e^2 y_e^2(\mathrm{mod}p)$	искривленная форма Эдвардса
$E_{a,b}$	$y_m^2 \equiv x_m^3 + ax_m^2 + bx_m(\mathrm{mod}p)$	форма Хадано

Таблица 1 — Исследуемые формы эллиптических кривых

1.1. Короткая форма Вейерштрасса

Кривая, в короткой форме Вейерштрасса [3]

$$W_{a,b}: \quad y_w^2 \equiv x_w^3 + ax_w + b \pmod{p}$$

(a) Переход из короткой формы Вейерштрасса в форму Монтгомери задается следующими соотношениями. Пусть θ - корень многочлена $x^3 + ax + b$ в поле $\mathbb{F}_p$ и элемент γ удовлетворяет равенству $\gamma^2 \equiv 3\theta^2 + a(\mathrm{mod}p)$ и $\gamma \not\equiv 0(\mathrm{mod}p)$, тогда определим

$$\mu \equiv \frac{3\theta}{\gamma} \pmod{p}$$

Одновременно, любая аффинная точка преобразуется следующим образом:

$$(x_w, y_w) \to \left(x_m \equiv \frac{x_w - \theta}{\gamma} \pmod{p}, \quad y_m \equiv \frac{y_w}{\gamma^2} \pmod{p} \right)$$

Групповой закон в аффинных координатах
Сложение точек:

$$x_3 = \frac{(y_2 - y_1)^2}{(x_2 - x_1)^2} - x_1 - x_2$$
$$y_3 = \frac{(2x_1 + x_2)(y_2 - y_1)}{x_2 - x_1} - \frac{(y_2 - y_1)^3}{(x_2 - x_1)^3} - y_1$$

Удвоение точки:

$$x_3 = \frac{(3x_1^2 + a)^2}{(2y_1)^2} - x_1 - x_1$$
$$y_3 = \frac{(2x_1 + x_1)(3x_1^2 + a)}{2y_1} - \frac{(3x_1^2 + a)^3}{(2y_1)^3} - y_1$$

Эллиптическая кривая в короткой форме Вейерштрасса имеет следующий вид в проективных координатах:

$$y^2 z = x^3 + axz^2 + bz^3$$

Формулы сложения точек в проективных координатах [4]:

$$\begin{aligned}
y_1 z_1 &= y_1 z_2 \\
x_1 z_2 &= x_1 z_2 \\
z_1 z_2 &= z_1 z_2 \\
u &= y_2 z_1 - y_1 z_2 \\
uu &= u \times u \\
v &= x_2 z_1 - x_1 z_2 \\
vv &= v \times v \\
vvv &= v \times vv \\
R &= vv \times x_1 z_2 \\
A &= uu \times z_1 z_2 - vv - 2R \\
x_3 &= v \times A \\
y_3 &= u(R - A) - vvv \times y_1 z_2 \\
z_3 &= vvv \times z_1 z_2
\end{aligned}$$

В результате получается 7 сложений, 14 умножений.

Формулы удвоения точек [4]:

$$\begin{aligned}
w &= a \times z_1^2 + 3 \times x_1^2 \\
s &= y_1 \times z_1 \\
ss &= s \times s \\
sss &= s \times ss \\
r &= y_1 \times s \\
b &= x_1 \times r \\
h &= w^2 - 8 \times b \\
x_3 &= 2 \times h \times s \\
y_3 &= w(4b - h) - 8 \times r^2 \\
z_3 &= 8 \times sss
\end{aligned}$$

В результате получается 6 сложений, 16 умножений.

1.2. Форма Монтгомери

Кривая в форме Монтгомери определяется следующим сравнением [5]

$$M_{\mu,\gamma}: \quad \gamma y_m^2 \equiv x_m^3 + \mu x_m^2 + x_m \pmod{p}$$

Переход из формы Монтгомери в короткую форму Вейерштрасса задается одним из следующих способов.

Следуя (7), определим $\theta \equiv \frac{\gamma\mu}{3} (\mathrm{mod} p)$. Тогда,

$$\begin{aligned}
a &\equiv \gamma^2 - 3\theta^2 \equiv \gamma^2 - \frac{\gamma^2\mu^2}{3} \equiv \gamma^2 \left(1 - \frac{\mu^2}{3}\right) \pmod{p} \\
b &\equiv -\left(\theta^3 + a\theta\right) \equiv -\theta\left(\gamma^2 - 2\theta^2\right) \equiv \frac{2\gamma^3\mu^3}{27} - \frac{\gamma^3\mu}{3} \equiv \frac{\gamma^3\mu}{3}\left(\frac{2\mu^2}{9} - 1\right) \pmod{p}
\end{aligned}$$

При этом, любая аффинная точка преобразуется следующим образом.

$$(x_m, y_m) \to \left(x_w \equiv \gamma x_m + \frac{\gamma\mu}{3} \pmod{p}, \quad y_w \equiv \gamma^2 y_m \pmod{p}\right)$$

Действительно, подставляя полученные равенства в уравнение кривой, получим

$$\begin{aligned} x_w^3 + ax_w + b &\equiv \left(\gamma x_m + \frac{\gamma\mu}{3}\right)^3 + \gamma^2\left(1 - \frac{\mu^2}{3}\right)\left(\gamma x_m + \frac{\gamma\mu}{3}\right) + \frac{\gamma^3\mu}{3}\left(\frac{2\mu^2}{9} - 1\right) \equiv \\ &\equiv \gamma^3\left(x_m^3 + x_m^2\mu + \frac{x_m\mu^2}{3} + \frac{\mu^3}{27} + x_m - \frac{x_m\mu^2}{3} + \frac{\mu}{3} - \frac{\mu^3}{9} + \frac{2\mu^3}{27} - \frac{\mu}{3}\right) \equiv \\ &\equiv \gamma^3\left(x_m^3 + x_m^2\mu + x_m\right) \equiv \gamma^4 y_m^2 \equiv y_w^2 \pmod{p}. \end{aligned}$$

Переход из формы Монтгомери в искривленную форму Эдвардса задается следующим образом. Если $\mu \not\equiv \pm 2 (\mathrm{mod} p)$ и $\gamma \not\equiv 0 (\mathrm{mod} p)$, то коэффициенты кривой определяются сравнениями

$$e \equiv \frac{\mu + 2}{\gamma} \pmod{p}, \quad d \equiv \frac{\mu - 2}{\gamma} \pmod{p}$$

а преобразование отличных от нуля точек определяется следующим образом

$$(x_m, y_m) \to \left(x_e \equiv \frac{x_m}{y_m} \pmod{p}, \quad y_e \equiv \frac{x_m - 1}{x_m + 1} \pmod{p}\right)$$

Действительно, подставляя полученные равенства в уравнение кривой в искривленной форме Эдвардса, получим

$$\begin{aligned} ex_e^2 + y_e^2 &\equiv \frac{(\mu+2)}{\gamma}\frac{x_m^2}{y_m^2} + \frac{(x_m-1)^2}{(x_m+1)^2} \equiv \frac{(\mu+2)x_m^2}{x_m\left(x_m^2 + \mu x_m + 1\right)} + \frac{x_m^2 - 2x_m + 1}{x_m^2 + 2x_m + 1} \equiv \\ &\equiv \frac{(\mu+2)x_m^2\left(x_m^2 + 2x_m + 1\right) + \left(x_m^2 - 2x_m + 1\right)\left(x_m^3 + \mu x_m^2 + x_m\right)}{\left(x_m^3 + \mu x_m^2 + x_m\right)\left(x_m^2 + 2x_m + 1\right)} \equiv ? \equiv \\ &\equiv \frac{\left(x_m^3 + \mu x_m^2 + x_m\right)\left(x_m^2 + 2x_m + 1\right) + (\mu-2)x_m^2\left(x_m - 1\right)^2}{\left(x_m^3 + \mu x_m^2 + x_m\right)\left(x_m^2 + 2x_m + 1\right)} \equiv 1 + dx_e^2 y_e^2 \pmod{p}. \end{aligned}$$

Групповой закон в аффинных координатах выглядит следующим образом:
Сложение точек

$$\begin{aligned} x_3 &= b \cdot \frac{(y_2 - y_1)^2}{(x_2 - x_1)^2} - a - x_1 - x_2 \\ y_3 &= \frac{(2 \cdot x_1 + x_2 + a) \cdot (y_2 - y_1)}{x_2 - x_1} - b \cdot \frac{(y_2 - y_1)^3}{(x_2 - x_1)^3} - y_1 \end{aligned}$$

Удвоение точки

$$\begin{aligned} x_3 &= b \cdot \frac{(3 \cdot x_1^2 + 2 \cdot a \cdot x_1 + 1)^2}{(2 \cdot b \cdot y_1)^2} - a - x_1 - x_1 \\ y_3 &= \frac{(2 \cdot x_1 + x_1 + a) \cdot (3 \cdot x_1^2 + 2 \cdot a \cdot x_1 + 1)}{2 \cdot b \cdot y_1} - b \cdot \frac{(3 \cdot x_1^2 + 2 \cdot a \cdot x_1 + 1)^3}{(2 \cdot b \cdot y_1)^3} - y_1 \end{aligned}$$

Эллиптическая кривая в форме Монтгомери имеет следующий вид в проективных координатах:

$$a \cdot y^2 z = x^3 + b \cdot x^2 z + x z^2$$

Формулы сложения точек: [6]

$$
\begin{aligned}
a &= x_1 - z_1 \\
b &= x_1 + z_1 \\
c &= x_2 - z_1 \\
d &= x_2 + z_2 \\
e &= a \cdot d \\
f &= b \cdot c \\
ef &= e + f \\
emf &= e - f \\
ef^2 &= ef \cdot ef \\
emf^2 &= emf \cdot emf \\
x_3 &= \text{minusz} \cdot ef^2 \\
z_3 &= \text{minusx} \cdot emf^2
\end{aligned}
$$

где minusz и minusx это подаваемые параметры, равны базовой точке во время лестницы Монтгомери. В результате получается 6 сложений, 6 умножений

Формулы удвоения точек: [6]

$$
\begin{aligned}
a &= x_1 + z_1 \\
b &= x_1 - z_1 \\
c &= a \cdot a \\
d &= b \cdot b \\
e &= c - d \\
x_3 &= c \cdot d \\
C &= \frac{(a+2)}{4} \quad \text{(считается заранее)} \\
z_3 &= e \cdot (d + C \cdot e)
\end{aligned}
$$

В результате получается 4 сложения, 5 умножений

1.3. Искривленная форма Эдвардса

Кривая в искривленной форме Эдвардса определяется следующим сравнением [7]

$$E_{e,d}: \quad ex_e^2 + y_e^2 \equiv 1 + dx_e^2 y_e^2 \pmod{p}$$

Переход из искривленной формы Эдвардса в форму Монтгомери задается следующим образом. Рассматривая (10) как систему уравнений относительно неизвестных μ и γ, легко получить

$$
\begin{aligned}
\gamma(e+d) = 2\mu => \mu &= \frac{\gamma(e+d)}{2} \\
e = \frac{\mu+2}{\gamma} = \frac{\frac{\gamma(e+d)}{2}+2}{\gamma} &= \frac{e+d}{4} + \frac{2}{\gamma}
\end{aligned}
$$

тогда

$$\frac{3e-d}{4} = e - \frac{e+d}{4} = \frac{2}{\gamma} => \gamma = \frac{8}{3e-d}$$
$$\mu = \frac{\gamma(e+d)}{2} = \frac{4(e+d)}{3e-d}$$

$$\mu \equiv \frac{2(e+d)}{e-d} \pmod{p}, \quad \gamma \equiv \frac{4}{e-d} \pmod{p}$$
$$(x_e, y_e) \to \left(x_m \equiv \frac{1+y_e}{1-y_e} \pmod{p}, \quad y_m \equiv \frac{x_m}{x_e} \pmod{p} \right)$$

Групповой закон в аффинных координатах

Сложение точек

$$x_3 = \frac{x_1 \cdot y_2 + y_1 \cdot x_2}{1 + d \cdot x_1 \cdot x_2 \cdot y_1 \cdot y_2}$$
$$y_3 = \frac{y_1 \cdot y_2 - a \cdot x_1 \cdot x_2}{1 - d \cdot x_1 \cdot x_2 \cdot y_1 \cdot y_2}$$

Удвоение точки

$$x_3 = \frac{x_1 \cdot y_1 + y_1 \cdot x_1}{1 + d \cdot x_1 \cdot x_1 \cdot y_1 \cdot y_1}$$
$$y_3 = \frac{y_1 \cdot y_1 - a \cdot x_1 \cdot x_1}{1 - d \cdot x_1 \cdot x_1 \cdot y_1 \cdot y_1}$$

Эллиптическая кривая в форме Скрученного Эдвардса имеет следующий вид в проективных координатах:

$$a \cdot x^2 z^2 + y^2 z^2 = z^4 + d \cdot x^2 \cdot y^2$$

Формулы сложения точек: [8]

$$A = z_1 \cdot z_2$$
$$B = A \cdot A$$
$$C = x_1 \cdot x_2$$
$$D = y_1 \cdot y_2$$
$$E = d \cdot C \cdot D$$
$$F = B - E$$
$$G = B + E$$
$$x_3 = A \cdot F \cdot ((x_1 + y_1) \cdot (x_2 + y_2) - C - D)$$
$$y_3 = A \cdot G \cdot (D - a \cdot C)$$
$$z_3 = F \cdot G$$

В результате получается 6 сложений, 13 умножений

Формулы удвоения точек: [8]

$$
\begin{aligned}
B &= (x_1 + y_1) \\
B &= B \cdot B \\
C &= x_1 \cdot x_1 \\
D &= y_1 \cdot y_1 \\
E &= a \cdot C \\
F &= E + D \\
H &= z_1 \cdot z_1 \\
J &= F - 2 \cdot H \\
x_3 &= (B - C - D) \cdot J \\
y_3 &= F \cdot (E - D) \\
z_3 &= F \cdot J
\end{aligned}
$$

В результате получается 6 сложений, 9 умножений

1.4. Форма Хадано

Пусть $\gamma \in \mathbb{C}$ произвольный, отличный от нуля элемент, а θ - корень многочлена, стоящего в правой части равенства (2). Сделаем замену переменных

$$u = \gamma(x - \theta), \quad v = \gamma^{\frac{3}{2}} y$$

Тогда, записывая равенства

$$x = \frac{u}{\gamma} + \theta, \quad y = \frac{1}{\gamma^{\frac{3}{2}}} v$$

и подставляя их в (2), получим, что новые переменные удовлетворяют равенству

$$\frac{1}{\gamma^3} v^2 = \frac{u^3}{\gamma^3} + \frac{3u^2\theta}{\gamma^2} + \frac{3u\theta^2}{\gamma} + \theta^3 + \frac{au}{\gamma} + a\theta + b$$

Сокращая на отличный от нуля элемент γ^{-3} получим равенство

$$v^3 = u^3 + 3\theta\gamma u^2 + \left(3\theta^2 + a\right)\gamma^2 u$$

которое мы будем называть эллиптической кривой, заданной в форме Хадано [9].
Формулы удвоения точки

$$
\begin{aligned}
\lambda &= \frac{3x_1^2 + 2a_2x_1 + a_4}{2y_1} \\
x_3 &= \lambda^2 - a_2 - 2x_1 \\
y_3 &= (\lambda)(x_1 - x_3) - y_1
\end{aligned}
$$

Формулы сложения точек

$$
\begin{aligned}
\lambda &= \frac{y_2 - y_1}{x_2 - x_1} \\
x_3 &= \lambda^2 - a_2 - x_1 - x_2 \\
y_3 &= (\lambda)(x_1 - x_3) - y_1
\end{aligned}
$$

Алгоритм удвоения точек

$$
\begin{aligned}
y_1 z_1 &= y_1 \times z_1 \\
y_1^2 z_1 &= y_1 z_1 \times y_1 \\
a_2 z_1 &= a_2 \times z_1 \\
x_1^2 &= x_1 \times x_1 \\
a_2 x_1^2 z_1 &= a_2 z_1 \times x_1^2 \\
z_1^2 &= z_1 \times z_1 \\
a_4 z_1^2 &= a_4 \times z_1^2 \\
\lambda &= 3x_1^2 + a_2 x_1^2 z_1 + a_4 z_1^2 \\
\lambda^2 &= \lambda \times \lambda \\
y_1^2 z_1^4 &= 4 \times y_1^2 z_1 \\
y_1^3 z_1^2 &= y_1^2 z_1^4 \times y_1 z_1 \\
a_2 z_1 + 2x_1 &= a_2 z_1 + x_1 + x_1 \\
z_3 &= y_1^3 z_1^2 \times (z_1 + z_1) \\
x_3 &= -y_1^2 z_1^4 \times (a_2 z_1 + 2x_1) + \lambda^2 \\
y_3 &= y_1^2 z_1^4 \times (-(y_1^2 z_1 + y_1^2 z_1) + x_1 \times \lambda) - x_3 \times \lambda \\
x_3 &= x_3 \times (y_1 z_1 + y_1 z_1)
\end{aligned}
$$

В результате получили 10 сложений и 17 умножений.

Алгоритм сложения двух точек

$$
\begin{aligned}
z_1 z_2 &= z_1 \times z_2 \\
y_2 z_1 &= y_2 \times z_1 \\
y_1 z_2 &= y_1 \times z_2 \\
x_2 z_1 &= x_2 \times z_1 \\
x_1 z_2 &= x_1 \times z_2 \\
\lambda &= y_2 z_1 - y_1 z_2 \\
\delta &= x_2 z_1 - x_1 z_2 \\
\lambda^2 &= \lambda \times \lambda \\
\delta^2 &= \delta \times \delta \\
\delta' &= x_2 z_1 + x_1 z_2 \\
a_2 \delta^2 &= a_2 \times \delta^2 \\
x_3 &= z_1 z_2 \times (\lambda^2 - a_2 \delta^2) - \delta^2 \times \delta' \\
z_3 &= z_1 z_2 \times \delta^2 \times \delta \\
y_3 &= z_2 \times \delta^2 \times (-y_1 \times \delta + \lambda \times x_1) - \lambda \times x_3 \\
x_3 &= x_3 \times \delta
\end{aligned}
$$

В результате получили 7 сложений и 18 умножений.

Глава 2. Эндоморфизм эллиптической кривой

Пусть $d \in \mathbb{N}$ свободное от квадратов число, $\tau \in \mathbb{Q}(\sqrt{-d})$ - мнимая квадратичная иррациональность, принадлежащая верхней комплексной полуплоскости, и $\Lambda_\tau = \{n + m\tau, n, m \in \mathbb{Z}\}$ - решетка в $\mathbb{C}$. Известно, см. [10], что существует только 11 решеток Λ_τ с числом классов эквивалентных квадратичных форм, равным единице, и значением модулярной функции $j(\tau)$, отличным от нуля и 1728. Каждой такой решетке соответствует одна, с точностью до изоморфизма, эллиптическая кривая вида(2) с целым значением инварианта $j(E) = 1728\frac{4a^3}{4a^3+27b^2} \in \mathbb{Z}$ и кольцом эндоморфизмов, изоморфным Λ_τ. Для пяти из рассматриваемых кривых существует рациональный корень θ многочлена. Перечень таких кривых приводится в таблице 2, третий столбец таблицы содержит значение нормы величины τ, а в четвертый столбец - точное значение инварианта эллиптической кривой [2].

№	τ	$N(\tau)$	$j(E)$	E_τ	θ
1	$2\sqrt{-1}$	4	287496	$y^2 = x^3 - \frac{11}{4}x - \frac{7}{4}$	-1
2	$\sqrt{-2}$	2	8000	$y^2 = x^3 - \frac{15}{2}x - 7$	-2
3	$\sqrt{-3}$	3	54000	$y^2 = x^3 - \frac{15}{4}x - \frac{11}{4}$	-1
5	$\frac{1}{2}(1+\sqrt{-7})$	2	-3375	$y^2 = x^3 - \frac{35}{4}x - \frac{49}{4}$	$\frac{7}{2}$
6	$\sqrt{-7}$	7	16581375	$y^2 = x^3 - \frac{595}{4}x - \frac{2793}{4}$	-7

Таблица 2 — Эллиптические кривые с целым j-инвариантом и рациональным корнем θ Полный список из 11 кривых предоставлен в работе [2]

Наличие рационального корня θ позволяет построить отображения перечисленных эллиптических кривых в различные формы, и провести сравнение реализаций.

В данной работе рассмотрена эллиптическая кривая №3 в короткой форме Вейерштрасса:

$$y^2 = x^3 - \frac{15}{4}x - \frac{11}{4},$$

с параметрами $\theta = -1$ и $\delta = \sqrt{-3}$.

Форма Хадано для этой кривой при $\beta = 2$ выражается как:

$$v^2 = u^3 - 6u^2 - 3u.$$

Эндоморфизм кривой задаётся следующим образом:

$$\phi : (u, v) = \left(-\frac{u(u-3)^2}{3(u+1)^2}, -\frac{(u^3 + 3u^2 - 21u + 9)v}{3\alpha(u+1)^3} \right).$$

Алгоритм для проективных координат представляется следующим образом:

$$\begin{aligned}
xx1 &= x + z, \\
xx1xx1 &= xx1 \cdot xx1, \\
z_{13} &= 3 \cdot z, \\
xx3 &= x - z_{13}, \\
\delta &= \alpha \cdot xx1, \\
z_{\text{new}} &= z_{13} \cdot \delta \cdot xx1xx1, \\
x_{\text{new}} &= -\delta \cdot x \cdot xx3^2, \\
x1z_{13} &= x \cdot z_{13}, \\
y_{\text{new}} &= xx3 \cdot (x^2 + x1z_{13} + x1z_{13} - z_{13} \cdot z) \cdot y.
\end{aligned}$$

Получается 5 сложений и 13 умножений.

Глава 3. Алгоритмы возведения в кратную точку

Алгоритм add-and-double

Основным алгоритмом, используемым для получения кратной точки, является *add-and-double*. Этот алгоритм лежит в основе многих криптографических операций, таких как генерация ключей и создание цифровых подписей. Он применяет основные операции с эллиптическими кривыми: сложение и удвоение точек.

Алгоритм *add-and-double* выполняется следующим образом:

1) Имеется точка P на эллиптической кривой E, которую нужно умножить на число k, чтобы получить kP.

2) Число k представляется в бинарном виде. Например, для $k = 12$ бинарное представление будет 1100.

3) Процесс итерации по битам числа k (1 и 0):

   - Инициализируется точка на бесконечности N.
   - Для каждого бита, начиная со старшего, происходит удвоение текущей точки N. Если бит равен 1, к результату добавляется точка P.

Данный алгоритм эффективен для вычисления кратных точек на эллиптических кривых, сокращая количество необходимых операций по сравнению с простым последовательным сложением.

Алгоритм *add-and-double* не подходит для задачи вычисления кратной точки с эндоморфизмом в виде комплексного умножения по двум причинам:

- Комплексные числа не могут быть представлены в бинарном виде.
- В алгоритме не используется эндоморфизм.

Алгоритм add-and-map

В работе [2] предложен алгоритм, позволяющий представить число в виде многочлена из комплексного числа.

1) Задается элемент a, для которого определены $tr(a) = a + \bar{a}$ и $N(a) = a \cdot \bar{a}$.

2) Создается массив $wArray$, ограниченный сверху значением $\mathrm{Round}[2 \cdot \log N(a) \cdot k] + 4$ (согласно Теореме 1.6 [2]).

3) Устанавливаются начальные значения: $s_0 = k$, $s_1 = 0$, $w = 0$, $\theta(a) = a \mod 2$, $nAa = (N(a) - \theta(a))/2$.

4) Итерация по массиву:

   - Вычисляется $x_w = s_0 \mod nAa$.
   - Если $x_w > nAa$, то корректируется $x_w = x_w - N(a)$.
   - Увеличивается q на 1.
   - В массив $wArray$ на позицию w записывается значение x_w.
   - Обновляются $s_0 = q \cdot tr(a) + s_1$ и $s_1 = -q$.
   - Инкременируем w.

5) Алгоритм завершается, когда выполняются условия: $|s_0| < nAa$ и $s_1 = 0$.

Для сборки полученного массива для получения числа используется следующий алгоритм:

1) Исходная точка обозначается как P, а вторая точка Q инициализируется как O.

2) Итерация по каждому элементу массива:

   - Если элемент массива $\neq 0$, то $Q = Q + \text{element} \cdot R$.
   - Происходит обновление $Q = \phi(Q)$.

3) В результате получается последовательность, аналогичная той, что была использована для разбиения скаляра, только теперь она собрана в одно целое.

При этом эти алгоритмы не подходят для эллиптической кривой в форме Монтгомери с XZ координатами.

Алгоритм Montgomery Ladder

В 1987 году американский математик Питер Монтгомери [5] предложил алгоритм, который устраняет необходимость в использовании y-координаты для вычисления скалярного произведения точки на эллиптической кривой, при этом обеспечивая дополнительную защиту от атак, связанных с анализом энергопотребления.

Алгоритм "Лестницы Монтгомери"представляет собой метод ускоренного скалярного умножения точки P на эллиптической кривой по скаляру k. Он использует две временные переменные, Q_0 и Q_1.

1) Инициализация переменных: Устанавливаются начальные значения, включая $Q_0 = P$ и $Q_1 = 2P$.

2) Итерации: Цикл выполняется для каждого бита скаляра k (начиная с самого старшего бита).

   - Если бит равен 0, выполняется операция удвоения точки Q_0 и сложения точки Q_1.
   - Если бит равен 1, выполняется операция удвоения точки Q_1 и сложения точки Q_0.

Алгоритм "Лестницы Монтгомери"минимизирует количество операций умножения, что делает его эффективным для криптографических операций на эллиптических кривых, таких как создание открытых ключей и цифровые подписи.

Подсчет Q_0 и Q_1 на каждом шаге алгоритма является необходимым условием. Поскольку Монтгомери отказывается от использования y-координаты[9], необходимо иметь точный порядок действий для вычисления проективных координат на каждой итерации.

Глава 4. Построение эллиптической кривой и экспериментальные вычисления

Определение требований

В рамках построения эллиптических кривых с комплексным умножением существует общий подход, предложенный А.Ю. Нестеренко. В данной работе рассматривается специфический случай, когда коэффициенты a и b, эндоморфизм, а также число классов уже известны. Основная задача заключается в подборе такого простого числа p, которое удовлетворяет следующим условиям:

1) Условие для существования эндоморфизма в виде комплексного умножения: Символ Лежандра $(d|p)$ должен быть равен 1.

2) Условие для существования отображения в форму Хадано: Символ Лежандра $(\gamma|p)$ должен быть равен 1.

3) Условие для существования отображения в форму Скрученного Эдвардса: Из уравнения Корначи $x^2 + dy^2 = 4p$ находим x. Затем, определяем $m_\Delta = p + 1 + x$, и устанавливаем, что $m_\Delta = mq$, где m должен быть равен 4 для существования перевода в форму Скрученного Эдвардса, при этом q являться простым числом.

Рассмотрим данные условия на примере $d = -3, \gamma = 2$.

Символ Лежандра $(-3|p)$ определяется для нечетного простого p и целого a, и он равен 1, если a является квадратичным вычетом по модулю p. Для $d = -3$, мы хотим узнать, при каких условиях на p, -3 является квадратичным вычетом.

Квадратичный закон взаимности гласит, что для двух нечетных простых чисел p и q:

$$(p|q) = \begin{cases} (q|p), & \text{если одно из чисел } p \text{ или } q \equiv 1 \mod 4, \\ -(q|p), & \text{в противном случае.} \end{cases}$$

Свойства символа Лежандра:

- $(-1|p) = 1$, если $p \equiv 1 \mod 4$; -1, если $p \equiv 3 \mod 4$.
- $(3|p) = 1$, если $p \equiv 1 \mod 3$; -1, если $p \equiv -1 \mod 3$.

Таким образом, $(-3|p) = (-1|p) \times (3|p)$. Чтобы $(-3|p) = 1$, оба символа Лежандра должны быть либо оба равны 1, либо оба равны -1. Это достигается, когда p удовлетворяет одному из следующих условий:

- $p \equiv 1 \mod 12$, так как $p \equiv 1 \mod 4$ и $p \equiv 1 \mod 3$, что удовлетворяет обоим условиям.
- $p \equiv 7 \mod 12$, так как $p \equiv 3 \mod 4$ и $p \equiv -1 \mod 3$, что также удовлетворяет обоим условиям.

Аналогично для $\left(\frac{2}{p}\right)$. Он равен 1, если $p \equiv \pm 1 \mod 8$, и равен -1, если $p \equiv \pm 3 \mod 8$.

Следовательно, для того чтобы $(-3|p) = 1$ и $(2|p) = 1$, простое число p должно удовлетворять одному из условий: $p \equiv 1 \mod 24$ или $p \equiv 7 \mod 24$.

Следующим шагом в построении эллиптической кривой является решение уравнения $x^2 - d*y^2 = 4p$. Значение x, полученное из этого уравнения, используется для определения порядка кривой.

Алгоритм Корначи для решения уравнения. Алгоритм Корначи – это эффективный метод для решения квадратных уравнений вида $x^2 - Dy^2 = N$, где D и N – целые числа. [11]

Применение найденного значения x. Значение x, полученное из уравнения, используется для вычисления порядка эллиптической кривой через формулу $m_\Delta = p + 1 + x = m * q$ [2]. Это позволяет определить точное количество точек q на соответствующей эллиптической кривой.

Значение $m = 4$ и выбор простого числа q. Выбор $m = 4$ в формуле $m_\Delta = 4q$ обусловлен необходимостью обеспечения возможности перехода к форме Эдвардса эллиптической кривой. Выбор простого числа q критичен для обеспечения криптографической стойкости кривой.

Полученные результаты

В результате была построена эллиптическая кривая по указанным требованиям в размерности 256 бит.

Ключ	**Значение**
point.x	1
point.y	13701848152091537784126124568394138694167115365498299250854956305008813794 58
point.z	1
a	$2^{253} + 83372$
b	$2^{253} + 83373$
p	$2^{255} + 333503$
q	$2^{253} + 11923205279750705705744091938718965141 9$

Таблица 3 — Параметры построенной эллиптической кривой 256 бит

Эллиптическая кривая 256 бит.

Сравнение результатов количества действий в разных кривых на разных операциях:

Операция	**Сложения**	**Умножения**
Эндоморфизм в форме Хадано	5	13
Вейерштрасс (сложение точек)	7	14
Вейерштрасс (удвоение точки)	10	13
Монтгомери (сложение)	6	6
Монтгомери (удвоение)	4	5
Хадано (сложение)	7	18
Хадано (удвоение)	10	17
Скрученный Эдвардс (сложение)	6	13
Скрученный Эдвардс (удвоение)	6	9

Таблица 4 — Количество действий в операциях в разных формах эллиптических кривых

Для проведения эксперимента в рамках дипломной работы был использован следующий алгоритм:

1) Установить i в диапазоне от 2 до 74 (включительно).
2) Вычислить j как округленное значение логарифма по основанию 2 от 10^i, умноженное на 2.
3) Инициализировать пустой список $numbers$ для хранения чисел.
4) Для каждого значения j:
   - Сгенерировать случайное целое число x в диапазоне от 10^i до $10^{(i+1)-1}$.
   - Добавить x в список $numbers$.

Этот алгоритм использовался для подготовки данных для дальнейших сравнений. Он включал в себя генерацию случайных целых чисел в заданных диапазонах для различных степеней 10. Полученные числа использовались в дипломной работе для проведения экспериментов и анализа результатов.

Сравнение результатов подсчета количества умножений в каждой из форм

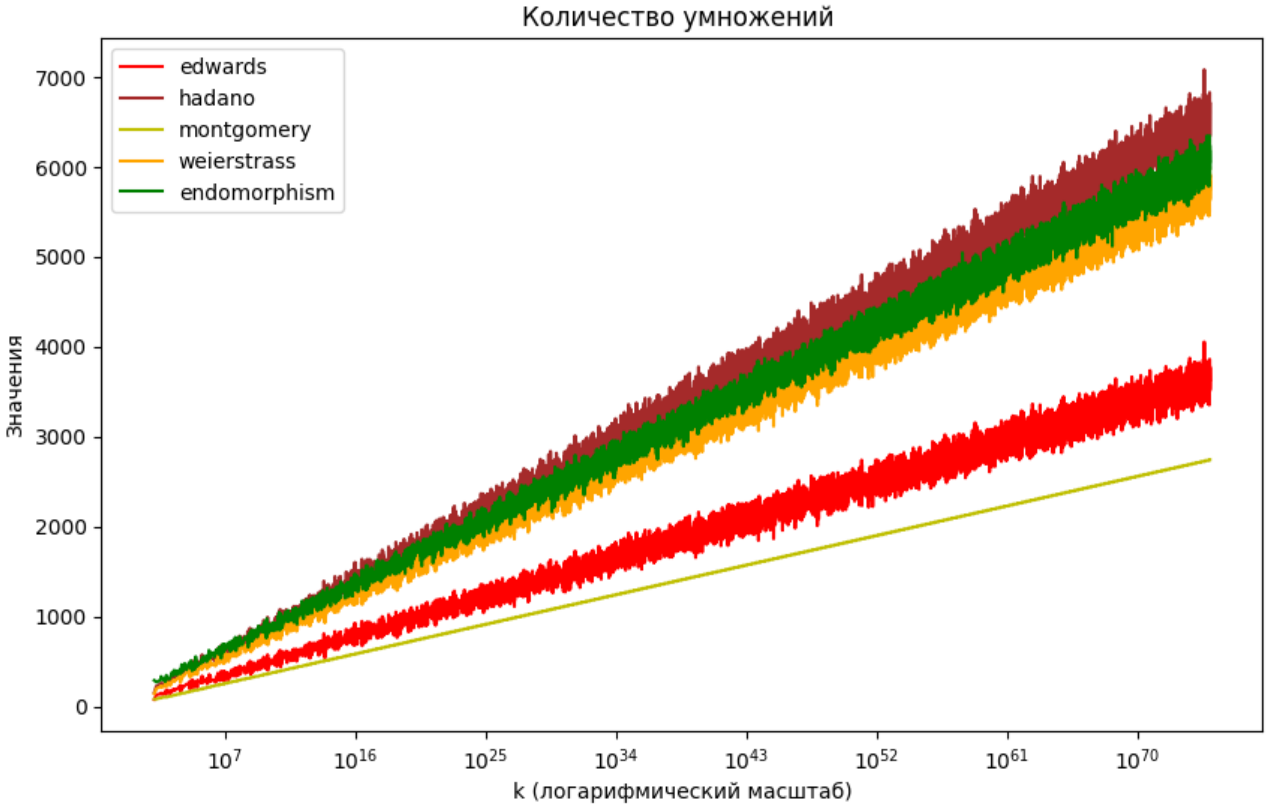

Рис. 1 — Связь количества умножений от формы кривой и выбранной кратной точки

Экспериментальные исследования показали, что среди выбранных форм, эллиптические кривые в форме Монтгомери и в форме Эдвардса имеют наименьшее количество умножений. Эллиптическая кривая в форме Хадано с изучаемым эндоморфизмом имеет незначительно больше умножений чем форма Вейештрасса.

В среднем, кривая в форме Монтгомери эффективнее кривой в форме Скрученного Эдвардса на 30%, эффективнее формы Вейештрасса на 110%, эффективнее формы Хадано с эндоморфизмом на 120% и эффективнее формы Хадано без эндоморфизма на 135%.

Сравнение результатов подсчета количества сложений в каждой из форм

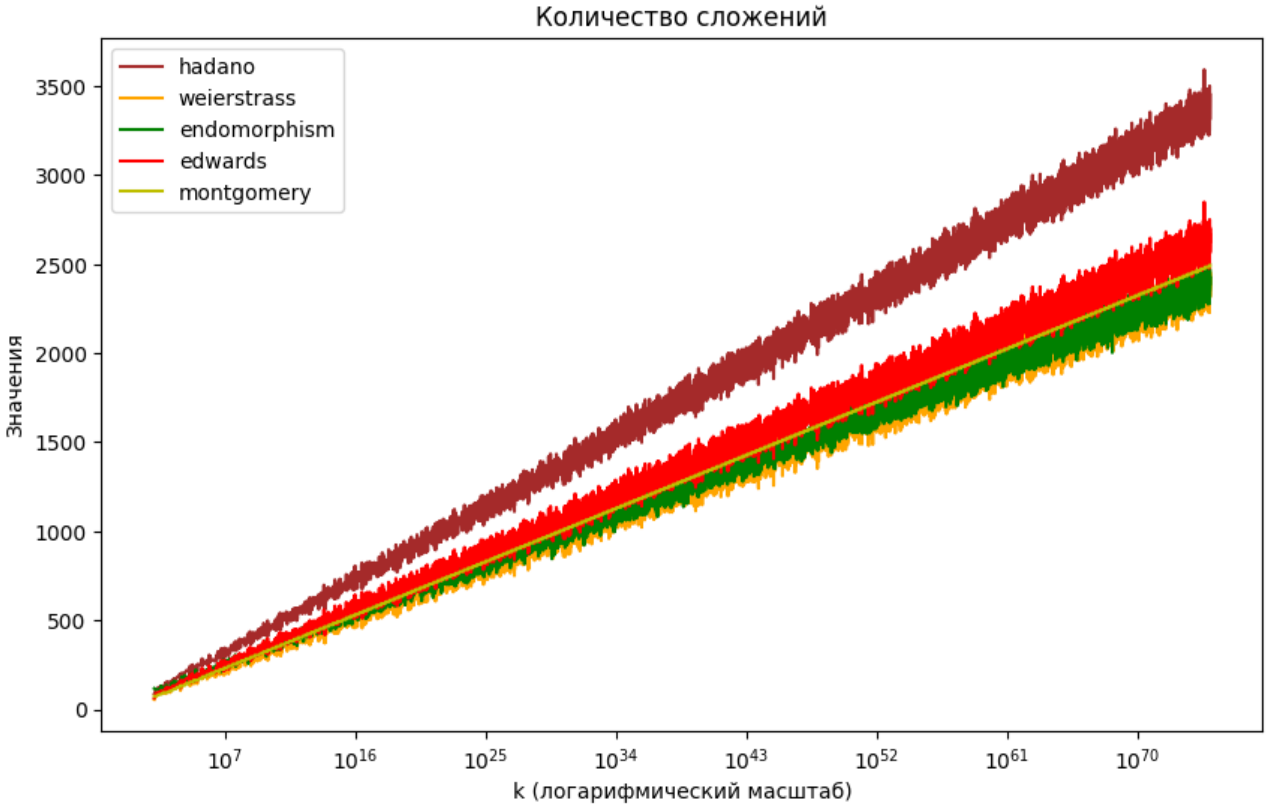

Рис. 2 — Связь количества сложений от формы кривой и выбранной кратной точки

Экспериментальные исследования показали, что среди выбранных форм, эллиптические кривые в форме Вейештрасса и в форме Хадано с эндоморфизмом имеют наименьшее количество сложений. Следом идут форма Монтгомери и форма Скрученного Эдвардса.

В среднем, кривая в форме Вейерштрасса эффективнее кривой в форме Скрученного Эдвардса на 10%, эффективнее формы Хадано на 40%, эффективнее формы Монтгомери на 5% и эффективнее формы Хадано с эндоморфимом на 1%. Для суммирования количества умножений и сложений было предложено

сопоставить 1 умножение как 1.5 сложения для простоты.

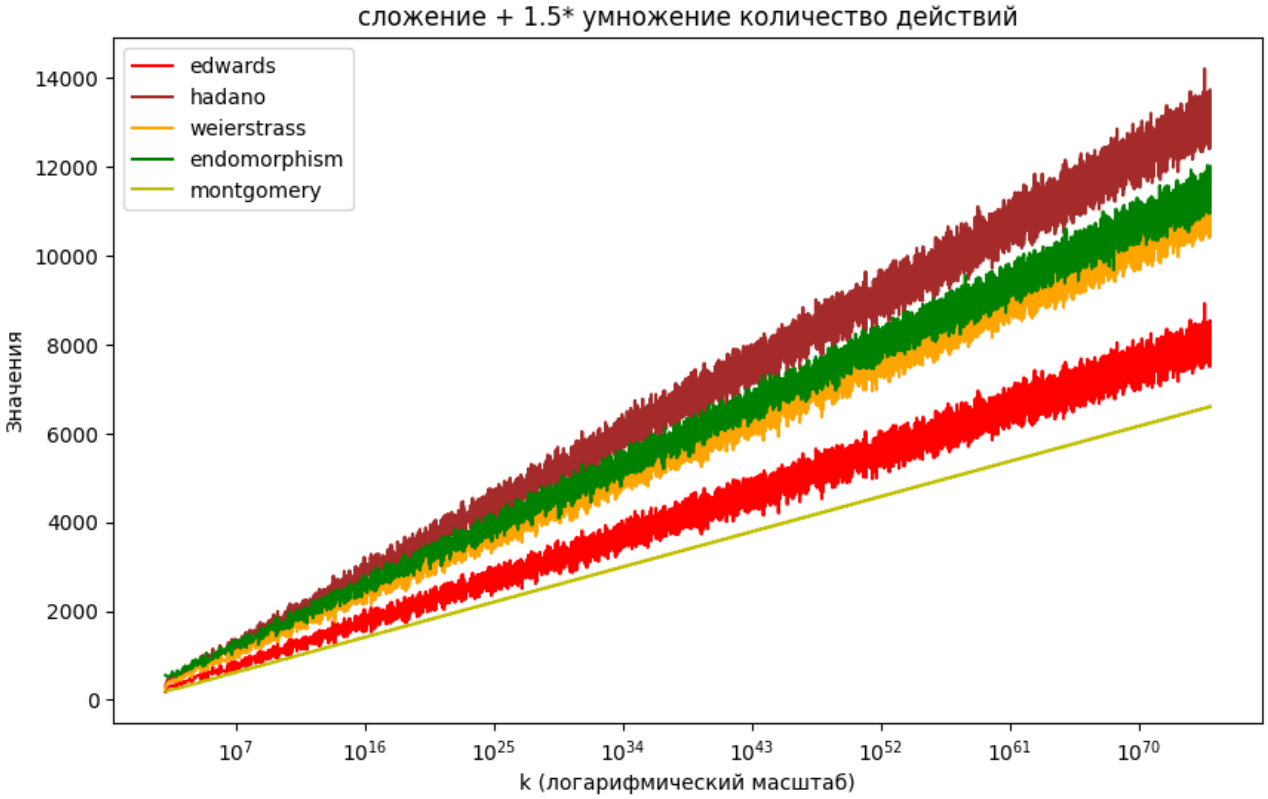

Рис. 3 — Связь количества умножений и сложений от формы кривой и выбранной кратной точки

Экспериментальные исследования показали, что среди выбранных форм, эллиптические кривые в форме Монтгомери и Эдвардса имеют суммарно наименьшее количество действий. Эллиптическая кривая в форме Вейештрасса и в форме Хадано с эндоморфизмом идут дальше и наименее эффективной является форма Хадано.

Сравнение результатов подсчета времени в каждой из форм (усредненное для логарифма по основанию 10)

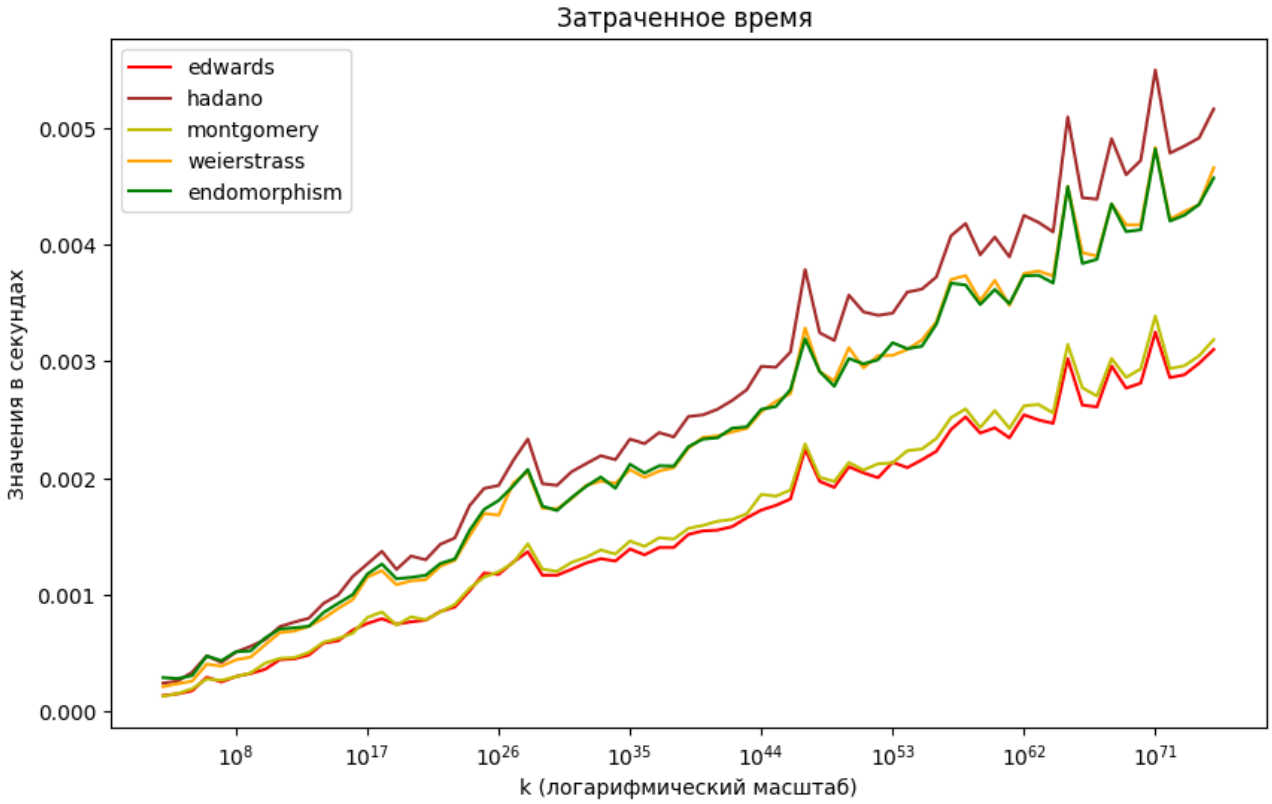

Рис. 4 — Время в секундах, потраченное на возведение в кратную точку

В результате наиболее эффективными формами эллиптических кривых относительно времени выполнения являются форма Монтгомери и форма Скрученного Эдвардса.

В среднем, кривая в форме Скрученного Эдвардса по времени эффективнее кривой в форме Хадано на 65%, эффективнее формы Монтгомери на 2%, эффективнее формы Хадано с эндоморфизмом на 50% и эффективнее формы Вейерштрасса на 47%.

Заключение

Данная выпускная квалификационная работа посвящена построение и исследованию эффективности операций в разных формах эллиптических кривых с использованием языков программирования Python и SageMath. Внимание уделено анализу количества операций, необходимых для вычисления кратных точек, и изучению влияния эндоморфизма на эффективность этих алгоритмов.

В рамках работы построена эллиптическая кривая с комплексным умножением, реализованы алгоритмы поиска кратной точки в 4 разных формах эллиптических кривых, подсчитаны количество операций на каждой форме кривой и проведен анализ.

На основе проведенных исследований стало ясно, что наиболее эффективной реализацией среди представленных форм является форма Монтгомери. Эта форма обладает минимальным количеством операций сложения и умножения под XZ координаты, что делает ее привлекательной для использования в цифровой подписи и шифровании с обменом ключей. Для вычисления y-координаты предпочтительной является форма Скрученного Эдвардса. Форма Хадано с эндоморфизмом эффективнее формы Хадано без эндоморфизма в количестве сложений и умножений, но есть более эффективные формы.

В дальнейшем планируется развитие исследования с целью поиска и оптимизации других методов вычисления кратной точки, а также сравнительный анализ эффективности различных форм эллиптических кривых и их эндоморфизмов. Это позволит способствовать повышению производительности практических криптографических систем.

Список использованных источников

1. *А.Б. Лось А.Ю. Нестеренко М. Р.* Криптографические методы защиты информации //. — 2016.
2. Математические методы обеспечения защищенного взаимодействия средств защиты информации. — URL: https://dissovet.msu.ru/download/4cab3142-14d6-11ee-99c4-005056b96f20 (дата обр. 15.01.2024).
3. *С. Л.* Эллиптические функции //. — 1984.
4. Projective coordinates for short Weierstrass curves [электронный ресурс]. — URL: https://www.hyperelliptic.org/EFD/g1p/auto-shortw-projective.html (дата обр. 15.01.2024).
5. *P.L. M.* Speeding the Pollard and elliptic curve methods of factorization //. — 1987.
6. XZ coordinates for Montgomery curves [электронный ресурс]. — URL: https://www.hyperelliptic.org/EFD/g1p/auto-montgom-xz.html (дата обр. 15.01.2024).
7. *D. Bernstein P. Birkner M. J. e. a.* Twisted Edwards Curve [электронный ресурс]. — URL: https://eprint.iacr.org/2008/013.pdf (дата обр. 15.01.2024).
8. Projective coordinates for twisted Edwards curves [электронный ресурс]. — URL: https://www.hyperelliptic.org/EFD/g1p/auto-twisted-projective.html (дата обр. 15.01.2024).
9. *T. H.* Conductor of Elliptic Curves with Complex Multiplication and Elliptic Curves with Prime Conductor //. — 1975.
10. *Под ред. Дж. Касселс А. Ф.* Алгебраическая теория чисел //. — МИР. 1969.
11. *H. C.* Course In Computational Algebraic Number Theory //. — 1996.

Построение эллиптической кривой

```
1
2  def ecIsSafePrime(p, endo, gamma):
3      if not is_prime(p):
4          return False
5      if kronecker_symbol(endo, p) != 1:
6          return False
7      if kronecker_symbol(gamma, p) != 1:
8          return False
9      return True
10
11 def ecCreateBasePoint(a, b, p):
12     K = GF(p)
13     ec = EllipticCurve([K(a), K(b)])
14     for x in range(1, 1010):
15         y = K(x**3 + a*x + b)
16         if kronecker_symbol(y, p) == 1:
17             try:
18                 P = ec.lift_x(K(x), all=False)
19                 return P
20             except:
21                 pass
22     return False
23
24 def norm_equation(D, p):
25     assert D < 0 and D\%4 in (0,1) and is_pseudoprime(p)
26     if -D >= 4*p: return ()
27     if p==2: return (ZZ(sqrt(D+8)),1) if is_square(D+8) else ()
28     if kronecker(D,p) == -1: return ()
29     F=GF(p,proof=false)
30     x0=F(D).sqrt().lift()
31     assert not (x0^2 - D) \% p
32     if (x0-D)\%2: x0 = p - x0
33     a,t,m = 2*p, x0, integer_floor(2*sqrt(p))
34     while t > m: a,t=t,a\%t
35     if (4*p-t^2) \% (-D): return ()
36     c = (4*p-t^2)//(-D)
37     if not is_square(c): return ()
38     v=sqrt(c)
39     assert 4*p == t^2-v^2*D
40     return (t,v)
41
42 results = []
43 endo = -3
44 gamma = 2
45 aClear = -15/4
46 bClear = -11/4
47 p=2**255
48 while True:
49     p=p.next_prime()
50     if not ecIsSafePrime(p, endo, gamma):
51         continue
52     K = GF(p)
53     a = K(aClear)
54     b = K(bClear)
55     ec = EllipticCurve([a, b])
56     try:
```

```
    resultNormEquation = norm_equation(endo, p)
    if len(resultNormEquation) != 2:
        continue
    m_delta = p + 1 + abs(resultNormEquation[0])
    m__delta = p + 1 - abs(resultNormEquation[1])
    if m_delta %4 == 0 :
        m_delta_factored = factor(m_delta)
    else:
        m_delta_factored = []
    if m__delta %4 == 0 :
        m__delta_factored = factor(m__delta)
    else:
        m__delta_factored = []
    if len(m_delta_factored) == 2:
        pass
    elif len(m__delta_factored) == 2:
        pass
    else:
        continue
    point = ecCreateBasePoint(a, b, p)
    if(point):
        results.append({
        "point.x": point[0],
        "point.y": point[1],
        "point.z": point[2],
        "a": a,
        "b": b,
        "p": p,
        "m_delta": m_delta,
        "m__delta": m__delta
        })
    if len(results) > 0:
        break
    except e:
        pass
```

Формулы сложения и удвоения точек в разных формах

```
1
2  def add(a, b):
3    global additions
4    additions += 1
5    return a+b
6  def mul(a, b):
7    global multiplications
8    multiplications += 1
9    return a*b
10 def addTwoPointsWeierstrass(a, b, x1, y1, z1, x2, y2, z2):
11   if(x1==x2 and y1==y2):
12     w=add(mul(a,mul(z1,z1)),mul(3, mul(x1,x1)))
13     s=mul(y1,z1)
14     ss=mul(s, s)
15     sss = mul(s, ss)
16     r=mul(y1, s)
17     b=mul(x1, r)
18     b4=mul(4, b)
19     h=add(mul(w, w), -add(b4, b4))
20     x3=mul(h, s)
21     x3=add(x3,x3)
22     y3=add(mul(w, add(b4, -h)),-mul(8, mul(r,r)))
23     z3=mul(8, sss)
24   else:
25     y1z1=mul(y1,z2)
26     x1z2=mul(x1,z2)
27     z1z2=mul(z1,z2)
28     u=add(mul(y2,z1),-y1z2)
29     uu=mul(u,u)
30     v=add(mul(x2,z1),-x1z2)
31     vv=mul(v,v)
32     vvv=mul(v, vv)
33     R=mul(vv, x1z2)
34     A=add(mul(uu, z1z2),-add(vvv, add(R, R)))
35     x3=mul(v, A)
36     y3=add(mul(u, add(R, -A)), -mul(vvv, y1z2))
37     z3=mul(vvv, z1z2)
38   return x3, y3, z3
39
40 def addTwoPointsEdwards(e, d, x1, y1, z1, x2, y2, z2):
41   if(x1==x2 and y1==y2 and z1 == z2):
42     B = add(x1,y1)
43     B = mul(B, B)
44     C = mul(x1, x1)
45     D = mul(y1, y1)
46     E = mul(e*C)
47     F = add(E, D)
48     H = mul(z1, z1)
49     J = add(F, -add(H, H))
50     x3 = mul(add(B, -add(C, D)), J)
51     y3 = mul(F, add(E, -D))
52     z3 = mul(F, J)
53   else:
54     A = mul(z1, z2)
55     B = mul(A, A)
56     C = mul(x1, x2)
```

```
    D = mul(y1, y2)
    E = mul(d, mul(C, D))
    F = add(B, -E)
    G = add(B, E)
    x3 = mul(mul(A, F), add(mul(add(x1, y1), add(x2, y2)), -add(C, D)))
    y3 = mul(mul(A, G), add(D, -mul(e, C)))
    z3 = mul(F, G)
  return x3, y3, z3

def addTwoPointsHadano(a, b, x1, y1, z1, x2, y2, z2):
  if(x1==x2 and y1==y2 and z1 == z2):
    y1z1 = mul(y1, z1)
    y12z1 = mul(y1z1, y1)
    az1 = mul(a, z1)
    x12 = add(x1, x1)
    ax12z1 = mul(az1, x12)
    z12 = mul(z1, z1)
    bz12 = mul(b, z12)
    delta = add(add(mul(3, mul(x1, x1)), ax12z1), bz12)
    delta2 = mul(delta, delta)
    y12z14 = mul(4, y12z1)
    y13z12 = mul(y12z14, y1z1)
    az1p2x = add(az1, x12)
    z3 = mul(y13z12, add(z1, z1))
    x3 = add(-mul(y12z14, az1p2x), delta2)
    y3 = add(mul(y12z14, add(-add(y12z1, y12z1),
                             mul(x1, delta))), -mul(x3, delta))
    x3 = mul(x3, add(y1z1,y1z1))
  else:
    z1z2 = mul(z1, z2)
    y2z1 = mul(y2, z1)
    y1z2 = mul(y1, z2)
    x2z1 = mul(x2, z1)
    x1z2 = mul(x1, z2)
    llambda = add(y2z1, -y1z2)
    delta = add(x2z1, -x1z2)
    llambda2 = mul(llambda, llambda)
    delta2 = mul(delta, delta)
    deltap = add(x2z1, x1z2)
    adelta2 = mul(a, delta2)
    x3 = add(mul(z1z2, add(llambda2, -a2delta2)), -mul(delta2, deltap))
    z3 = mul(mul(z1z2, delta2), delta)
    y3 = add(mul(mul(z2, delta2), add(-mul(y1, delta),
                        mul(llambda, x1))), -mul(llambda, x3))
    x3 = mul(x3, delta)
  return x3, y3, z3

def endomorphismHadano(alpha, x, y, z):
  xx1 = add(x, K(1))
  xx1xx1 = mul(xx1, xx1)
  z13 = mul(K(3), z)
  xx3 = add(x, -z13)
  delta = mul(alpha, xx1)
  znew = mul(mul(z13,delta), xx1xx1)
  xnew = -mul(mul(delta, x), mul(xx3, xx3))
  x1z13 = mul(x, z13)
  ynew = mul(mul(xx3, add(add(mul(x, x),
                              add(x1z13, x1z13)), mul(z13, z))), y)
  return xnew, ynew, znew

def montgomeryAddPoints(x1, z1, x2, z2, mx1, mz1):
```

```
  a = add(x1,-z1);
  b = add(x1, z1);
  c = add(x2,-z2);
  d = add(x2, z2)
  e = mul(a,d);
  f = mul(b, c);
  epf = add(e, f)
  emf = add(e, -f)
  x3 = mul(mz1, mul(epf, epf))
  z3 = mul(mx1, mul(emf, emf))
  return [x3, z3]

def montgomeryDoublePoint(x, z, C):
  a=add(x, z);
  b=add(x, -z);
  c=mul(a, a);
  d=mul(b, b);
  e=add(c, -d)
  x3 = mul(c, d)
  z3 = mul(e, add(d, mul(C, e)))
  return [x3, z3]
```

Формулы сложения и удвоения точек в разных формах

```
1
2  def add(a, b):
3    global additions
4    additions += 1
5    return a+b
6  def mul(a, b):
7    global multiplications
8    multiplications += 1
9    return a*b
10 def addTwoPointsWeierstrass(a, b, x1, y1, z1, x2, y2, z2):
11   if(x1==x2 and y1==y2):
12     w=add(mul(a,mul(z1,z1)),mul(3, mul(x1,x1)))
13     s=mul(y1,z1)
14     ss=mul(s, s)
15     sss = mul(s, ss)
16     r=mul(y1, s)
17     b=mul(x1, r)
18     b4=mul(4, b)
19     h=add(mul(w, w), -add(b4, b4))
20     x3=mul(h, s)
21     x3=add(x3,x3)
22     y3=add(mul(w, add(b4, -h)),-mul(8, mul(r,r)))
23     z3=mul(8, sss)
24   else:
25     y1z1=mul(y1,z2)
26     x1z2=mul(x1,z2)
27     z1z2=mul(z1,z2)
28     u=add(mul(y2,z1),-y1z2)
29     uu=mul(u,u)
30     v=add(mul(x2,z1),-x1z2)
31     vv=mul(v,v)
32     vvv=mul(v, vv)
33     R=mul(vv, x1z2)
34     A=add(mul(uu, z1z2),-add(vvv, add(R, R)))
35     x3=mul(v, A)
36     y3=add(mul(u, add(R, -A)), -mul(vvv, y1z2))
37     z3=mul(vvv, z1z2)
38   return x3, y3, z3
39
40 def addTwoPointsEdwards(e, d, x1, y1, z1, x2, y2, z2):
41   if(x1==x2 and y1==y2 and z1 == z2):
42     B = add(x1,y1)
43     B = mul(B, B)
44     C = mul(x1, x1)
45     D = mul(y1, y1)
46     E = mul(e*C)
47     F = add(E, D)
48     H = mul(z1, z1)
49     J = add(F, -add(H, H))
50     x3 = mul(add(B, -add(C, D)), J)
51     y3 = mul(F, add(E, -D))
52     z3 = mul(F, J)
53   else:
54     A = mul(z1, z2)
55     B = mul(A, A)
56     C = mul(x1, x2)
```

```
    D = mul(y1, y2)
    E = mul(d, mul(C, D))
    F = add(B, -E)
    G = add(B, E)
    x3 = mul(mul(A, F), add(mul(add(x1, y1), add(x2, y2)), -add(C, D)))
    y3 = mul(mul(A, G), add(D, -mul(e, C)))
    z3 = mul(F, G)
  return x3, y3, z3

def addTwoPointsHadano(a, b, x1, y1, z1, x2, y2, z2):
  if(x1==x2 and y1==y2 and z1 == z2):
    y1z1 = mul(y1, z1)
    y12z1 = mul(y1z1, y1)
    az1 = mul(a, z1)
    x12 = add(x1, x1)
    ax12z1 = mul(az1, x12)
    z12 = mul(z1, z1)
    bz12 = mul(b, z12)
    delta = add(add(mul(3, mul(x1, x1)), ax12z1), bz12)
    delta2 = mul(delta, delta)
    y12z14 = mul(4, y12z1)
    y13z12 = mul(y12z14, y1z1)
    az1p2x = add(az1, x12)
    z3 = mul(y13z12, add(z1, z1))
    x3 = add(-mul(y12z14, az1p2x), delta2)
    y3 = add(mul(y12z14, add(-add(y12z1, y12z1),
                             mul(x1, delta))), -mul(x3, delta))
    x3 = mul(x3, add(y1z1,y1z1))
  else:
    z1z2 = mul(z1, z2)
    y2z1 = mul(y2, z1)
    y1z2 = mul(y1, z2)
    x2z1 = mul(x2, z1)
    x1z2 = mul(x1, z2)
    llambda = add(y2z1, -y1z2)
    delta = add(x2z1, -x1z2)
    llambda2 = mul(llambda, llambda)
    delta2 = mul(delta, delta)
    deltap = add(x2z1, x1z2)
    adelta2 = mul(a, delta2)
    x3 = add(mul(z1z2, add(llambda2, -a2delta2)), -mul(delta2, deltap))
    z3 = mul(mul(z1z2, delta2), delta)
    y3 = add(mul(mul(z2, delta2), add(-mul(y1, delta),
                        mul(llambda, x1))), -mul(llambda, x3))
    x3 = mul(x3, delta)
  return x3, y3, z3

def endomorphismHadano(alpha, x, y, z):
  xx1 = add(x, K(1))
  xx1xx1 = mul(xx1, xx1)
  z13 = mul(K(3), z)
  xx3 = add(x, -z13)
  delta = mul(alpha, xx1)
  znew = mul(mul(z13,delta), xx1xx1)
  xnew = -mul(mul(delta, x), mul(xx3, xx3))
  x1z13 = mul(x, z13)
  ynew = mul(mul(xx3, add(add(mul(x, x),
                              add(x1z13, x1z13)), mul(z13, z))), y)
  return xnew, ynew, znew

def montgomeryAddPoints(x1, z1, x2, z2, mx1, mz1):
```

```
  a = add(x1,-z1);
  b = add(x1, z1);
  c = add(x2,-z2);
  d = add(x2, z2)
  e = mul(a,d);
  f = mul(b, c);
  epf = add(e, f)
  emf = add(e, -f)
  x3 = mul(mz1, mul(epf, epf))
  z3 = mul(mx1, mul(emf, emf))
  return [x3, z3]

def montgomeryDoublePoint(x, z, C):
  a=add(x, z);
  b=add(x, -z);
  c=mul(a, a);
  d=mul(b, b);
  e=add(c, -d)
  x3 = mul(c, d)
  z3 = mul(e, add(d, mul(C, e)))
  return [x3, z3]
```

Изогении между формами кривых

```
1
2  def convertWeierstrassToHadanoCoeffs(a, b, gamma):
3      R.<x> = PolynomialRing(K, 'x')
4      equation = x^3 + a*x + b
5      roots = equation.roots(multiplicities=False)
6      theta = roots[0]
7      newA = 3*theta*gamma
8      newB = (3*theta^2 + a) * gamma^2
9      return newA, newB
10
11 def convertWeierstrassToHadanoPoints(a, b, gamma, xWeier, yWeier):
12     R.<x> = PolynomialRing(K, 'x')
13     equation = x^3 + a*x + b
14     roots = equation.roots(multiplicities=False)
15     theta = roots[0]
16     xHadano = gamma * (xWeier-theta)
17     yHadano = (gamma.sqrt() ** 3) * yWeier
18     return xHadano, yHadano
19
20 def convertMontgomeryToEdwardsCoeffs(gamma, mu):
21     e = (mu + 2)/gamma
22     d = (mu - 2)/gamma
23     return e, d
24
25 def convertMontgomeryToEdwardsPoints(gamma, mu, x, y):
26     xEdwards = x/y
27     yEdwards = (x-1)/(x+1)
28     return xEdwards, yEdwards
29
30 def convertEdwardsToMontgomeryCoeffs(e, d):
31     mu = 2 * (e + d) / (e - d)
32     gamma = 4 / (e - d)
33     return gamma, mu
34
35 def convertEdwardsToMontgomeryPoints(e, d, x, y):
36     xMont = (1 + y) / (1 - y)
37     yMont = xMont/x
38     return xMont, yMont
39
40 def convertWeierStrassToMontgomeryCoeffs(a, b):
41     R.<x> = PolynomialRing(K, 'x')
42     equation = x^3 + a*x + b
43     roots = equation.roots(multiplicities=False)
44     theta = roots[0]
45     gamma = sqrt(3*theta^2+a)
46     mu=3*theta/gamma
47     return gamma, mu
48
49 def convertWeierStrassToMontgomeryPoints(a, b, x, y):
50     R.<x> = PolynomialRing(K, 'x')
51     equation = x^3 + a*x + b
52     roots = equation.roots(multiplicities=False)
53     theta = roots[0]
54     gamma = sqrt(3*theta^2+a)
55     xMont = (xWeier-theta)/gamma
56     yMont = yWeier / gamma^2
```

```
57      return xMont, yMont
58
59  def convertMontgomeryToWeierstrassCoeffs(gamma, mu):
60      theta = gamma * mu / 3
61      a = gamma^2 - 3*theta^2
62      b = -(theta^3 + a*theta)
63      return a, b
64
65  def convertMontgomeryToWeierstrassPoints(gamma, mu, x, y):
66      theta = gamma * mu / 3
67      xWeier = gamma * x + theta
68      yWeier = gamma^2 * y
69      return xWeier, yWeier
```

Алгоритмы вычисления кратной точки

```
1
2  def tr(a):
3      return (a + conjugate(a)).real().n(digits=1024)
4
5  def nA(a):
6      return (conjugate(a) * a).real().n(digits=1024)
7
8  def calculateW(a, k):
9      na = nA(a)
10     trA = tr(a)
11     sigmaA = na % 2
12     nAa = (na - sigmaA) // 2
13     s0 = k
14     s1 = 0
15     wArray = [0] * (round(2 * log(k) / log(na)) + 5)
16     w = 0
17
18     while True:
19     q = s0 // na
20     xW = s0 % na
21     if xW > nAa:
22         xW = xW - na
23         q += 1
24     wArray[w] = round(xW.real())
25     s0 = round((q * trA + s1).real())
26     s1 = -q
27     w += 1
28     if abs(s0) < nAa and s1 == 0:
29         break
30
31     return wArray
32
33
34 def multiplyPointStandardVersion(Px, Py, Pz, k, a, b, func):
35     binary = Integer(k).digits(base=2)[::-1]
36     resx, resy, resz = 0, 1, 0
37     px, py, pz = Px, Py, Pz
38
39     for digit in binary:
40     if digit == 1:
41         resx, resy, resz = func(resx, resy, resz, px, py, pz, a, b)
42     px, py, pz = func(px, py, pz, px, py, pz, a, b)
43
44     return resx, resy, resz
45
46
47 def multiplyPointEndoVersion(Px, Py, Pz, k, a, b, func1, func2, AComplex, p):
48     binary = calculateW(AComplex, k)
49     resx, resy, resz = 0, 1, 0
50     px, py, pz = Px, Py, Pz
51
52     for digit in binary:
53     if digit == 1:
54         resx, resy, resz = func1(resx, resy, resz, px, py, pz, a, b)
55     if digit == -1:
56         tmpy = add(p, -py)
```

```
57          resx, resy, resz = func1(resx, resy, resz, px, tmpy, pz, a, b)
58      px, py, pz = func2(px, py, pz, a, b)
59
60      return resx, resy, resz
61
62  def montgomeryLadder(x, z, n, mu):
63      Q = [[x, z], mdbl(x, z, C)]
64      b = n.digits(2)
65      C = (mu + 2) / 4
66      for i in range(len(b)-2,-1,-1):
67          Q[1-b[i]] = montgomeryAddPoints(Q[1][0], Q[1][1], Q[0][0]. Q[0][1], x, z)
68          Q[b[i]] = montgomeryDoublePoint(Q[b[i]][0], Q[b[i]][1],C)
69      return Q[0][0], Q[0][1]
```

Построение граффиков

```
import pandas as pd
import json
import matplotlib.pyplot as plt
import random
import seaborn as sns

results = []

for i in range(2, 75):
    j = round(log(10**i, 2)*1.5)
    numbers = [random.randint(10**i, 10**(i+1) - 1) for _ in range(j)]
    for number in numbers:
        additions = 0
        multiplications = 0

        multiplyPointStandardVersion(xWeier, yWeier, zWeier, number, aWeier, bWeier,
            addTwoPointsWeierstrass)

        weierAdditions = additions
        weierMultiplications = multiplications

        additions = 0
        multiplications = 0

        montgomeryLadder(xMont, zMont, number, muMont)

        montAdditions = additions
        montMultiplications = multiplications

        additions = 0
        multiplications = 0

        multiplyPointStandardVersion(xHadano, yHadano, zHadano, number, aHadano, bHadano,
            addTwoPointsHadano)

        hadanoAdditions = additions
        hadanoMultiplications = multiplications

        additions = 0
        multiplications = 0

        multiplyPointStandardVersion(xEdwards, yEdwards, zEdwards, number, eEdwards, dEdwards,
            addTwoPointsEdwards)

        edwardsAdditions = additions
        edwardsMultiplications = multiplications

        additions = 0
        multiplications = 0

        multiplyPointEndoVersion(xHadano, yHadano, zHadano, number, aHadano, bHadano, addTwoPointsHadano,
            endomorphismHadano, sqrt(-3), p)
        endoAddition = additions
        endoMultiplications = multiplications

```

```
        results.append({'number': str(number), "endoMultiplications": str(endoMultiplications), "
            endoAdditions": str(endoAddition),
                    "weierAdditions": str(weierAdditions), "weierMultiplications": str(
                        weierMultiplications),
                    "montAdditions": str(montAdditions), "montMultiplications": str(montMultiplications),
                    "hadanoAdditions": str(hadanoAdditions), "hadanoMultiplications": str(
                        hadanoMultiplications),
                    "edwardsAdditions": str(edwardsAdditions), "edwardsMultiplications": str(
                        edwardsMultiplications)})



with open('resultsWithEndo.json', 'w') as f:
    json.dump(results, f)
df = pd.DataFrame.from_dict(json.load(open("resultsWithEndo.json")))
for col in df.columns:
  df[col] = df[col].astype(float)

plt.figure(figsize=(10, 6))
sns.lineplot(x=df['k'], y=df['hadanoAdditions'], label='hadano', color='brown')
sns.lineplot(x=df['k'], y=df['weierAdditions'], label='weierstrass', color='orange')
sns.lineplot(x=df['k'], y=df['edwardsAdditions'], label='edwards', color='r')
sns.lineplot(x=df['k'], y=df['endoAdditions'], label='endomorphism', color='g')
sns.lineplot(x=df['k'], y=df['montAdditions'], label='montgomery', color='y')
plt.xscale('log')
plt.show()

plt.figure(figsize=(10, 6))

sns.lineplot(x=df['k'], y=df['edwardsMultiplications'], label='edwards', color='r')
sns.lineplot(x=df['k'], y=df['hadanoMultiplications'], label='hadano', color='brown')
sns.lineplot(x=df['k'], y=df['montMultiplications'], label='montgomery', color='y')
sns.lineplot(x=df['k'], y=df['weierMultiplications'], label='weierstrass', color='orange')
sns.lineplot(x=df['k'], y=df['endoMultiplications'], label='endomorphism', color='g')
plt.xscale('log')
plt.show()

intervals = [(10 ** i, 10 ** (i + 1)) for i in range(2, 76)]

result_df = pd.DataFrame(columns=['interval', 'averageEdwardsTime', 'averageHadanoTime', 'averageMontTime',
      'averageWeierTime', 'averageEndoTime'])

for interval in intervals:
  lower_bound, upper_bound = interval
  interval_data = df[(df['k'] >= lower_bound) & (df['k'] < upper_bound)]
  average_value_edwards = interval_data['edwardsTime'].mean()
  average_value_hadano = interval_data['hadanoTime'].mean()
  average_value_mont = interval_data['montTime'].mean()
  average_value_weier = interval_data['weierTime'].mean()
  average_value_endo = interval_data['endoTime'].mean()
  result_df = result_df.append({'interval': upper_bound, 'averageEdwardsTime': average_value_edwards, '
       averageHadanoTime': average_value_hadano, 'averageMontTime': average_value_mont, 'averageWeierTime':
        average_value_weier, 'averageEndoTime': average_value_endo}, ignore_index=True)

plt.figure(figsize=(10, 6))

sns.lineplot(x=result_df['interval'], y=result_df['averageEdwardsTime'], label='edwards', color='r')
sns.lineplot(x=result_df['interval'], y=result_df['averageHadanoTime'], label='hadano', color='brown')
sns.lineplot(x=result_df['interval'], y=result_df['averageMontTime'], label='montgomery', color='y')
sns.lineplot(x=result_df['interval'], y=result_df['averageWeierTime'], label='weierstrass', color='orange')
sns.lineplot(x=result_df['interval'], y=result_df['averageEndoTime'], label='endomorphism', color='g')
```

```
106 plt.xscale('log')
107 plt.show()
```