МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки / специальности)

Компьютерные технологии и математическое моделирование
(направленность (профиль)/специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Исследование и реализация криптографических алгоритмов на эллиптических кривых»

Обучающийся П.П. Сидоров _____
(Инициалы Фамилия) (личная подпись)

Руководитель к.ф.-м.н., доцент, Г.А. Тырыгина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант Е.В. Косс
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2022

Аннотация

Тема бакалаврской работы – “Исследование и реализация криптографических алгоритмов на эллиптических кривых”.

Данная работа включает в себя введение, три главы, заключение и список используемой литературы.

Данная дипломная работа состоит из пояснительной записки на 40 страниц, введения, включая 29 рисунков, 11 формул, списка 20 источников, в том числе 9 источников на иностранном языке и одного приложения.

Актуальность темы заключается в том, что использование эллиптической криптографии обеспечивает меньшую нагрузку на сеть и вычислительные мощности, что особенно важно для устройств с ограниченными возможностями хранения и обработки.

Задачами бакалаврской работы являются описание математической теории эллиптических кривых, исследование алгоритмов криптографии на эллиптических кривых, и реализация криптографических алгоритмов на эллиптических кривых.

Целью бакалаврской работы является реализация криптографических алгоритмов на основе эллиптических кривых.

Объектом исследования бакалаврской работы являются эллиптические кривые.

Предметом исследования бакалаврской работы являются криптографические алгоритмы на эллиптических кривых.

В первой главе описывается математическая теория эллиптических кривых, применяющаяся в криптографических алгоритмах.

Во второй главе происходит исследование криптографических алгоритмов на эллиптических кривых, реализующих обмен ключами, цифровуя подпись и шифрование.

В третьей главе показана программная реализация эллиптических кривых на языке C++, а также реализация алгоритма ECDH.

Abstract

The title of the graduation work is “Research and implementation of cryptographic algorithms on elliptic curves”.

This work includes an introduction, three parts, a conclusion and a list of references.

The graduation work consists of an explanatory note on 40 pages, introduction, including 29 figures, 11 formulas, the list of 20 references including 9 foreign sources and one appendix.

The relevance of the research is that the use of elliptic cryptography requires a lesser load for the network and computing power, which is especially important for devices with limited storage and processing capabilities.

The objectives of the graduation work are the description of the mathematical theory of elliptic curves, the study of cryptographic algorithms on elliptic curves, and the implementation of cryptographic algorithms on elliptic curves.

The aim of the work is to implement the cryptographic algorithms on the elliptic curves.

The object of the graduation work is the elliptic curves.

The subject of the graduation work is the cryptographic algorithms on the elliptic curves.

The first chapter describes the mathematical theory of elliptic curves used in cryptographic algorithms.

The second chapter is a study of cryptographic algorithms on elliptic curves that implement key exchange, digital signature and encryption.

The third chapter shows the software implementation of elliptic curves in C++, as well as the implementation of the ECDH cryptographic algorithm.

Оглавление

Введение

Криптография на эллиптических кривых представляет собой мощный подход к криптографии и альтернативу хорошо известному алгоритму RSA. Этот подход для шифрования с открытым ключом использует математику эллиптических кривых для обеспечения безопасности между парами ключей. В последние несколько лет эллиптическая криптография медленно набирала популярность благодаря своей способности обеспечивать тот же уровень безопасности, что и RSA, но с гораздо меньшим размером ключа.

Ресурсы, доступные для взлома зашифрованных ключей, продолжают расширяться, а это означает, что размер зашифрованных ключей должен продолжать расти, чтобы обеспечивать безопасность. Это может оказаться бременем для некоторых устройств, особенно мобильных, которые не обладают такой большой доступной вычислительной мощностью. Однако криптография на эллиптических кривых помогает решить эту проблему.

Актуальность бакалаврской работы объясняется тем, что эллиптическая криптография успешно используется в течение последних нескольких лет и зарекомендовала себя как надежный и эффективный метод шифрования. Этот метод обладает необходимыми функциями для цифрового мира с его обширным обменом объемами данных и может гарантировать более высокий уровень безопасности, лучшую производительность и более короткие ключи для увеличения скорости.

Целью бакалаврской работы является реализация криптографических алгоритов на основе эллиптических кривых.

Задачи бакалаврской работы:

- описать теоретические основы эллиптических кривых, использующиеся в криптографии;
- исследовать алгоритмы криптографии на эллиптических кривых;
- реализовать алгоритм обмена ключами ECDH.

Объектом исследования бакалаврской работы являются эллиптические кривые.

Предметом исследования бакалаврской работы являются криптографические алгоритмы на эллиптических кривых.

Бакалаврская работа состоит из трех глав.

В первой главе описывается теория эллиптических кривых.

Во второй главе рассмотрены и проанализированы популярные алгоритмы эллиптической криптографии: алгоритм Диффи-Хеллмана для обмена ключами, алгоритм создания цифровой подписи ECDSA и схема ECIES.

В третьей главе реализована арифметика эллиптических кривых и алгоритм ECDH.

Бакалаврская работа содержит 29 рисунков и 11 формул.

Глава 1 Теоретические основы эллиптических кривых

1.1 Определение эллиптической кривой и эллиптической кривой как группы

Эллиптическая кривая – это алгебраическая кривая, представленная в формуле (1).

$$y^2 = x^3 + ax + b, \quad (1)$$

где a и b – действительные числа.

Этот тип уравнений называется эллиптической функцией Вейерштрасса и является популярным выбором для эллиптических кривых [2, 7, 15].

Пример эллиптической кривой изображен на рисунке 1.

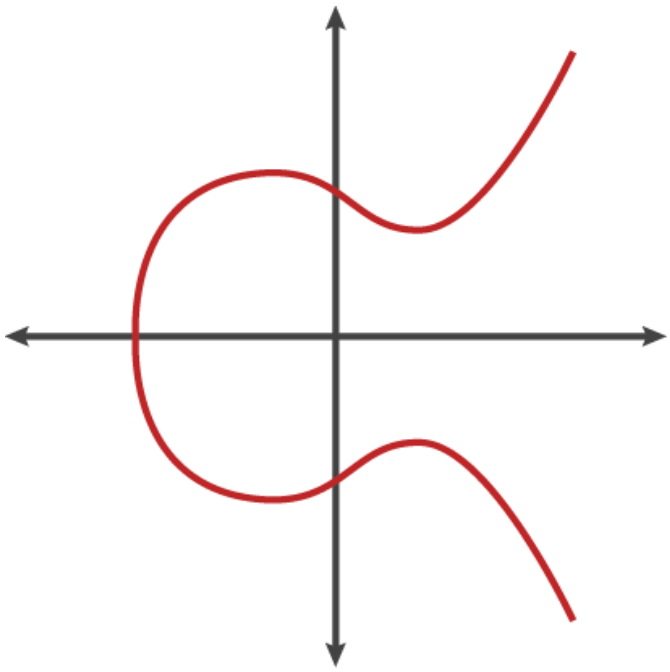

Рисунок 1 – эллиптическая кривая

Для того чтобы кривая была эллиптической по своей природе, существует важное свойство: она должна быть несингулярной [4, 10]. Чтобы

выполнить это условие, дискриминант вышеприведенного уравнения должен быть не равен нулю. Расчет дискриминанта представлен в формуле (2).

$$\Delta = -16(4a^3 + 27b^3) \qquad (2)$$
$$\Delta \neq 0$$

Сингулярные кривые изображены на рисунке 2.

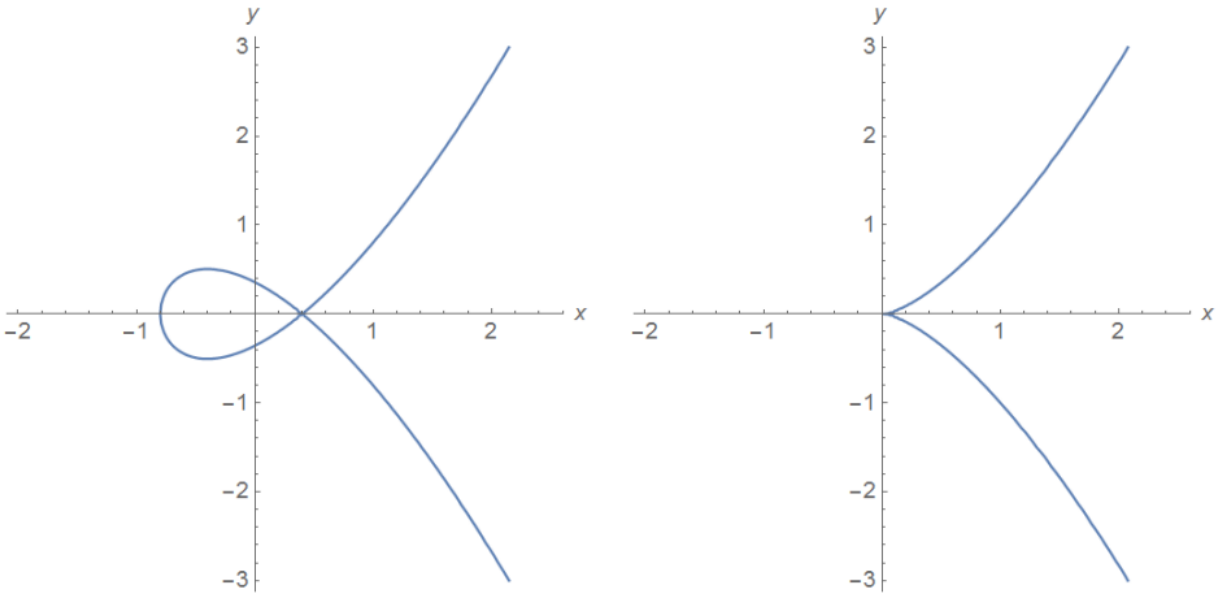

Рисунок 2 – сингулярные кривые

Для случая дискриминанта меньше нуля несингулярная кривая имеет одну компоненту или одну непрерывную кривую, например кривая $y^2 = x^3 - x + 1,\ \Delta < 0$ изображена на рисунке 3.

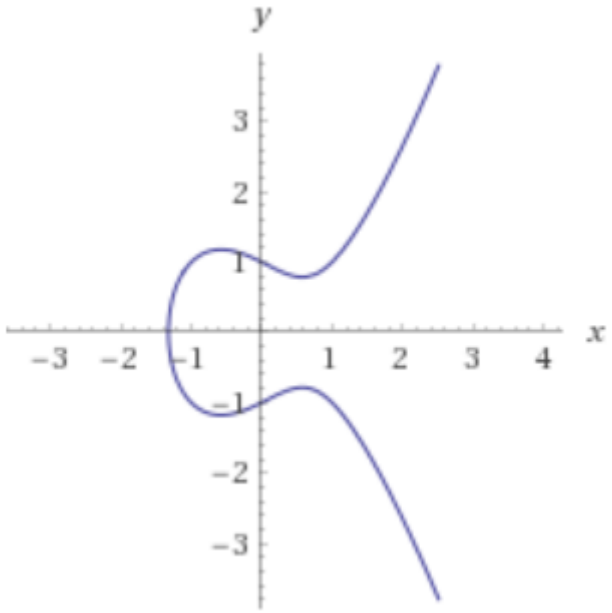

Рисунок 3 – несингулярная кривая с одной компонентой

Или две компоненты для $\Delta > 0$ $y^2 = x^3 - x$, кривая изображена на рисунке 4.

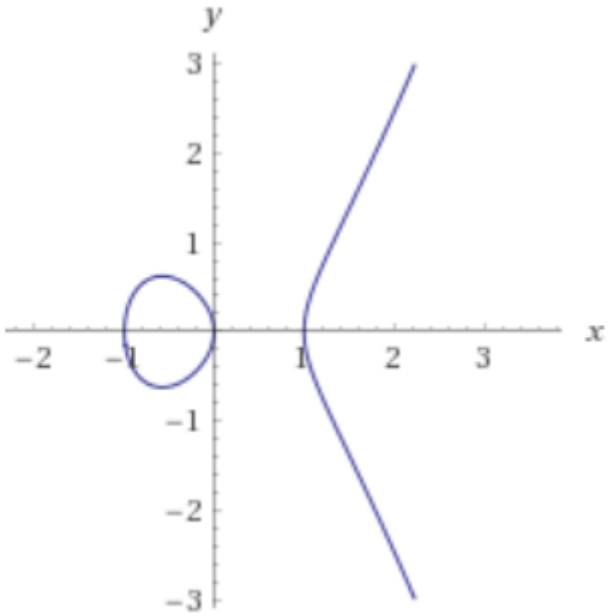

Рисунок 4 – несингулярная кривая с двумя компонентами

Можно определить группу на любой гладкой кубической кривой [20]. Группа – это множество с допустимой бинарной операцией (между двумя

элементами множества) для формирования третьего элемента, при котором выполняются четыре условия (аксиомы группы). Вот эти четыре аксиомы:

– в группу входят точки эллиптической кривой и точка в бесконечности;

– сложение представляется следующим образом: точки P, Q и R лежат на одной прямой и $P + Q + R = O$;

– существование нейтрального элемента O – точки в бесконечности;

– существование обратного элемента: для точки $P(x_1, y_1)$ существует точка $-P(x_1, -y_2)$.

Пример сложение двух точек изображен на рисунке 5.

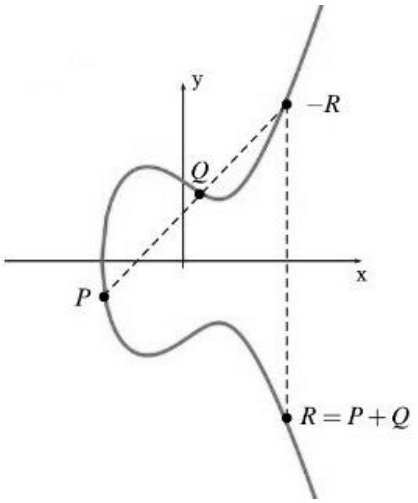

Рисунок 5 – пример сложения точек

Заметим, что при использовании второго правила нам требуется только три точки на одной прямой, причем три точки складываются без учета порядка. Это означает, что оператор $+$ является ассоциативным и коммутативным: группа является абелевой.

1.2 Сложение точек элиптической кривой

Рассмотрим возможные случаи сложения точек.

Первый случай. $P = (x_1, y_1)$ и $Q = (x_2, y_2)$, где $x_1 \neq x_2$ и $y_1 \neq y_2$. Нужно провести через них прямую и найти точку пересечения с кривой. Уравнение линии имеет вид, представленный в формуле (3).

$$y = kx + m = kx + (y_1 - kx_1) = k(x - x_1) + y_1, \quad (3)$$

где $k = \frac{y_2 - y_1}{x_2 - x_1}$.

Запишем формулу (4).

$$y^2 = (k(x - x_1) + y_1)^2 = x^3 + Ax + B \quad (4)$$

И переставим члены, чтобы получить формулу (5).

$$x^3 - k^2x^2 + \cdots = 0 \quad (5)$$

Мы можем записать любой кубический многочлен в виде формулы (6).

$$x^3 + ax^2 + bx + x = (x - x_1)(x - x_2)(x - x_3) = x^3 - (x_1 + x_2 + x_3)x^2 + \cdots = 0. \quad (6)$$

Таким образом, имея два корня x_1, x_2, третий корень можно получить по формуле (7).

$$k^2 = x_1 + x_2 + x_3 \rightarrow x_3 = k^2 - x_1 - x_2 \rightarrow y_3 = k(x_3 - x_1) + y_1 \rightarrow \quad (7)$$
$$\rightarrow -y_3 = k(x_1 - x_3) - y_1.$$

Таким образом, $P + Q = -R = (x_3, -y_3)$.

Второй случай. Обе точки имеют одинаковую координату x, но разные координаты y: $P = (x, y_1)$ и $Q = (x, y_2)$. Линия, проведенная через них, будет

вертикальной и пересечет кривую в точке O, поэтому P + Q = O. Пример этого случая изображен на рисунке 6.

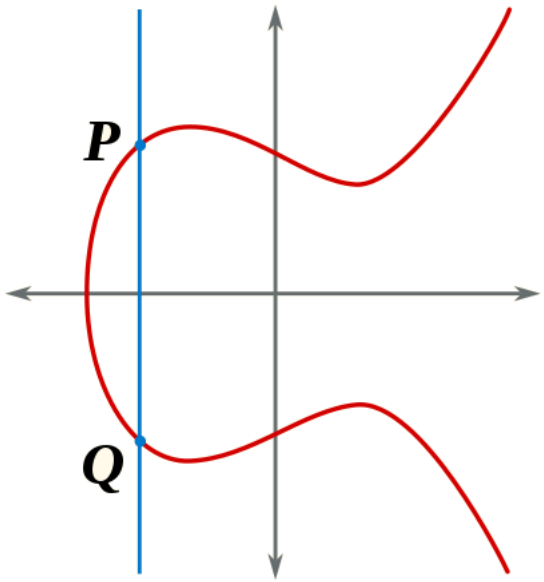

Рисунок 6 – сложение точек одинаковой координатой x

Третий случай. Также может случиться, что мы хотим добавить точку $P = (x_1, y_1)$ к самой себе. Можно представить что мы прибавляем точки P и Q, где Q приближается к P, так что расстояние между ними пренебрежимо мало. Тогда линия через через них также изменяется по мере приближения Q к P и становится прямой касательной к кривой в точке P. Это упрощает нахождение уравнения линии, так как мы можем использовать дифференцирование, чтобы получить угол налона прямой [14]. Вспомним уравнение $y^2 = x^3 + Ax + B$, тогда запишем формулу (8).

$$\frac{d}{dx}(y^2) = \frac{d}{dx}(x^3 + Ax + B) \leftrightarrow 2yy' = 3x^2 + A \rightarrow y' = \frac{3x_1^2 + A}{2y_1}. \quad (8)$$

Следовательно получаем формулу (9).

$$\frac{dy}{dx} = y' = k \rightarrow y = k(x - x_1) + y_1. \quad (9)$$

Новая точка находится как в случае 1.

Случай изображен на рисунке 7.

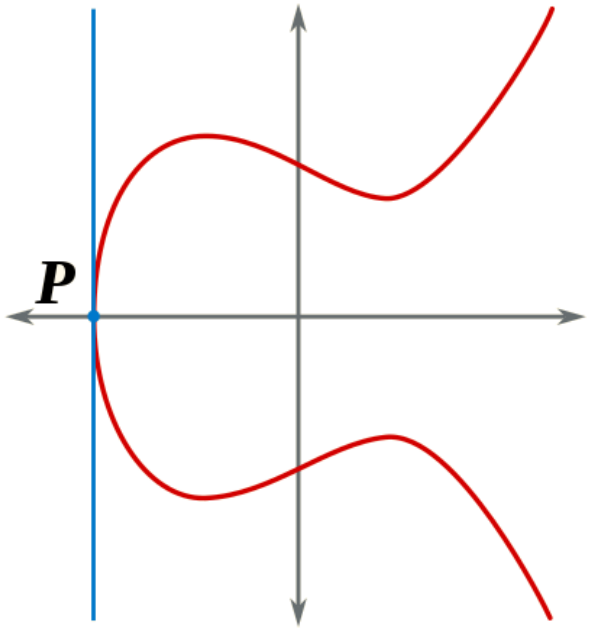

Рисунок 7 – сложение касательной точки

Четвертый случай. Наконец, может возникнуть ситуация, когда мы хотим сложить любую точку P с O. Интуитивно понятно, что любая прямая через P и O будет вертикальной прямой, она пересечет точку $-P$, отражение P. Отражение этой точки возвращает нас в P. Отсюда следует, что $P + O = P$.

1.3 Умножение точек эллиптической кривой

Скалярное умножение на эллиптической кривой определяется как повторное сложение. $P = nQ, n \epsilon \mathbb{Z}$ эквивалентно записи $P = Q + Q + \cdots + Q$, где сложение происходит n раз.

В любой реализации операций алгоритмов эллиптической криптографии скалярное умножение является вычислительно самой тяжелой

операцией.При больших n скалярное умножение будет требовать значительное количество времени. Для уменьшения количества действий следует реализовать более оптимальный алгоритм быстрого умножения, например, алгоритм “удвой и добавь”.

Алгоритм:

- $nP = 2(\frac{n}{2}P)$ если n четное;
- $nP = P + 2(\frac{n-1}{2}P)$ если n нечетное.

Алгоритм представлен на рисунке 8.

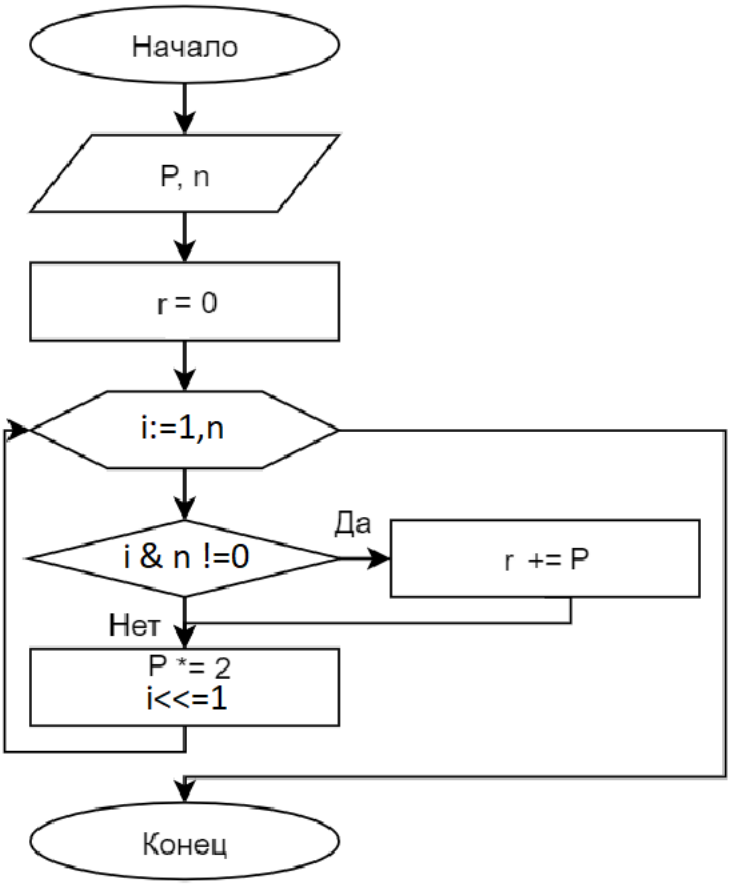

Рисунок 8 – алгоритм быстрого скалярного умножения

Алгоритм будет работать за $O(logn)$ действий.

1.4 Задача дискретного логарифмирования

Эта проблема, называемая дискретным логарифмом, является двигателем всей криптографии с открытым ключом. Именно благодаря ей все работает так гладко. Для целых чисел она определяется следующим образом:

Пусть $g, b -$ целые числа $(mod\ p)$ для некоторого простого p, где g, b не кратны p. Предположим, что существует целое число k такое, что выполняется формула (10).

$$g^k \equiv b(mod\ p). \tag{10}$$

Решение задачи дискретного логарифма сводится к нахождению такого k. Очень легко вычислить b из g и k, но но трудно найти k когда целые числа очень велики [3, 5, 9].

В качестве примера, пусть g – генератор мультипликативной группы $\mathbb{Z}_p^*$ для некоторого простого числа. Тогда группа представляется в формуле (11).

$$G = \{g^0, g^1, g^2, \dots, g^{p-2}\} = \mathbb{Z}_p^*. \tag{11}$$

Пусть $g = 5$ и $p = 277$. Тогда циклическая группа имеет вид $\{1, 5, 25, 125, 71, \dots, 117, \dots\}$. Конечно, чтобы найти, в какую экспоненту мы возвели 5, чтобы получить $117\ (mod\ p)$, нам нужно перебрать группу и найти этот элемент и его позицию. Для большого p это становится невыполнимым. Стоит заметить, что трудность задачи заключается в том, что не существует полиномиального времени алгоритма для ее решения.

Поскольку мы используем мультипликативные группы, можно спросить, почему бы не использовать группу эллиптических кривых. Действительно, это вполне возможно. ECDLP является модификацией DLP и работает следующим образом: целые числа $g, b \epsilon \mathbb{Z}_p$ меняются на элементы группы эллиптической кривой, которые являются точками на кривой. Таким образом, мы выбираем две точки $P, Q \in E(\mathbb{Z}_p)$ и пытаемся найти целое число k такое, что $kP = Q$ [6, 8].

Как видно, суть проблемы не изменилась. Для того чтобы решить ее, необходимо нужно найти k, удовлетворяющее приведенному выше уравнению. Уравнение представляет собой эллиптическую криптографию, и приложения в основном основаны на нем.

Выводы по первой главе. Были рассмотрены основные определения эллиптических кривых, связь с теорией групп, групповой закон для эллиптических кривых. Дано определение сложения точек на эллиптической кривой, рассмотрены способы сложения любых двух точек. Определено умножение точки на целое число, приведен алгоритм быстрого скалярного умножения. Рассмотрена задача дискретного логарифмирования и её роль в криптографии.

Глава 2 Алгоритмы эллиптической криптографии

2.1 Протокол ECDH

Чтобы понять ECDH, сначала нужно погрузиться в стандартный протокол обмена ключами Диффи-Хеллмана. Это один из первых протоколов с открытым ключом, разработанный еще в 1976 году, и он до сих пор широко используется.

Традиционно, когда реализовано симметричное шифрование, вам нужно будет обмениваться секретом по безопасному каналу – между двумя сторонами. Неотъемлемый недостаток этой методологии заключается в том, что третья сторона может перехватить секрет, если она имеет доступ к каналу связи. Затем этот секрет можно использовать для расшифровки зашифрованных данных между двумя сторонами, что делает его бесполезным.

В криптографии ключ – это строка символов, используемая в алгоритме шифрования для изменения данных, чтобы они выглядели случайными. В DH рекомендуется, чтобы размер ключа был выше 2000 бит, тогда как тот же уровень безопасности может быть достигнут в ECDH с 250 битами [1, 11].

DH позволяет двум сторонам, упомянутым выше, обмениваться своим секретом без необходимости использования безопасного канала для передачи секрета. Секрет, сгенерированный обменом, затем можно использовать для шифрования последующих сообщений с использованием симметричного шифрования с сгенерированным секретом.

Далее представлен алгоритм:

– A и B генерируют простое число p и число g, взаимно простое с $p - 1$;

– A выбирает секрет a. Затем вычисляет $kA = (g^a \, mod \, p)$ и отправляет его B;

– B делает то же самое: выбирает секрет b, вычисляет $kB = (g^b \ mod \ p)$ и отправляет его A;

– A вычисляет $(kB^a \ mod \ p)$;

– B вычисляет $(kA^b \ mod \ p)$.

Полученные числа в двух последних шагах одинаковые.

Итак, теперь, когда мы знаем, как работает обычный обмен ключами Диффи-Хеллмана, мы можем перейти к эллиптической кривой Диффи-Хеллмана. Концепция более-менее та же. Происходит тот же процесс, но также он использует алгебраические кривые для генерации ключей, которые будут использоваться сторонами. Кроме того, обе стороны должны заранее согласовать эллиптическую кривую. Использование эллиптических кривых также намного быстрее, чем использование больших чисел, необходимых в обычном DH [17]. Проблему дискретного логарифмирования эллиптической кривой труднее решить, чем обычную задачу дискретного логарифма. Это означает, что мы можем обойтись меньшими ключами, чем с DH.

Введём следующие обозначения:

E – сама эллиптическая кривая

G – точка на E, которая устанавливается в качестве базовой точки

Алгоритм генерации ключей:

– A генерирует случайное целое число a в качестве своего приватного ключа;

– A генерирует свой публичный ключ kA , вычисляя aG;

– B генерирует случайное целое число b в качестве своего приватного ключа;

– B генерирует свой публичный ключ kB , вычисляя bG;

– A и B обмениваются открытыми ключами;

– A вычисляет K как $a \cdot kB$;

– B вычисляет K как $b \cdot kA$.

Графически алгоритм представлен на рисунке 9.

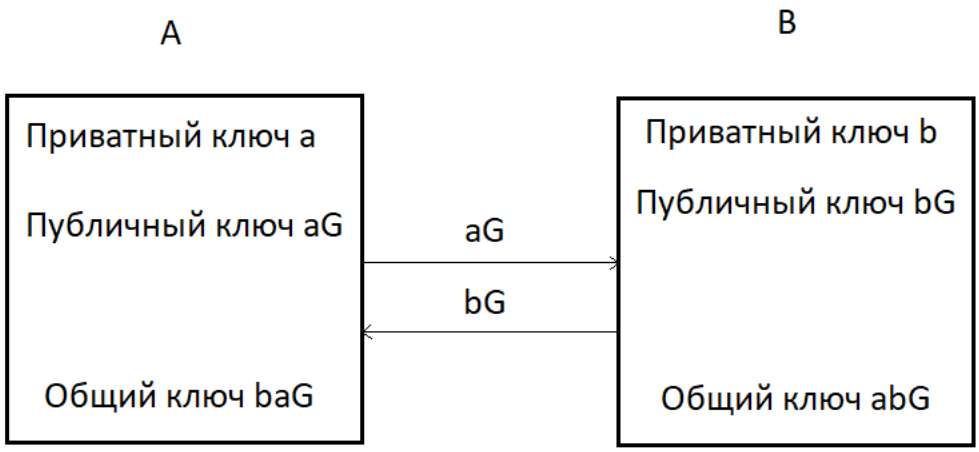

Рисунок 9 – алгоритм ECDH

Такой обмен ключами с эллиптической кривой трудно взломать, поскольку для этого необходимо решить задачу дискретного логарифма.

Как видно, разница между ECDH и DH не слишком велика и не слишком сложна в реализации. Следует помнить, что для больших объемов вычислений (например, веб-сайт с большим трафиком) время, сэкономленное на использовании эллиптических кривых вместо больших простых чисел, суммируется. На маленьком веб-сайте можно не увидеть разницы, но чем больше трафика, тем заметнее рост производительности [16].

2.2 Алгоритм цифровой подписи ECDSA

В криптографии на эллиптических кривых существует множество групп алгоритмов для цифровой подписи, шифрования и согласования ключей. Наиболее известной схемой подписи, использующей эллиптические кривые, является алгоритм цифровой подписи на эллиптических кривых (ECDSA). Это система шифрования, альтернативная классической системе алгоритма цифровой подписи (DSA) с использованием эллиптических кривых. Впервые она была предложена Скоттом Ванстоуном.

ECDSA все чаще используется для шифрования программного обеспечения и генерации ключей Bitcoin, blockchain и подписания криптовалютных транзакций. Благодаря более короткой длине ключа и высокой эффективности алгоритм стал довольно популярным.

ECDSA опирается на математику циклических групп эллиптических кривых над конечными полями и на сложность проблемы дискретного логарифма эллиптических кривых. Ключи и подписи ECDSA короче, чем в RSA, при том же уровне безопасности. 256-битная подпись ECDSA имеет такую же степень защиты, как 3072-битная подпись RSA [12].

Подпись ECDSA

Алгоритм подписи ECDSA принимает на вход сообщение m и приватный ключ $pKey$ и выдает на выходе подпись, состоящую из пары целых чисел $\{r, s\}$ [19]. Алгоритм подписи ECDSA основан на схеме подписи Эль Гамаля и работает следующим образом:

– Вычислить хэш сообщения, используя одну из хэш-функций, например SHA-256: $h = hash(m)$;

– Сгенерировать случайное число k в диапазоне $[1 \dots n - 1]$;

– Вычислить точку $R = k \cdot G$ и взять её координату по оси X $r = R.x$;

– Вычислить доказательство подписи: $s = k^{-1} \cdot (h + r \cdot pKey)(mod\ n)$;

– Вернуть подпись $\{r, s\}$.

Вычисленная подпись $\{r, s\}$ представляет собой пару целых чисел, каждое из которых находится в диапазоне $[1 \dots n - 1]$. Она кодирует точку $R = k \cdot G$, а также доказательство s, подтверждающее, что подписавшему известно сообщение h и приватный ключ $pKey$. Доказательство s по идее может быть проверено с помощью соответствующего публичного ключа $publicKey$.

Подписи ECDSA в 2 раза длиннее, чем приватный ключ подписывающего для кривой, используемой в процессе подписи. Например,

для 256-битных эллиптических кривых (например, secp256k1) подпись ECDSA составляет 512 бит (64 байта), а для 521-битных кривых (например, secp521r1) - 1042 бита.

Проверка подписи ECDSA.

Алгоритм проверки подписи ECDSA принимает на вход подписанное сообщение m, а также подпись $\{r, s\}$, полученную алгоритмом подписи и публичным ключом $publicKey$, соответствующим закрытому ключу подписывающего. На выходе получается булево значение: действительная или недействительная подпись. Алгоритм проверки подписи ECDSA работает следующим образом:

– Вычислить хэш сообщения с помощью хэш-функции, использованной для подписи: $h = hash(m)$;

– Вычислить по модулю обратную величину доказательства подписи: $s' = s^{-1} \ (mod\ n)$;

– Восстановить случайную точку, использованную во время подписания: $R' = (h \cdot s') \cdot G + (r \cdot s') \cdot publicKey$;

– Взять из точки R' ее координату по оси X: $r' = R'.x$;

– Вычислить результат проверки подписи, сравнивая, $r' == r$.

Общая идея проверки подписи заключается в том, чтобы восстановить точку R' с помощью публичного ключа и проверить, является ли она той же самой точкой R, сгенерированной случайным образом в процессе подписания.

Алгоритм представлен на рисунке 10.

Рисунок 10 – алгоритм ECDSA

По сравнению с RSA, ECDSA оказался более защищенным от современных методов взлома благодаря своей сложности. ECDSA обеспечивает тот же уровень безопасности, что и RSA, но при этом использует гораздо меньшую длину ключа. Поэтому при использовании более длинных ключей ECDSA потребуется значительно больше времени для взлома с помощью атак перебора [18].

Еще одним большим преимуществом ECDSA перед RSA является производительность и масштабируемость. Поскольку эллиптическая криптография обеспечивает оптимальную безопасность при меньшей длине ключа, требуется меньшая нагрузка на сеть и вычислительные мощности. Это отлично подходит для устройств с ограниченными возможностями хранения и обработки данных [13]. В сертификатах SSL/TLS алгоритмы на эллиптической криптографии сокращают время, необходимое для выполнения рукопожатий SSL/TLS, и может помочь быстрее загрузить сайт.

2.3 Схема ECIES

Интегрированная схема шифрования на эллиптических кривых (ECIES) – это стандарт шифрования, использующий открытый ключ для шифрования и закрытый ключ для расшифровки. Открытый ключ и закрытый ключ основаны на криптографии на эллиптических кривых.

ECIES предоставляет возможности для шифрования, обмена ключами и цифровой подписи одновременно. Схема называется интегрированной схемой шифрования, поскольку это гибридная схема, использующая систему открытых ключей для передачи сеансового ключа для дальнейшего использования симметричным шифром. В ECIES общий секрет Диффи-Хеллмана используется для получения двух симметричных ключей $K1$ и $K2$. Ключ $K1$ используется для шифрования открытого текста с использованием шифра с симметричным ключом, в то время как ключ $K2$ используется для аутентификации результирующего зашифрованного текста.

ECIES использует следующие криптографические алгоритмы:

– KA – функция получения общего секрета;

– KDF – функция получения общих ключей, которая строится на основе хэш-функции;

– ENC – это функция шифрования с симметричным ключом. DEC – функция дешифрования;

– MAC – это алгоритм кода аутентификации сообщения.

Графически алгоритм шифрования изображен на рисунке 11.

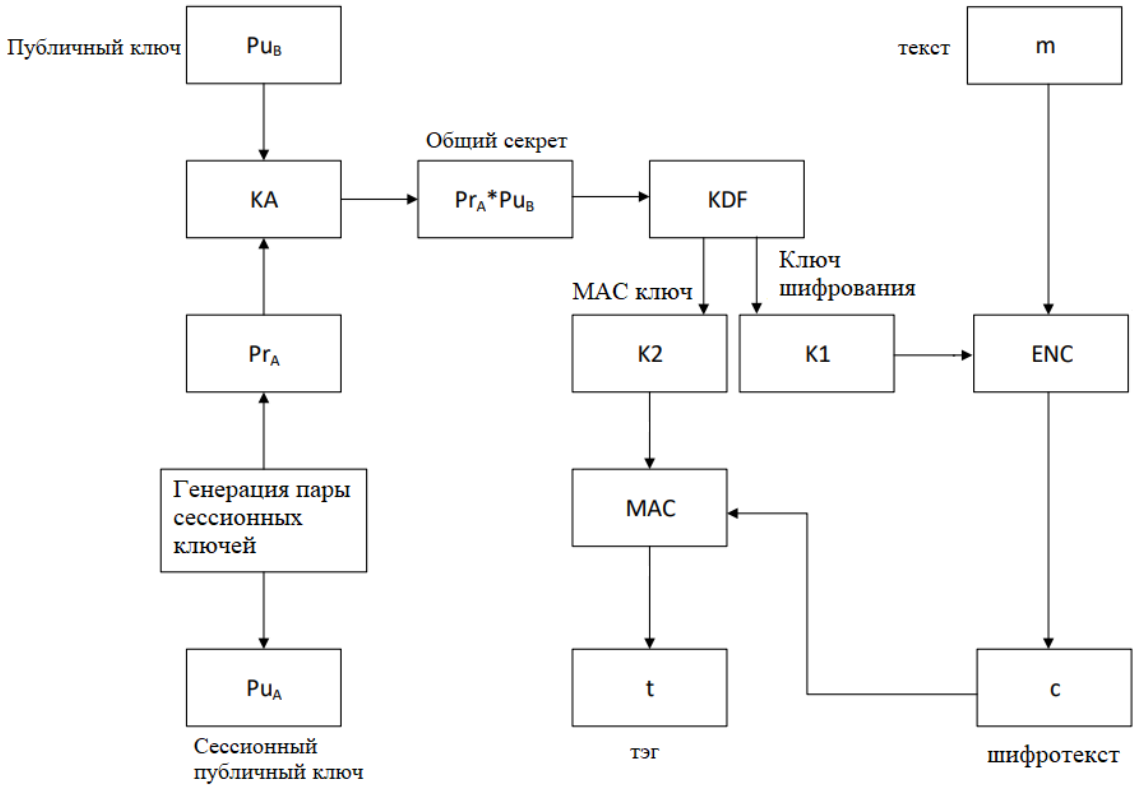

Рисунок 11 – шифрование ECIES

Алгоритм шифрования:

– А должен создать пару сеансовых ключей в виде случайного секретного значения: Pr_A в качестве закрытого ключа сеанса и Pu_A в качестве связанного открытого ключа сеанса;

– А использует функцию обмена ключами KA для создания общего секретного значения на основе закрытого ключа сессии А Pr_A и открытого ключа В Pu_B;

– А должнен взять общее секрет в качестве входных данных для функции KDF. Выходом этой функции является ключ симметричного шифрования $K1$, и MAC ключ $K2$;

– С $K1$ и открытым текстом m А использует алгоритм симметричного шифрования ENC для создания зашифрованного текста c;

– С зашифрованным текстом c А должен использовать выбранную функцию MAC с MAC ключом $K2$ для создания тега;

– Открытый ключ сеанса Pu_A, тег t и зашифрованный текст c отправляются получателю.

Графически алгоритм дешифрования представлен на рисунке 12.

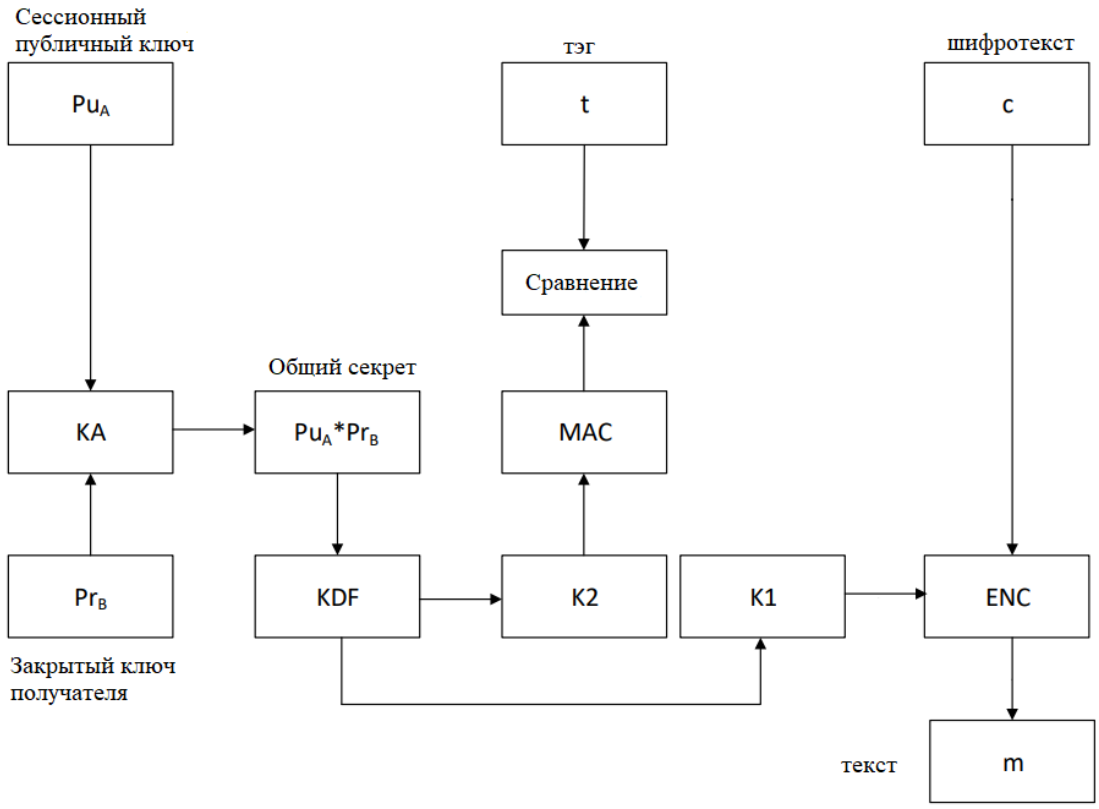

Рисунок 12 – дешифрование ECIES

Алгоритм дешифрования:

– После получения криптограммы $\{Pu_A, t, c\}$ от A, B должен получить открытый ключ сессии Pu_A, тэг t, и шифрованный текст c. B может работать с этими элементами по отдельности;

– Используя полученный открытый ключ сессии Pu_A и свой собственный закрытый ключ Pr_B, B создает общий секрет $Pr_B \cdot Pu_A$. Результат этого вычисления совпадает с расчитанным$Pr_A \cdot Pu_B$;

– Принимая общий секрет в качестве входных данных для KDF, B получает те же самые ключи $K1$ и $K2$ с помощью KDF;

– С помощью MAC-ключа $K2$ и зашифрованного сообщения c B вычисляет тэг и сравнивает его с тэгом, который он получил. Если значения отличаются, B должен отклонить криптограмму из-за сбоя в проверке MAC;

– Если значение тэга, сгенерированного B, верно, он может продолжить процесс, расшифровав текст шифра c с помощью симметричного алгоритма ENC и $K1$. В конце процесса расшифровки B может получить доступ к открытому тексту, который A намеревался отправить ему.

Вывод по итогам второй главы. Рассмотрен алгоритм ECDH, способ получения общих ключей. Рассмотрен алгоритм ECDSA для электронной цифровой подписи. Алгоритм вычисления и проверки подписи. Рассмотрен гибридный алгоритм ECIES.

Глава 3 Реализация эллиптической криптографии

3.1 Реализация арифметики на эллиптических кривых

Эллиптическая криптография реализована на языке C++. Вся программа реализована в файле ECC.cpp.

Для реализации была выбрана кривая $y^2 = x^3 + 7$, соответственно коэффициенты $a = 0, b = 7$. Кривая представлена на рисунке 13.

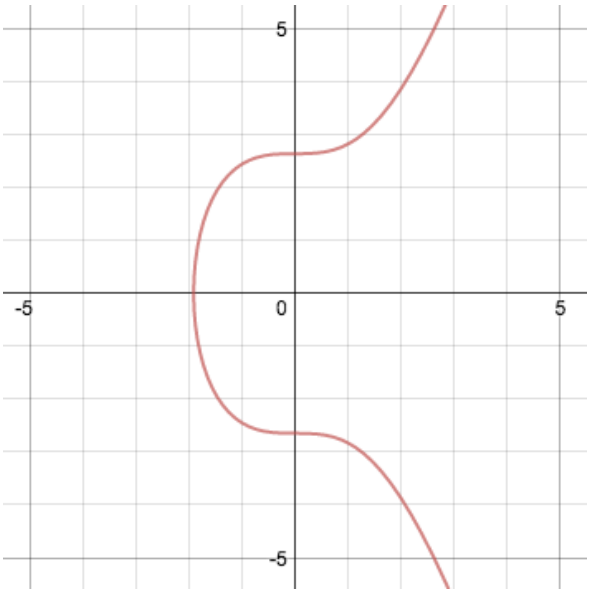

Рисунок 13 – кривая вида $y^2 = x^3 + 7$

Вся арифметика находится в классе ElPoint. Диаграмма класса представлена на рисунке 14.

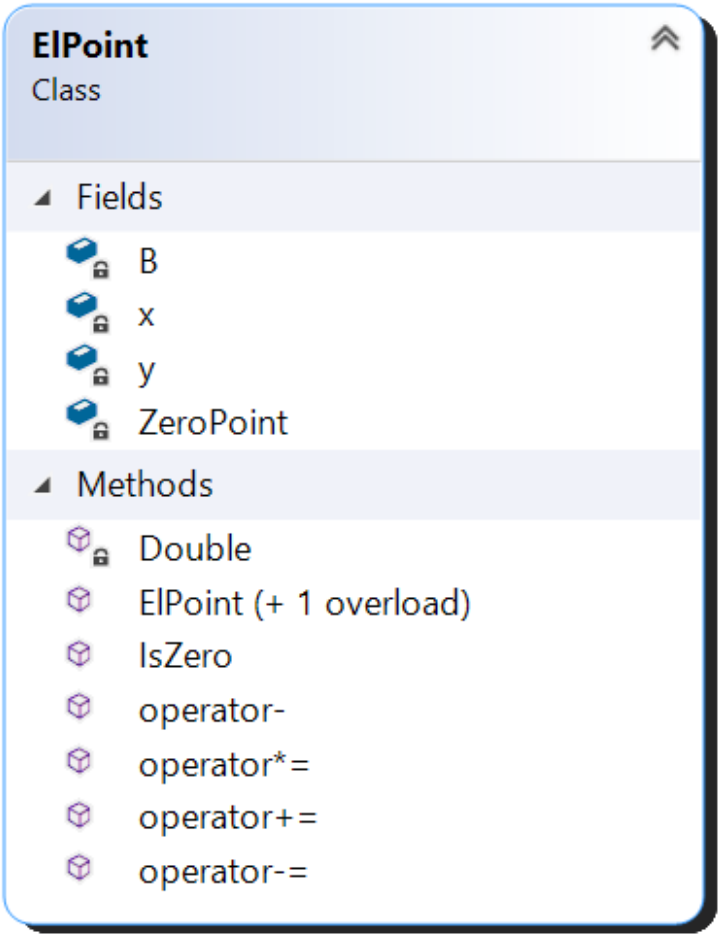

Рисунок 14 – диаграмма класса

В классе определены следующие поля:

– B – коэффициент b для кривой;

– x – координата точки по оси X;

– y – координата точки по оси Y;

– $ZeroPoint$ – точка в бесконечности.

Также определены следующие методы:

– $Double$ – удвоение точки;

– Конструкторы класса;

– Проверка на точку в бесконечности;

– Перегружен оператор отрицания точки;

– Перегружен оператор скалярного умножения;

– Перегружен оператор сложения точек ($P+= Q$);

– Перегружен оператор вычитания точек ($P-= Q$).

Конструктор класса для точки в бесконечности представлен на рисунке 15.

```
constexpr ElPoint() noexcept : x(0), y(ZeroPoint * 1.01) {}
```

Рисунок 15 – конструктор для точки в бесконечности

Конструктор класса, в который была передана координата y. Для рассматриваемой кривой существует только одна координата x. Конструктор представлен на рисунке 16.

```
explicit ElPoint(double yCoordinate) noexcept
{
    y = yCoordinate;
    x = cbrt(y * y - B);
}
```

Рисунок 16 – конструктор для обычной точки

Метод, проверяющий является ли точка точкой в бесконечности, представлен на рисунке 17.

```
bool IsZero() const noexcept
{
    bool isNotZero = abs(y) < ZeroPoint;
    return !isNotZero;
}
```

Рисунок 17 – проверка на точку в бесконечности

Реализация метода для удвоения точки представлена на рисунке 18.

```
void Double() noexcept
{
    if (IsZero())
    {
        return;
    }
    if (y == 0)
    {
        *this = ElPoint();
    }
    else
    {
        double L = (3 * x * x) / (2 * y);
        double newX = L * L - 2 * x;
        y = L * (x - newX) - y;
        x = newX;
    }
}
```

Рисунок 18 – метод удвоения точки

В основе оператора, обеспечивающего скалярное умножение точки, лежит оптимальный алгоритм ”удвой и добавь”. В реализации используется метод $Double$. Оператор представлен на рисунке 19.

```
ElPoint& operator*=(int rhs) noexcept
{
    ElPoint r;
    ElPoint p = *this;

    if (rhs < 0)
    {
        rhs = -rhs;
        p = -p;
    }

    for (int i = 1; i <= rhs; i <<= 1)
    {
        if (i & rhs) r += p;
        p.Double();
    }

    *this = r;
    return *this;
}
```

Рисунок 19 – оператор для скалярного умножения точек

Оператор для отрицания точки ($R = -R$) представлен на рисунке 20.

```
ElPoint operator-() const noexcept
{
    ElPoint negPt;
    negPt.x = x;
    negPt.y = -y;

    return negPt;
}
```

Рисунок 20 – оператор для отрицания точки

Оператор, реализующий сложение двух точек($P+= Q$), представлен на рисунке 21.

```
ElPoint& operator+=(const ElPoint& rhs) noexcept
{
    if (IsZero())
    {
        *this = rhs;
    }
    else if (rhs.IsZero())
    {
        // Если точка представляет собой точку в бесконечности
    }
    else
    {
        double L = (rhs.y - y) / (rhs.x - x);
        if (isfinite(L))
        {
            double newX = L * L - x - rhs.x;
            y = L * (x - newX) - y;
            x = newX;
        }
        else
        {
            if (signbit(y) != signbit(rhs.y))
            {
                // rhs == -lhs
                *this = ElPoint();
            }
            else
            {
                // rhs == lhs
                Double();
            }
        }
    }
    return *this;
}
```

Рисунок 21 – оператор для сложения двух точек

Оператор, реализующий вычитание двух точек ($P-= Q$),, представлен на рисунке 22.

```
ElPoint& operator-=(const ElPoint& rhs) noexcept
{
    *this += -rhs;
    return *this;
}
```

Рисунок 22 – оператор для вычитания двух точек

Также перегружен оператор сложения ($P + Q$), он представлен на рисунке 23.

```
ElPoint operator+(ElPoint lhs, const ElPoint& rhs) noexcept
{
    lhs += rhs;
    return lhs;
}
```

Рисунок 23 – оператор сложения

Перегружен оператор вычитания ($P - Q$), он представлен на рисунке 24.

```
ElPoint operator-(ElPoint lhs, const ElPoint& rhs) noexcept
{
    lhs += -rhs;
    return lhs;
}
```

Рисунок 24 – оператор вычитания

Перегружен оператор скалярного умножения для любого порядка следования операндов: $(n \cdot P)$ и $(P \cdot n)$. Операторы представлены на рисунке 25.

```
ElPoint operator*(ElPoint lhs, const int rhs) noexcept
{
    lhs *= rhs;
    return lhs;
}

ElPoint operator*(const int lhs, ElPoint rhs) noexcept
{
    rhs *= lhs;
    return rhs;
}
```

Рисунок 25 – операторы скалярного умножения

С помощью всех этих методов можно осуществить все необходимые действия над точками на эллиптической кривой.

3.2 Реализация ECDH

Для реализации алгоритма была написана функция ECDH.

На вход функции подается базисная точка P, личные ключи $keyA$ и $keyB$, а также по ссылке передаются переменные Ka и Kb, в которых будет записан общий секрет.

Реализация функции представлена на рисунке 26.

```
void ECDH(ElPoint p, int keyA, int keyB, ElPoint & Ka, ElPoint& Kb) {
    Ka = keyA * p;
    Kb = keyB * p;
    Ka = Ka * keyB;
    Kb = Kb * keyA;
}
```

Рисунок 26 – функция ECDH

В результате работы функции в переменных Ka и Kb будут храниться одинаковые значения. Полный код программы представлен в Приложении А.

3.3 Тестирование программы

Для тестирования программы был определен оператор вывода. Его реализация представлена на рисунке 27.

```
ostream& operator<<(ostream& os, const ElPoint& pt)
{
    if (pt.IsZero()) cout << "(Zero)\n";
    else cout << "(" << pt.x << ", " << pt.y << ")\n";
    return os;
}
```

Рисунок 27 – оператор вывода

В качестве входных данных была взята базовая точка с координатой $y = 100$, личный ключ для пользователя A: $a = 47$, личный ключ для пользователя B: $b = 78$. Реализация входных данных и вызов функции представлены на рисунке 28.

```
const ElPoint p(100);
cout << "p = " << p;
int keyA = 47;
int keyB = 78;
ElPoint Ka, Kb;
ECDH(p, keyA, keyB, Ka, Kb);
cout << "Ka= " << Ka << "Kb= " << Kb;
```

Рисунок 28 – входный данные и вызов функции

И далее результат работы программы представлен на рисунке 29.

```
p = (21.5393, 100)
Ka= (0.604282, -2.68713)
Kb= (0.604282, -2.68713)
```

Рисунок 29 – вывод программы

В результате работы программы значения Ka и Kb оказались равны, значит программа работает правильно.

Выводы по третьей главе. В третьей главе была реализована арифметика на эллиптических кривых: был создан класс, определяющий точку на эллиптической кривой, также реализованы все основные операции над точками. В классе перегружен конструктор для точки в бесконечности и констурктор для обычной точки, определен метод удвоения точки, метод проверки на точку в бесконечности, оператор быстрого скалярного умножения, оператор для нахождения отрицания точки, операторы для сложения точек, операторы для вычитания точек. Реализован алгоритм ECDH, он состоит из одной функции. Проведено тестирование разработанного алгоритма. Для этого предварительно был перегружен оператор вывода.

Заключение

В ходе выполнения выпускной квалификационной работы были рассмотрены теоретические основы эллиптических кривых, основные криптографические алгоритмы, программно реализована математика эллиптических кривых. Все поставленные задачи были решены.

Из теории эллиптических кривых были выделены значимые части, стоящие в основе криптографии на эллиптических кривых.

Исследованы следующие алгоритмы криптографии на эллиптических кривых: обмен ключами Диффи–Хеллмана, алгоритм создания цифровой подписи ECDSA, схема шифрования ECIES. Было замечено, что эллиптические кривые приобретают все большее значение в области криптографии и как они используются для цифровых подписей.

На основании проделанной работы можно выделить основные достоинства эллиптической криптографии: в криптографии основанной на эллиптических кривых длина ключа гораздо меньшая по сравнению с другими алгоритмами. Более высокая безопасность: системы на основе эллиптических кривых более надежны, чем RSA, и могут выдерживать квантовые вычисления. Мы получаем повышенную производительность, так как между сервером и клиентом проверяется меньше данных, что приводит к повышению производительности сети. Это особенно удобно для сайтов с высоким уровнем трафика.

Для реализации был выбран язык C++. Программа представляет собой файл ECC.cpp, в котором реализован класс ElPoint. Класс определяет точку на эллиптической кривой, также в классе определены все математические операции, необходимые для реализации алгоритмов эллиптической криптографии.

Используя вышеописанный класс, был реализован ранее рассмотренный алгоритм обмена ключами ECDH, состоящий из одной функции.

Список используемой литературы

1. Бабаш А.В. Информационная безопасность и защита информации: учебное пособие / А.В. Бабаш, П.Н. Башлы, Е.К. Баранова. – М.: РИОР, 2013. – 222 с.

2. Болотов А.А., Гашков С.Б., Фролов А.Б., Часовских А.А. Элементарное введение в эллиптическую криптографию. Алгебраические и алгоритмические основы.. – Москва: КомКнига, 2006. – 328 с.

3. Василенко О.Н. Теоретико-числовые алгоритмы в криптографии.- М.: МЦНМО, 2003.-328 с.

4. Васильева И.Н. Криптографические методы защиты информации : учебник и практикум для академического бакалавриата / И.Н. Васильева. – М. : Издательство Юрайт, 2016. – 349 с. – Серия : Бакалавр. Академический курс.

5. Глухов М.М., Круглов И.А., Пичкур А.Б., Черемушкин А.В. Введение в теоретико-числовые методы криптографии: Учебное пособие. – СПб.: Издательство «Лань», 2011. – 400 с. – (Учебники для вузов. Специальная литература).

6. Жданов О.Н., Золотарев, В.В. Методы и средства криптографической защиты информации: Учебное пособие / О.Н. Жданов, В. В. Золотарев; СибГАУ. – Красноярск, 2007. – 217 с.

7. Жданов О.Н., Чалкин В.А. Эллиптические кривые. Основы теории и криптографические приложения.-М.: Кн. дом «ЛИБРОКОМ», 2012.-200с.

8. Иванов М.А. Криптографические методы защиты и информации в компьютерных системах и сетях: учеб. пособие / М.А. Иванов, И.В. Чугунков. – М.: МИФИ, 2012 – 400 с.

9. Кияев В.И. Безопасность информационных систем: учеб. пособие / В.И. Кияев, Граничин О.Н. – М.: ИНТУИТ, 2016 – 192 с.

10. Рубцова Р.Г. Математические основы защиты информации: учеб. пособие / Р.Г. Рубцова, Ш.Т. Ишмухаметов. – Казань: Казанский федеральный университет, 2012 – 138 с.

11. Рябко Б.Я., Фионов А.Н. Криптографические методы защиты информации: учебное пособие. – М.: Горячая линия–Телеком, 2005. – 229 с.

12. ECDSA: Elliptic Curve Signatures [Электронный ресурс]. URL: https://cryptobook.nakov.com/digital-signatures/ecdsa-sign-verify-messages (дата обращения: 16.05.2022).

13. ECDSA vs RSA: Everything You Need to Know [Электронный ресурс]. URL: https://sectigostore.com/blog/ecdsa-vs-rsa-everything-you-need-to-know/ (дата обращения: 16.05.2022).

14. Harady Hasan Elliptic Curves A journey through theory and its applications [Электронный ресурс]. URL: https://uu.diva-ortal.org/smash/get/diva2:1334316/FULLTEXT01.pdf (дата обращения: 16.05.2022).

15. Mark Hughes How Elliptic Curve Cryptography Works [Электронный ресурс]. URL: https://www.allaboutcircuits.com/technical-articles/elliptic-curve-cryptography-in-embedded-systems/ (дата обращения: 16.05.2022).

16. Nelson Josias Gbètoho Saho, Eugène C. Ezin Comparative Study on the Performance of Elliptic Curve Cryptography Algorithms with Cryptography through RSA Algorithm [Электронный ресурс]. URL: https://hal.archives-ouvertes.fr/hal-02926106/document (дата обращения: 16.05.2022).

17. Rakel Haakegaard and Joanna Lang: The Elliptic Curve Diffie-Hellman (ECDH) [Электронный ресурс]. URL: http://koclab.cs.ucsb.edu/teaching/ecc/project/2015Projects/Haakegaard+Lang.pdf (дата обращения: 16.05.2022).

18. Simone Catania Elliptic curve cryptography: your future-proof standard for encryption [Электронный ресурс]. URL:

https://www.internetx.com/en/news-detailview/elliptic-curve-cryptography-your-future-proof-standard-for-encryption/ (дата обращения: 16.05.2022).

19. The Elliptic Curve Digital Signature Algorithm (ECDSA) [Электронный ресурс]. URL: https://www.cs.miami.edu/home/burt/learning/Csc609.142/ecdsa-cert.pdf (дата обращения: 16.05.2022).

20. The Elliptic Curve Group Law [Электронный ресурс]. URL: https://www.math.brown.edu/reschwar/M1540B/elliptic.pdf (дата обращения: 16.05.2022).

Приложение А

Реализация алгоритма ECDH

```
#include <cmath>
#include <iostream>

using namespace std;


class ElPoint
{
    double x, y;
    static constexpr double ZeroPoint = 1e20;
    static constexpr double B = 7; //  'b' в y^2 = x^3 + a * x + b
                                   //  'a' = 0

    void Double() noexcept
    {
        if (IsZero())
        {
            return;
        }
        if (y == 0)
        {
            *this = ElPoint();
        }
        else
        {
            double L = (3 * x * x) / (2 * y);
            double newX = L * L - 2 * x;
            y = L * (x - newX) - y;
            x = newX;
        }
    }

public:
    friend std::ostream& operator<<(std::ostream&, const ElPoint&);

    constexpr ElPoint() noexcept : x(0), y(ZeroPoint * 1.01) {}

    explicit ElPoint(double yCoordinate) noexcept
    {
        y = yCoordinate;
        x = cbrt(y * y - B);
    }

    bool IsZero() const noexcept
    {
        bool isNotZero = abs(y) < ZeroPoint;
        return !isNotZero;
    }

    ElPoint operator-() const noexcept
    {
        ElPoint negPt;
        negPt.x = x;
        negPt.y = -y;

        return negPt;
    }
```

```
ElPoint& operator+=(const ElPoint& rhs) noexcept
{
    if (IsZero())
    {
        *this = rhs;
    }
    else if (rhs.IsZero())
    {
        // Если точка представляет собой точку в бесконечности
    }
    else
    {
        double L = (rhs.y - y) / (rhs.x - x);
        if (isfinite(L))
        {
            double newX = L * L - x - rhs.x;
            y = L * (x - newX) - y;
            x = newX;
        }
        else
        {
            if (signbit(y) != signbit(rhs.y))
            {
                // rhs == -lhs
                *this = ElPoint();
            }
            else
            {
                // rhs == lhs
                Double();
            }
        }
    }
    return *this;
}

ElPoint& operator-=(const ElPoint& rhs) noexcept
{
    *this += -rhs;
    return *this;
}

ElPoint& operator*=(int rhs) noexcept
{
    ElPoint r;
    ElPoint p = *this;

    if (rhs < 0)
    {
        rhs = -rhs;
        p = -p;
    }

    for (int i = 1; i <= rhs; i <<= 1)
    {
        if (i & rhs) r += p;
        p.Double();
    }

    *this = r;
    return *this; }
```

```
};

ElPoint operator+(ElPoint lhs, const ElPoint& rhs) noexcept
{
    lhs += rhs;
    return lhs;
}

ElPoint operator-(ElPoint lhs, const ElPoint& rhs) noexcept
{
    lhs += -rhs;
    return lhs;
}

ElPoint operator*(ElPoint lhs, const int rhs) noexcept
{
    lhs *= rhs;
    return lhs;
}

ElPoint operator*(const int lhs, ElPoint rhs) noexcept
{
    rhs *= lhs;
    return rhs;
}

void ECDH(ElPoint p, int keyA, int keyB, ElPoint & Ka, ElPoint& Kb) {
    Ka = keyA * p;
    Kb = keyB * p;
    Ka = Ka * keyB;
    Kb = Kb * keyA;
}

ostream& operator<<(ostream& os, const ElPoint& pt)
{
    if (pt.IsZero()) cout << "(Zero)\n";
    else cout << "(" << pt.x << ", " << pt.y << ")\n";
    return os;
}

int main(void) {
    const ElPoint p(100);
    cout << "p = " << p;
    int keyA = 47;
    int keyB = 78;
    ElPoint Ka, Kb;
    ECDH(p, keyA, keyB, Ka, Kb);
    cout << "Ka= " << Ka << "Kb= " << Kb;

    return 0;
       }
```