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)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки, специальности)

Мобильные и сетевые технологии
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(БАКАЛАВРСКАЯ РАБОТА)

на тему «Реализация и исследование криптографических алгоритмов на эллиптических кривых»

Студент С.С. Сигеева
(И.О. Фамилия) (личная подпись)

Руководитель к.ф.-м н., доцент, Г. А. Тырыгина
(ученая степень, звание, И.О. Фамилия)

Консультант М. В. Дайнеко
(ученая степень, звание, И.О. Фамилия)

Тольятти 2021

Аннотация

Бакалаврская работа посвящена реализации и исследованию криптографических алгоритмов на эллиптических кривых.

Актуальность темы заключается в широком распространении мобильных устройств и устройств с ограниченными ресурсами, которые подключены к беспроводным сетям. Для таких устройств криптография на эллиптических кривых является наиболее подходящим способом защиты информации.

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

Задачами бакалаврской работы являются описание математический теории эллиптических кривых, исследование алгоритмов криптографии на эллиптических кривых, и реализация криптографических алгоритмов на 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 по трем основным направлениям: обмен ключами, цифровая подпись и шифрование.

В третьей главе показана программная реализация эллиптических кривых на языке C++, а также реализация криптографических алгоритмов ECDH и ECDSA.

Бакалаврская работа состоит из введения, трех глав, включающих 28 изображений и 28 формул, заключения и списка используемой литературы, включающего 20 источников, все на иностранном языке.

Abstract

The present graduation work is devoted to implementation and research of cryptographic algorithms on elliptic curves.

The work consists of 28 figures, 28 formulas and the list of 20 references including foreign sources.

The goal of the investigation is to implement the cryptographic algorithms on the elliptic curves.

The object of the work is the elliptic curves.

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

The relevance of the research consists in the wide distribution of mobile and resource-constrained devices connected to wireless networks.

For such devices, elliptic curve cryptography is considered to be the most suitable and appropriate method of ensuring information security.

The first chapter of the work reveals the mathematical fundamentals of elliptic curve theory relevant to cryptography.

It is pointed out that elliptic curves form Abelian groups and contain cyclic subgroups when defined over prime finite fields.

We also prove that point addition is the main operation used in elliptic curve cryptography.

In the second chapter of the investigation, we explore the three main applications of elliptic curve cryptography by using the well-known algorithms as examples.

We elucidate that Elliptic-Curve Diffe-Hellman (ECDH) algorithm, Elliptic Curve Digital Signature Algorithm (ECDSA) and Elliptic Curve Integrated Encryption Scheme (ECIES) are the most common elliptic curve cryptography algorithms mainly used for key exchange, digital signing and encryption.

In the third chapter of the research, we develop C++ standard library by implementing the finite fields and the elliptic curves.

We finish the development by using the library to implement the previously studied Elliptic-Curve Diffe-Hellman and Elliptic Curve Digital Signature Algorithm protocols.

We come to the conclusion that the development of elliptic curve cryptography-based systems is not difficult, and the developed library can be used to implement elliptic curve cryptography algorithms.

The work is of interest for a wide circle of readers, as the implemented algorithms can be used to show various cryptographic processes.

Оглавление

Введение

В настоящий момент рынок мобильных устройств является одним из самых быстроразвивающихся на планете. У каждого человека есть если не смартфон, то обыкновенный мобильный телефон. Учитывая множество функций, которые выполняют в современном мире мобильные устройства, так или иначе они будут взаимодействовать с конфиденциальной информацией. Большинство этих устройств также имеют сетевые возможности, и там, где есть подключение к Интернету, есть злоумышленники, которые пытаются использовать слабые места в системе безопасности.

В дополнение к смартфонам и планшетам, Интернет вещей (IoT) также набирает популярность. Люди живут в "умных" домах, где лампочки, кондиционеры, гаражные двери и дверные звонки подключены к Интернету. Теперь, когда IoT размывает границы между цифровым и физическим миром, безопасность жизненно важна.

Актуальность бакалаврской работы объясняется тем, что криптография на эллиптических кривых является наиболее подходящим подходом к обеспечению информационной безопасности на устройствах с ограниченными вычислительными ресурсами [11]. К таким устройствам относятся как мобильные телефоны и смартфоны, так и маломощные домашние устройства IoT, работающие на беспроводных сетях. Криптография на эллиптических кривых достигает большей скорости выполнения при меньших размерах ключей, что позволяет экономить как память, так и процессорную мощность, и полосу пропускания [5]. Эти особенности делают ее идеальной для применения в современном мире, где мобильные устройства и беспроводная связь имеют большее распространение, чем когда-либо.

Целью бакалаврской работы является реализация криптографических алгоритмов 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 и алгоритм цифровой подписи ECDSA.

Объектом исследования бакалаврской работы являются эллиптические кривые.

Предметом исследования бакалаврской работы являются криптографические алгоритмы на эллиптических кривых.

Бакалаврская работа содержит три главы.

В первой главе описывается математическая теория эллиптических кривых, использующаяся в криптографических алгоритмах. Описываются как эллиптические кривые в действительных числах, так и на конечных полях.

Во второй главе проводится исследование криптографических алгоритмов на эллиптических кривых. Исследуются три направления: обмен ключами через алгоритм ECDH, цифровая подпись через алгоритм ECDSA и смешанные алгоритмы шифрования через алгоритм ECIES.

В третьей главе происходит разработка и программная имплементация алгоритмов криптографии на эллиптических кривых. Разрабатываются как математические основы эллиптических кривых, так и алгоритмы, на них построенные.

Бакалаврская работа содержит 28 рисунков и 28 формул.

Глава 1 Теоретические основы эллиптических кривых

1.1 Эллиптические кривые в действительных числах

1.1.1 Определение эллиптической кривой

Эллиптическая кривая – это вид кубической кривой, решения которой ограничены областью пространства, топологически эквивалентной тору. Алгебраически эллиптические кривые существуют над наборами чисел или полями. В качестве полей для эллиптических кривых зачастую используются действительные, комплексные и рациональные числа.

Для представления пространства тора эллиптической кривой в алгебраической форме используется функция Вейерштрасса. Функция Вейерштрасса для действительных чисел представлена на формуле (1).

$$y^2 = x^3 + ax + b. \qquad (1)$$

где a, b – действительные числа.

В зависимости от значений a и b эллиптические кривые могут принимать различные формы на плоскости. Кроме того, вне зависимости от формы, все эллиптические кривые симметричны относительно оси x. Примеры кривых, заданных функцией Вейерштрасса, представлены на рисунке 1.

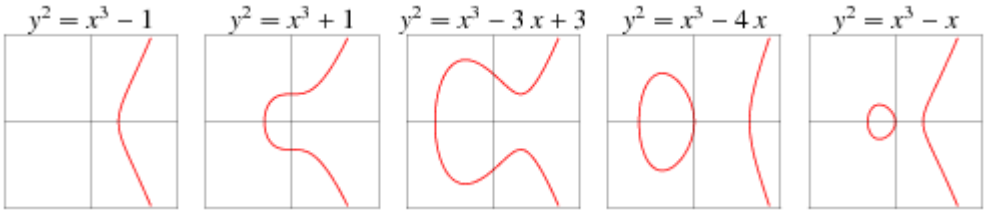

Рисунок 1 – Примеры эллиптических кривых

Эллиптические кривые, использующиеся в криптографии, обязательно являются несингулярными, то есть они не должны пересекаться сами с собой, не должны иметь точек возврата или изолированных точек. Достигается это за счет ограничения значений a и b. Кривая является несингулярной только в том случае, если дискриминант функции Вейерштрасса $4a^3 + 27b^2$ является равным нулю. Примеры сингулярных кривых представлены на рисунке 2.

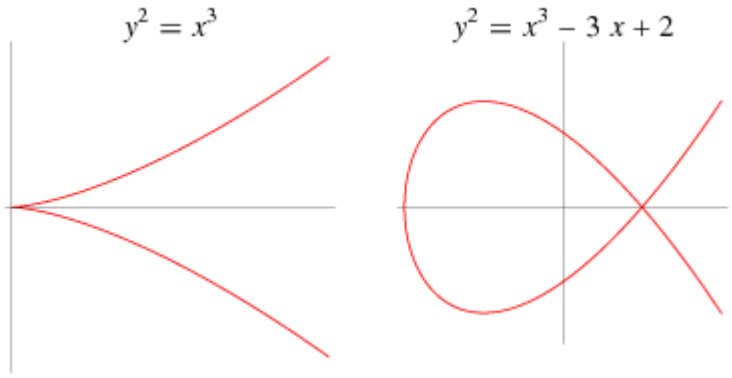

Рисунок 2 – Примеры сингулярных кривых

Принято считать, что эллиптические кривые находятся на проективной плоскости (то есть плоскости, не имеющей границ за счет добавления бесконечно удаленной прямой, на которой лежат все бесконечно удаленные точки) и соответственно имеют бесконечно удаленную точку, также называемую идеальной точкой. Идеальная точка обозначается как 0.

Общее определение эллиптической кривой, соответствующей направленности данной работы, представлено на формуле (2).

$$\{(x, y) \in R \vee y^2 = x^3 + ax + b, 4a^3 + 27b^2 \neq 0\} \cup \{0\}. \qquad (2)$$

1.1.2 Эллиптическая кривая как группа

Группой называется множество, для которого была определена операция, объединяющая любые два элемента в третий элемент. Чаще всего эта операция называется сложением и обозначается знаком «+».

Для того, чтобы некоторое множество M являлось группой, операция сложения должна быть определена таким образом, чтобы выполнялись следующие условия:

- замыкание: если a и b входят во множество M, то $a + b = c$ также входит во множество M;
- ассоциативность: $(a + b) + c = a + (b + c)$;
- существует нейтральный элемент 0 такой, что $a + 0 = 0 + a = a$;
- для каждого элемента a существует обратный элемент $-a$ такой, что $a + (-a) = 0$.

Если в группе также соблюдается свойство коммутативности ($a + b = b + a$), то группа называется абелевой группой.

Простейшим примером группы является множество целых чисел. Операция сложения, определенная для целых чисел, выполняет все необходимые свойства группы, а также свойство коммутативности, что делает множество целых чисел абелевой группой. С другой стороны, множество натуральных чисел не является группой, так как в нем не существует обратных элементов.

Из вышеописанных свойств операции сложения внутри группы следуют и другие свойства групп. Например, нейтральный элемент в группе может быть только один, и этот элемент уникален. Еще одно свойство групп состоит в том, что для каждого элемента обратный элемент также может быть только один, и этот обратный элемент уникален. Эти свойства широко используются в вычислениях внутри групп.

Для того, чтобы определить эллиптическую кривую как группу, отметим, что эллиптические кривые находятся на проективной плоскости.

Другими словами, бесконечно удаленные точки соединяются в бесконечно удаленную прямую таким образом, что любые две различные точки на плоскости всегда определяют прямую, и любые две прямые пересекаются в единственной точке. То есть прямые, параллельные на аффинной плоскости будут пересекаться в одной бесконечно удаленной точке на проективной плоскости.

Следовательно, законы группы для эллиптической кривой можно определить следующим образом [17]:

- за группу принимается множество точек на эллиптической кривой, а также одна бесконечно удаленная точка;
- за нейтральный элемент принимается бесконечно удаленная точка O;
- за обратный элемент точки P принимается точка $-P$, симметричная точке P относительно оси x;
- операция сложения определяется как $P + Q + R = O$, где P, Q и R – три точки на эллиптической кривой, принадлежащие к одной прямой.

Так как точки P, Q и R лежат на одной прямой, их порядок не имеет значения – они будут принадлежать к одной прямой в любом порядке. Из этого можно сделать вывод о том, что в определенной выше операции сложения порядок и группировка слагаемых не имеет значения, то есть выполняются свойства как ассоциативности, так и коммутативности. Это значит, что группа точек на эллиптической кривой является абелевой [10].

1.1.3 Сложение точек на эллиптической кривой

Ранее операция сложения для группы эллиптической кривой была определена как $P + Q + R = O$. Возникает вопрос о том, как происходит сложение двух точек, если операция определена для двух слагаемых. Чтобы ответить на этот вопрос, следует вспомнить, что нейтральный элемент O аналогичен нулю и обладает аналогичными арифметическими свойствами. Следовательно, одно из слагаемых можно перенести в левую часть

выражения, изменив знак: $P + Q = -R + O$, а из свойств нейтрального элемента известно, что $R + O = R$, то есть $P + Q = -R$.

Можно видеть, что для нахождения суммы двух точек необходимо найти точку и взять обратный для нее элемент. Если P и Q – точки на эллиптической кривой, то R – точка пересечения эллиптической кривой и прямой, проведенной через точки P и Q. Как уже было описано, точка, обратная точке R симметрична ей относительно оси x. Процесс представлен на рисунке 3.

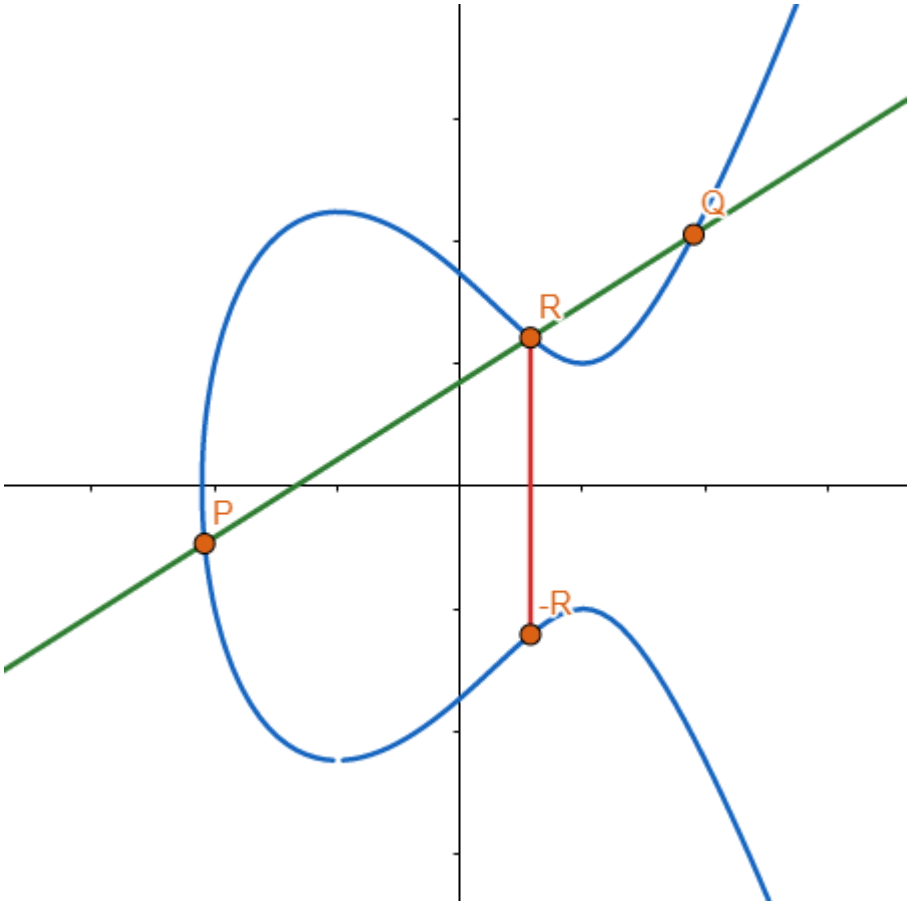

Рисунок 3 – Сложение на эллиптической кривой

Такой способ сложения работает в большинстве случаев, однако существуют некоторые особые случаи, которые также стоит рассмотреть.

При $P = O$ или $Q = O$ вступает в силу свойство нейтрального элемента. Если $P + O = P$ и $Q = O$, то $P + Q = P + O = P$.

При $P = -Q$ или $Q = -P$ прямая, проходящая через две точки вертикальна, и точки R не существует. Однако, так как эллиптическая кривая находится в проективных координатах, любая вертикальная прямая обязательно проходит через бесконечно удаленную точку O, что и будет являться результатом. Это можно подтвердить также с помощью арифметики: если $P = -Q$, то $P + Q = P + (-P)$. Как известно из свойств нейтрального элемента, $P + (-P) = O$.

При $P = Q$ точки находятся в одних и тех же координатах, следовательно через них проходит бесконечное число прямых. Чтобы вычислить R следует взять точку Q_0, не равную P. При приближении точки Q_0 к точке R прямая, проведенная через эти точки, приближается к касательной к эллиптической кривой. При наложении точек прямая становится равной касательной к эллиптической кривой. В таком случае, $P + Q = Q + Q = -R$, где R – точка пересечения эллиптической кривой и касательной к ней в точке Q. Процесс представлен на рисунке 4.

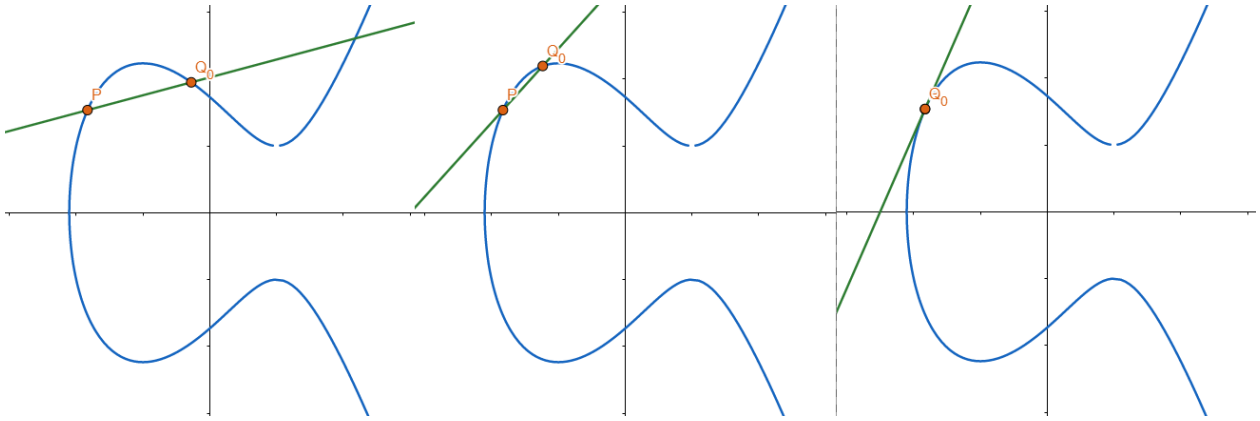

Рисунок 4 – Приближение к касательной к эллиптической кривой

При $P \neq Q$, но отсутствует третья точка пересечения с эллиптической кривой R, ситуация схожа с предыдущем случаем. Так как прямая проходит только через две точки, она касательна к эллиптической кривой. В таком случае результат сложения зависит от точки касания. Если точкой касания является P, то сложение принимает вид $P + Q = -P$. Если точкой касания является Q, то сложение принимает вид $Q + P = -Q$. Процесс представлен на рисунке 5.

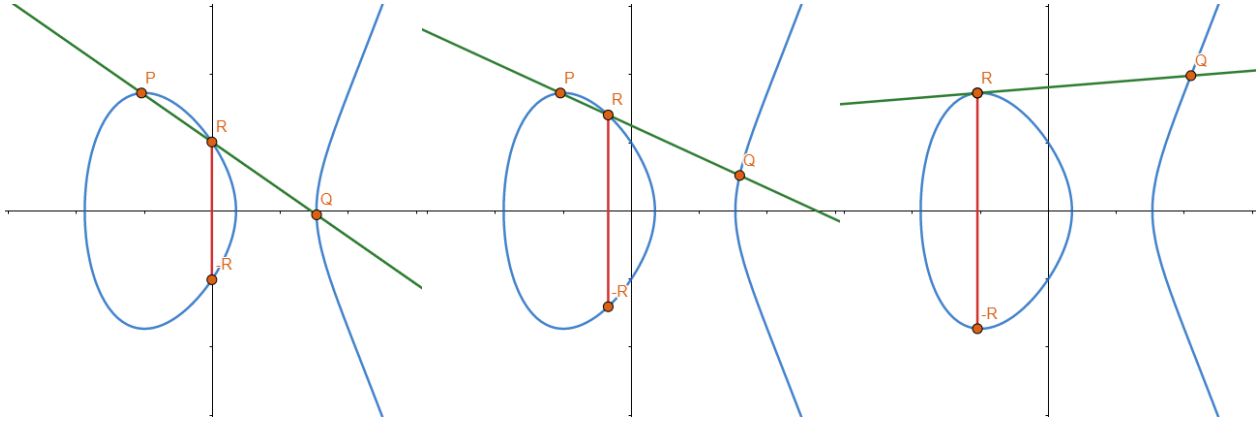

Рисунок 5 – Сложение с двумя точками

Можно заметить, что для сложения точек на эллиптической кривой используется графический метод. Такой подход не представляет проблем для единичных вычислений человеком, но для того, чтобы программно автоматизировать вычисления, необходимо представить их в более систематизированном, алгебраическом виде.

Существуют две точки P и Q. Через них проведена прямая. Уравнение прямой представлено на формуле (3).

$$y = kx + t. \tag{3}$$

Коэффициенты k и t этого уравнения можно найти, используя координаты точек, находящихся на этой прямой, то есть P и Q. Вычисления представлены на формулах (4) и (5).

$$k = \frac{y_P - y_Q}{x_P - x_Q}, \tag{4}$$

$$t = y_P - kx_P = y_Q - kx_Q. \tag{5}$$

Точки пересечения между двумя функциями можно найти, приравняв эти функции. Таким образом, для нахождения точек пересечения между прямой и эллиптической кривой, следует приравнять их так, как показано на формуле (6).

$$(kx + t)^2 = x^3 + ax + b. \tag{6}$$

Раскрытие выражения представлено на формуле (7).

$$\begin{gathered} (kx + y_P - kx_P)^2 = x^3 + ax + b, \\ (k(x - x_P) + y_P)^2 = x^3 + ax + b, \\ k^2(x^2 + x_P^2 - 2xx_P) + y_P^2 + 2y_P - 2y_P = x^3 + ax + b, \\ x^3 - k^2x^2 + (a - 2k^2x_P - 2y_P)x + (b + 2y_Px_P - y_P^2 - kx_P^2) = 0. \end{gathered} \tag{7}$$

Точки пересечения являются корнями кубического уравнения. Решение кубических уравнений зачастую занимает довольно много времени, однако в данном случае уже известны два корня: точки P и Q. Благодаря этому для нахождения последнего корня можно использовать формулы Виета.

Любой многочлен порядка n также имеет n корней. Формулы Виета показывают связь между коэффициентами многочлена и суммами и

произведениями корней многочлена. Формулы Виета представлены на формуле (8).

$$\begin{gathered} x_1 + x_2 + \cdots + x_n = \frac{-a_{n-1}}{a_n}, \\ x_1 x_2 \ldots x_n = (-1)^n \frac{a_0}{a_n}. \end{gathered} \tag{8}$$

Многочлен третьей степени, представленный на формуле (7), имеет три корня, два из которых известны. Следовательно, можно найти третий. Процесс нахождения третьего корня представлен на формуле (9).

$$\begin{gathered} x_P + x_Q + x_R = \frac{-(-k^2)}{1} = k^2, \\ x_R = k^2 - x_P - x_Q. \end{gathered} \tag{9}$$

Полученное значение можно подставить в изначальное уравнение прямой для получения второй координаты точки. Процесс представлен на формуле (10).

$$y_R = k(x_R - x_P) + y_P. \tag{10}$$

Таким образом, были найдены координаты точки R. Следует помнить, что результатом сложения точек на эллиптической кривой является не R, а $-R$, поэтому необходимо изменить знак -координаты найденной точки. Итоговые формулы представлены на формуле (11).

$$\begin{gathered} x = k^2 - x_P - x_Q, \\ y = -k(x - x_P) - y_P. \end{gathered} \tag{11}$$

Эти формулы применяются в том случае, если P и Q различны, включая тот случай, когда одна из точек является точкой касания к эллиптической кривой.

Необходимо вывести формулы и для остальных особых случаев, описанных ранее.

Бесконечно удаленная точка O обозначается как 0, соответственно $P + 0 = P$ и $P + (-P) = 0$ легко переводится в алгебраическую форму.

При $P = Q$ необходимо найти касательную прямую к эллиптической кривой, следовательно используется другая формула. Нужно вспомнить, что одно из определений производной функции гласит, что производная функции – это наклон касательной прямой к функции в определенной точке. Таким образом, производная подставляется на место переменной k формулы (10), что в результате дает формулу нахождения касательной прямой, представленную на формуле (12).

$$y = f'(x_P)(x - x_P) + f(x_P), \tag{12}$$

где $f(x)$ – формула эллиптической кривой.

Нахождение производной эллиптической кривой представлено на формуле (13).

$$f(x)' = \left(\pm\sqrt{x^3 + ax + b}\right)' = \frac{1}{2}\frac{\pm(3x^2 + a)}{\pm\sqrt{x^3 + ax + b}} = \frac{3x^2 + a}{2y}. \tag{13}$$

Результат подставляется в формулу (12) для выведения коэффициентов k и t, аналогичных тем, которые представлены на формулах (4) и (5). Коэффициенты представлены на формулах (14), (15) и (16).

$$y = \underbrace{\frac{3x_P^2 + a}{2y_P}}_{k} x + \underbrace{\left(y_P - \frac{3x_P^2 + a}{2y_P} x_P\right)}_{t}, \tag{14}$$

$$k = \frac{3x_P^2 + a}{2y_P}, \tag{15}$$

$$t = y_P - \frac{3x_P^2 + a}{2y_P} x_P. \tag{16}$$

Далее координаты R находятся аналогичным образом.

1.1.4 Скалярное умножение

В процессе вычислений на эллиптических кривых зачастую возникает необходимость производить операции скалярного умножения, то есть операции вида $A = nP$. По сути операция скалярного умножения аналогична n операциям сложения. При небольших размерах n эти вычисления не представляют проблемы, но с возрастанием n количество необходимых операций возрастает, что приводит к растрате ресурсов. Целесообразно определить метод скалярного умножения, который уменьшает количество необходимых операций.

Одним из таких методов является алгоритм «удвой и добавь» [9]. Алгоритм работает на основе двоичного представления множителя n.

Работа алгоритма «удвой и добавь» представлена на рисунке 6.

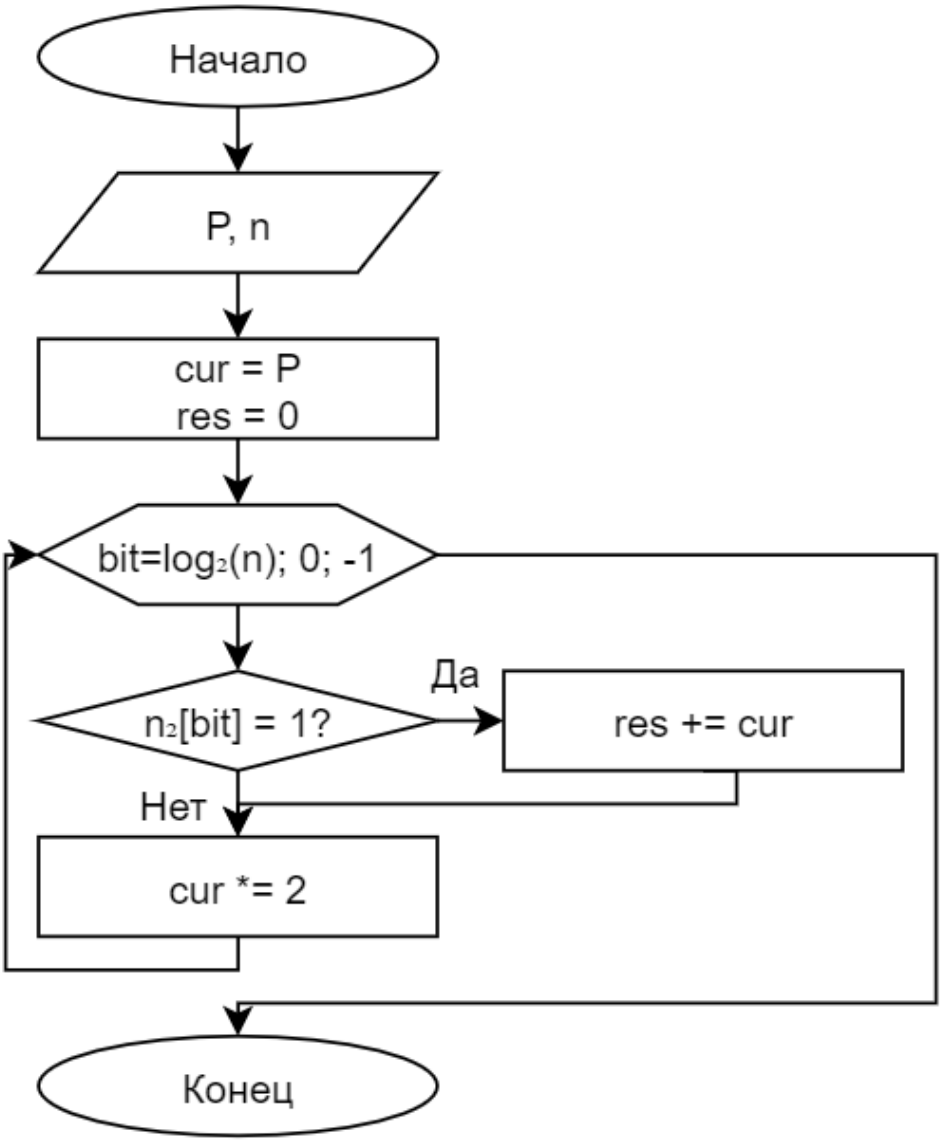

Рисунок 6 – Алгоритм «удвой и добавь»

Вычислительная сложность операции скалярного умножения в нормальных обстоятельствах, то есть если вычислять результат путем n сложений, составляет $O(n)$. Используя алгоритм «удвой и добавь», вычислительная сложность становится $O\big(log(n)\big)$, что является довольно хорошим улучшением по сравнению с обычным сложением.

1.1.5 Задача логарифмирования и ее значение в криптографии

В основе многих современных криптографических алгоритмов лежит так называемая односторонняя функция. Односторонняя функция представляет из себя операцию, которую легко выполнить, но невозможно обратить. Это свойство позволяет конструировать алгоритмы криптографии с открытым ключом, так как даже если ключ публично доступен, обратить функцию, которая использовалась для шифрования информации, невозможно.

Следует отметить, что из-за отсутствия строгого математического доказательства, существование истинных односторонних функций стоит под вопросом. Однако известны функции, которые достаточно приближены в своих свойствах к односторонним функциям, и поэтому носят это имя. Функция считается односторонней, если алгоритм обращения для нее еще не был составлен, или составленные алгоритмы занимают полиномиальное время выполнения на современных компьютерных системах.

Задача логарифмирования считается сложной. Имея некоторые n и P, достаточно легко вычислить $Q = nP$ без затраты большого количества ресурсов, что было продемонстрировано ранее алгоритмом «удвой и добавь». Однако при выполнении обратной операции, вычисления n при известных P и Q, возникают трудности. Задача логарифмирования на эллиптических кривых не является односторонней, но она является сложной. Для увеличения уровня безопасности, криптографические алгоритмы на эллиптических кривых используют вариант задачи логарифмирования, который считается односторонним.

При ограничении значений эллиптической кривой над конечным полем, операция скалярного умножения остается простой, а обратная операция логарифмирования становится намного более сложной.

1.2 Эллиптические кривые над конечными полями

1.2.1 Поле целых чисел по модулю p

Поле, а точнее конечное поле, это прежде всего множество с конечным количеством элементов. Типичный пример поля, а также поле, наиболее часто использующееся в криптографических алгоритмах – множество целых чисел по модулю p, где p – простое число. Такое поле обозначается как F_p.

На полях, в отличии от групп, определены две главные операции: сложение и умножение. Для того, чтобы множество M являлось полем, операции сложения и умножения должны быть определены таким образом, что они удовлетворяют следующим условиям:

- ассоциативность, как для сложения, так и для умножения: $a + (b + c) = (a + b) + c$ и $a \cdot (b \cdot c) = (a \cdot b) \cdot c$;
- коммутативность, как для сложения, так и для умножения: $a + b = b + a$ и $a \cdot b = b \cdot a$;
- существуют нейтральные элементы сложения и умножения 0 и 1 такие, что $a + 0 = a$ и $a \cdot 1 = a$;
- для каждого элемента a существует обратный элемент сложения $-a$ такой, что $a + (-a) = 0$;
- для каждого элемента $a \neq 0$ существует обратный элемент умножения a^{-1} такой, что $a \cdot a^{-1} = 1$;
- дистрибутивность: $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$.

Можно заметить, что обязательные свойства поля включают в себя все обязательные свойства абелевой группы. Следовательно, любое поле также является абелевой группой и сохраняет все свойства группы.

Поле целых чисел по модулю p состоит из множества всех целых чисел от 0 до $p - 1$. Операции сложения и умножения определены как сложение и умножение в модульной арифметике.

Для выполнения всех свойств поля, обязательно необходимо, чтобы p являлось простым числом. При других числах те или иные свойства не будут выполняться. Например, если взять F_4, то не будет выполняться требование обратного элемента (уравнение $2xmod4 = 1$ не имеет решений), то есть множество не будет являться полем.

Перед тем, как определять эллиптические кривые над конечными полями, следует определить, что из себя представляет деление на поле. На поле F_p деление x/y равно $x \cdot y^{-1}$, то есть произведение x и обратного элемента умножения для y. Из этого можно сделать вывод, что для того, чтобы произвести деление на конечном поле F_p, необходимо найти обратный элемент умножения и произвести одно умножение.

Нахождение обратного элемента умножения можно произвести с помощью расширенного алгоритма Евклида [18]. Расширенный алгоритм Евклида для нахождения обратного элемента умножения представлен на рисунке 7.

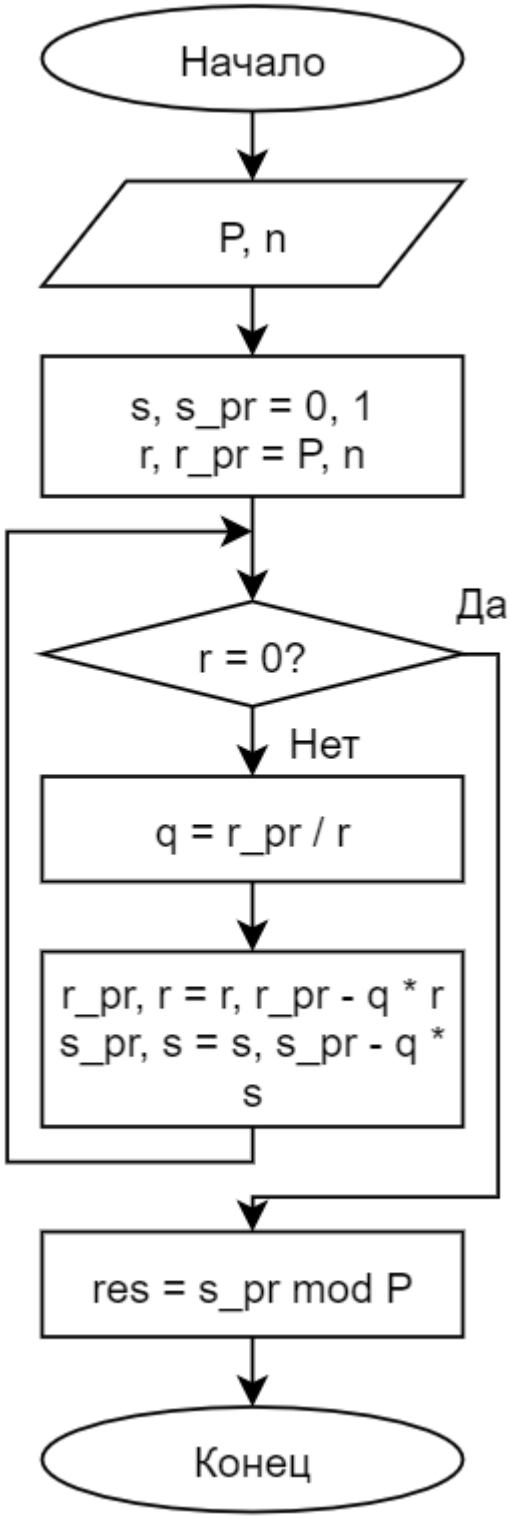

Рисунок 7 – Расширенный алгоритм Евклида для нахождения обратного элемента умножения

Расширенный алгоритм Евклида имеет вычислительную сложность $O(log(p))$ в худшем случае. Для операции деления следует также умножить результат работы алгоритма на x.

1.2.2 Эллиптические кривые над полем F_p

Эллиптическую кривую можно ограничить конечным полем. Для этого необходимо ввести операции по модулю. Определение эллиптической кривой, ограниченной конечным полем, представлено на формуле (17).

$$\{(x,y) \in F_p^2 \vee y^2 \equiv x^3 + ax + b(modp), 4a^3 + 27b^2 \not\equiv 0(modp)\} \cup \{0\}, \quad (17)$$

где 0 – бесконечно удаленная точка,

a, b – целые числа, принадлежащие F_p.

Вид эллиптической кривой над конечным полем представлен на рисунке 8.

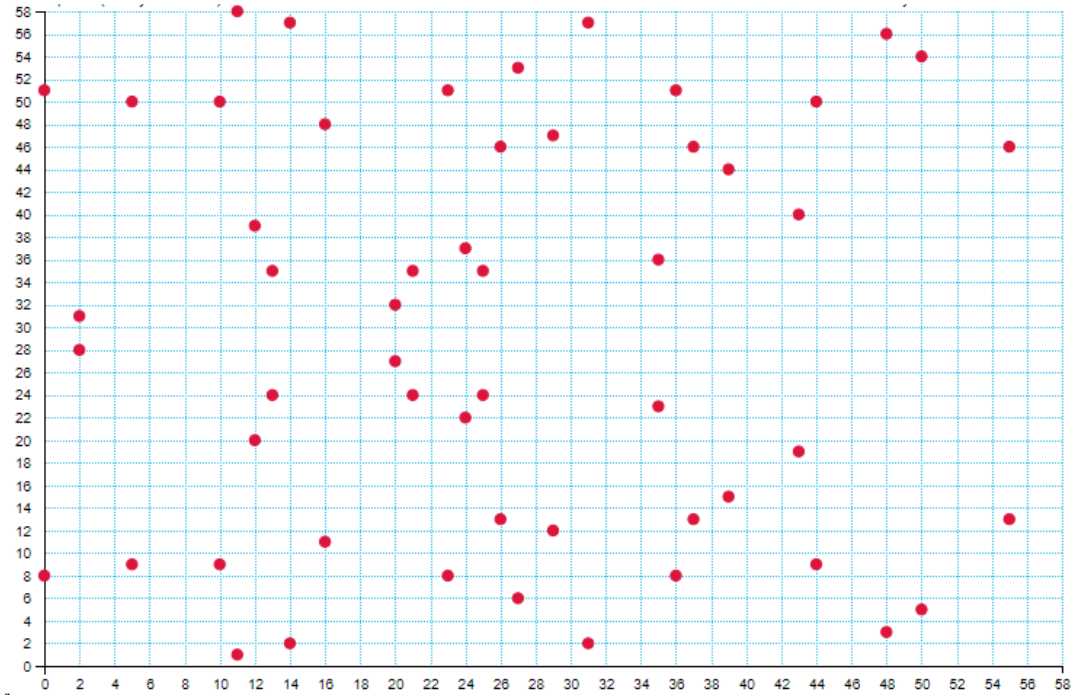

Рисунок 8 – Эллиптическая кривая на конечном поле, $p = 59$

Можно видеть, что гладкая кривая превратилась во множество отдельных точек на матрице [14]. Тем не менее, несмотря на непривычный вид, эллиптическая кривая сохраняет все свои свойства. По графику также видна симметрия, проходящая относительно прямой $y = p/2$.

1.2.3 Сложение точек на эллиптической кривой над конечным полем

Определение сложения точек на эллиптических кривых одинаково как для действительных чисел, так и для конечного поля. Три точки P, Q, R, лежащие на одной прямой, определяют операцию сложения как $P + Q + R = O$. Однако прямые, на которых лежат эти точки, выглядят по-разному на действительных числах и конечных полях.

Считается, что прямая на конечном поле F_p это множество точек (x, y), удовлетворяющих уравнению $ax + bx + c \equiv 0(modp)$, то есть типовое уравнение прямой, к которому добавлена операция модуля $(modp)$. Вид прямой на конечном поле представлен на рисунке 9.

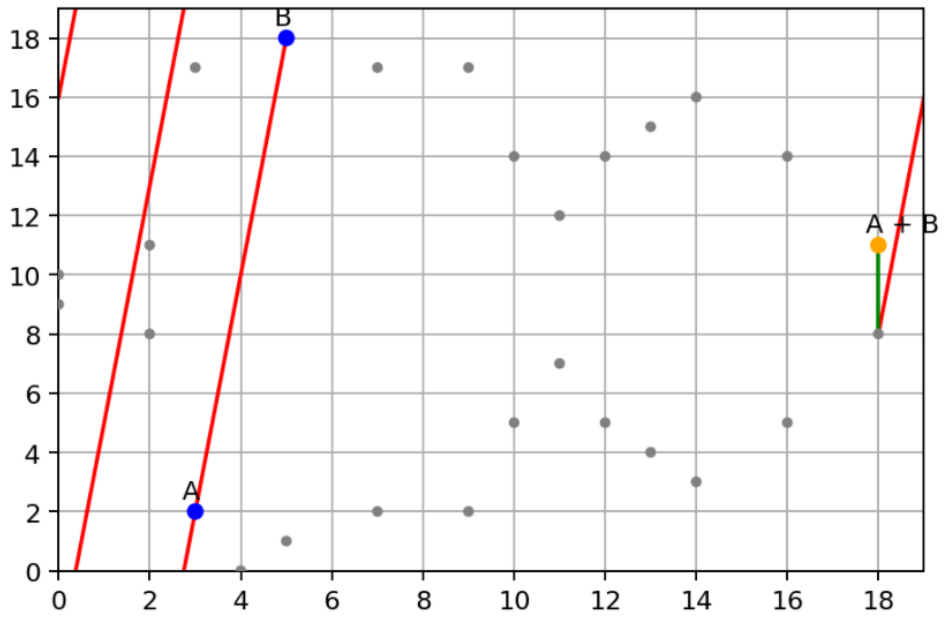

Рисунок 9 – Прямая на конечном поле и операция сложения точек

Из графика можно видеть, как прямая, соединяющая две точки, «повторяется» и образует множество параллельных прямых на поле.

Относительно самой операции сложения, все свойства, определенные в группе, сохраняются. Обратная точка симметрична не относительно оси x, а относительно прямой $y = p/2$ за счет того, что при отражении -координаты также выполняется операция по модулю $R(x, -ymodp)$.

Остальные случаи, такие как $P = Q$, сложно показать геометрически. Так как эти случаи требуют работать с касательными к кривой, отсутствие кривой в привычном виде затрудняет изображение графика сложения соответствующих точек.

Уравнения нахождения координат точки R над конечным полем идентичны уравнениям, полученным ранее, за исключением того, что в конце каждого выражения дописывается операция модуля $(modp)$. Таким образом, для точек P и Q сложение над конечным полем представлено на формулах (18) и (19).

$$x_R = \left(k^2 - x_P - x_Q\right) modp. \tag{18}$$

$$y_R = [y_P + k(x_R - x_P)] modp = \left[y_Q + k\left(x_R - x_Q\right)\right] modp. \tag{19}$$

Аналогично, при $P \neq Q$, наклон касательной k представлен на формуле (20).

$$k = \frac{y_P - y_Q}{x_P - x_Q} modp. \tag{20}$$

В случае $P = Q$, наклон касательной k принимает форму, представленную на формуле (21).

$$k = \frac{3x_P^2 + a}{2y_P} mod p. \quad (21)$$

Можно видеть, что все формулы сходятся с ранее выведенными. Более того, эти формулы будут правильны для любого поля, конечного или бесконечного, за исключением полей с характеристиками 2 и 3.

1.2.4 Порядок группы на эллиптической кривой

По определению, конечное поле имеет ограниченное число элементов. Следовательно, можно сделать вывод, что эллиптическая кривая, определенная над конечным полем, так же имеет ограниченное число точек. Подобная информация может понадобиться в последующих вычислениях, поэтому следует установить способ подсчета количества точек на эллиптической кривой.

Количество точек на эллиптической кривой называется порядком группы эллиптической кривой. Наиболее простым методом вычисления порядка группы является последовательная проверка всех точек конечного поля на принадлежность к эллиптической кривой. Такой способ надежен, но имеет вычислительную сложность $O(p)$, то есть занимает большое количество времени при больших значениях p.

Более быстрым способом вычисления количества точек на эллиптической кривой является алгоритм Шуфа [20]. Алгоритм Шуфа является первым детерминированным алгоритмом нахождения порядка группы эллиптической кривой за полиномиальное время. Подход Шуфа использует теорему Хассе для эллиптических кривых, китайскую теорему об остатках и многочлены деления.

Пусть E – эллиптическая кривая над полем F_p, заданная стандартной функцией Вейерштрасса. Теорема Хассе, представленная на формуле (22) показывает мощность группы точек.

$$\#E(F_p) = p + 1 - t, |t| \leq 2\sqrt{p}. \quad (22)$$

Пусть $\phi_p: E(\bar{F}_p) \to E(\bar{F}_p)$ так, что $\phi_p((x, y)) = (x^p, y^p)$. Это – карта точек с координатами в алгебраическом замыкании F_p. В таком случае ϕ_p является изоморфизмом под названием карта Фробениуса. Карты Фробениуса имеют одно свойство, важное для алгоритма Шуфа, представленное на формуле (23).

$$\phi_p^2 - t\phi_p + p = 0 \forall P \in E(\bar{F}_p). \quad (23)$$

Это свойство можно использовать для вычисления $t(modk)$ для множества N простых чисел, выполняющих условие, представленное на формуле (24).

$$K = \prod_{i=1}^{N} k_i > 2\sqrt{p}. \quad (24)$$

После этого примнется китайская теорема об остатках для нахождения $tmodK$, выполняющего условие, представленное на формуле (25).

$$|t| \leq 2\sqrt{p}. \quad (25)$$

После того, как известно t, можно вычислить порядок группы по формуле, представленной на формуле (26).

$$\#E(F_p) = p + 1 - t. \quad (26)$$

Шуф показал, что алгоритм имеет вычислительную сложность $O\left(log^9(p)\right)$, что не является наилучшим результатом, но тем не менее намного быстрее других вариантов, таких, как использование алгоритма baby-step-giant-step.

1.2.5 Циклические подгруппы и их характеристики

Ранее было установлено, что операция скалярного умножения $A = nP$ эквивалентна n операциям сложения. Это не меняется на конечных полях, и кроме того, для вычисления операций скалярного умножения все так же можно использовать алгоритм «удвой и добавь».

Однако скалярные умножения на эллиптических кривых над конечными полями обладают необычным свойством: они зацикливаются. При увеличении n ответы начинают повторяться, всегда в одном и том же порядке, что формирует цикл.

Это свойство проще всего продемонстрировать на примере. Пусть $y^2 = x^3 + 2a + 4(mod19)$ – эллиптическая кривая над полем F_{19}. Возьмем точку $P(5,14)$ на эллиптической кривой и найдем все ее произведения:

1. $0P = 0$;
2. $P = (5,14)$;
3. $2P = (13,2)$;
4. $3P = (8,0)$;
5. $4P = (13,17)$;
6. $5P = (5,5)$;
7. $6P = 0$;
8. $7P = (5,14)$;
9. $8P = (13,2)$;
10. $9P = (8,0)$;
11. $10P = (13,17)$;
12. $11P = (5,5)$.

Можно видеть, что ответы начинают повторяться при $n = 6$. Если продолжить умножение в этом же виде, произойдет еще один цикл. Продолжать умножение можно до бесконечности, результатом всегда будут эти шесть точек в этом порядке. Зная это, можно вывести формулы умножения:

1. $6kP = 0$;
2. $(6k + 1)P = P$;
3. $(6k + 2)P = 2P$;
4. $(6k + 3)P = 3P$;
5. $(6k + 4)P = 4P$;
6. $(6k + 5)P = 5P$.

Выражение можно упростить еще дальше, применив операцию модуля. Конечное выражение операции скалярного умножения для данного набора точек представлено на формуле (27).

$$kP = (kmod6)P, \quad (27)$$

где k – целое число.

Стоит отметить, что полученные повторяющиеся точки замкнуты относительно операции сложения. Не важно, сколько операций сложения производится, между этими точками, результатом всегда будет также одна из этих точек. Это легко подтвердить, разложив операции скалярного умножения. Пример приведен на формуле 28.

$$2P + 3P = (P + P) + (P + P + P) = P + P + P + P + P = 5P. \quad (28)$$

Можно видеть, что все операции умножения сводятся к операциям сложения, а операция сложения на этом множестве точек зациклена. Результат любого другого сложения этих точек будет аналогичным. Это

свойство говорит о том, что это множество точек является циклической подгруппой группы точек эллиптической кривой.

Подгруппой называют группу, являющуюся подмножеством другой группы. Циклической подгруппой называют такую подгруппу, в которой элементы циклически повторяются. Таким образом, описанное ранее множество, состоящее из точек 0, $(5,14)$, $(13,2)$, $(8,0)$, $(13,17)$, $(5,5)$ является циклической подгруппой группы точек, лежащих на эллиптической кривой $y^2 = x^3 + 2a + 4(mod19)$, что было показано ранее.

В циклической подгруппе точка P называется точкой-генератором или базисной точкой. Циклические подгруппы и их базисные точки составляют фундамент криптографических алгоритмов, основанных на эллиптических кривых [6].

Ранее был выведен алгоритм расчета порядка группы на эллиптической кривой, следует провести аналогичную операцию на подгруппах. К сожалению, для определения порядка подгруппы нельзя использовать алгоритм Шуфа, так как он работает только на полноценных эллиптических кривых. Однако существуют и другие способы определения порядка группы, и многие из них работают также на подгруппах.

Для того, чтобы приступить к вычислению порядка подгруппы, необходимо понимать, что внутри циклической подгруппы, порядок, кроме классического определения как количество точек в подгруппе, также может определяться как наименьшее положительное целое число такое, что $nP = 0$.

Порядок подгруппы связан с порядком группы с помощью теоремы Лагранжа. Теорема Лагранжа гласит, что для каждой конечной группы G, порядок каждой подгруппы H группы G делит порядок G.

Зная это, можно составить простой алгоритм нахождения порядка подгруппы с базисной точкой P:

– найти порядок группы эллиптической кривой N, используя алгоритм Шуфа;

– найти все делители N;
– для каждого делителя n порядка N, вычислить nP;
– наименьший n, при котором nP обращается в 0 является порядком подгруппы.

Особым случаем является эллиптическая кривая, порядок которой является простым числом. Когда это происходит, у порядка N появляются два единственных делителя: 1 и простое число N. В первом случае подгруппа содержит только бесконечно удаленную точку, а во втором случае подгруппа содержит всю группу.

Для использования в криптографических алгоритмах предпочитаются подгруппы больших порядков. Чем больше порядок подгруппы, тем безопаснее криптографический алгоритм. Следовательно, для построения криптографического алгоритма следует выбирать подгруппы больших порядков.

Однако единственным способом нахождения подгруппы большого порядка является перебор. Вычисляется подгруппа каждой точки эллиптической кривой, вычисляется ее порядок, и в конце выбирается та, которая имеет наибольшую защищенность. Такой подход довольно неэффективен.

Для того, чтобы облегчить задачу выбора подходящей базисной точки, целесообразно работать в обратном направлении: выбирается сначала желаемый порядок, а потом для него подбирается соответствующая точка на эллиптической кривой. Другими словами, выбирается эллиптическая кривая, вычисляется порядок ее группы, из делителей порядка выбирается достаточно большой, и для него подбирается соответствующая базисная точка.

В этом опять поможет теорема Лагранжа. Из теоремы Лагранжа следует, что число $h = N/n$ всегда целое, так как n является делителем N. Это число h носит название кофактора подгруппы. Для каждой точки P на

эллиптической кривой выражение $NP = O$ справедливо, так как N является некоторым количеством перемножений любого возможного n. Таким образом, $n(hP) = O$, что говорит о том, что точка $G = hP$ порождает подгруппу порядка n если n является простым числом. Единственными исключениями этому правилу являются случаи, когда n не является простым числом (тогда порядок подгруппы будет одним из делителей n) и когда $G \neq 0$ (тогда порядок подгруппы будет равен 1).

Используя это свойство, можно составить алгоритм нахождения базисной точки, порождающей подгруппу порядка n:

- найти порядок группы эллиптической кривой N, используя алгоритм Шуфа;
- выбрать простое число n, которое будет служить порядком подгруппы;
- вычислить кофактор подгруппы как $h = N/n$;
- Выбрать случайную точку P на эллиптической кривой;
- вычислить $G = hP$;
- если $G = 0$, то вернуться на шаг 4, в противном случае подходящая базисная точка была найдена, она породит подгруппу порядка n с кофактором h.

1.2.6 Задача дискретного логарифмирования

Аналогично работе с эллиптическими кривыми на действительных числах, для эллиптических кривых на конечных полях существует задача логарифмирования. Если известны точки P и Q на эллиптической кривой, как найти такое k, что $Q = kP$?

Эта задача называется задачей дискретного логарифмирования на эллиптических кривых [7]. На данный момент, задача дискретного логарифмирования считается односторонней функцией, так как не существует алгоритмов, которые могут ее решить за достаточно короткое

время на современных компьютерных системах [12]. Однако, как и для любых других функций, считающихся односторонними, математического доказательства ее односторонности не существует, и поэтому даже сейчас ведется ее криптоанализ с целью выведения полиномиального алгоритма ее решения.

Тем не менее, на данный момент задача дискретного логарифмирования является одной из самых сложных односторонних задач, и поэтому она широко используется при построении асимметричных криптографических алгоритмов. Многие системы вместо скалярного умножения используют слегка модифицированную задачу дискретного логарифмирования, где используются операции возведения в степень по модулю. Если известны точки a и b на эллиптической кривой, как найти такое k, что $b = a^k mod p$?

Обе эти задачи используют свойства эллиптических кривых над конечными полями, и поэтому имеют одно и то же название. Логарифмирование в этом случае называется дискретным, потому что задачи включают в себя работу с конечными множествами, то есть цикличными подгруппами.

Именно задача дискретного логарифмирования делает криптографические алгоритмы, основанные на эллиптических кривых, одними из самых надежных в современном мире. Кроме того, задача дискретного логарифмирования позволяет достигать большого уровня защищенности с меньшими затратами памяти. Эти качества делают криптографию на эллиптических кривых одним из самых перспективных направлений защиты информации.

Выводы по итогам первой главы: эллиптическая кривая является особым случаем кубической кривой, эллиптическая кривая является группой, сложение точек на эллиптической кривой происходит посредством нахождения точки пересечения между кривой и прямой, соединяющей две точки, скалярное умножение точек на эллиптической кривой производится с

помощью алгоритма «удвой и добавь», группа эллиптической кривой над конечным полем образует циклические подгруппы, задача дискретного логарифмирования на эллиптической кривой является односторонней функцией, использующейся в криптографии.

Глава 2 Алгоритмы эллиптической криптографии

2.1 Обмен ключами – ECDH

Шифрование данных всегда было важной частью протоколов передачи информации. Эти протоколы требуют, чтобы у двух сообщающихся сторон, был общий ключ для зашифровки и расшифровки переданной и полученной информации. Однако возникает проблема безопасной передачи ключа. Сообщающиеся стороны зачастую находятся далеко друг от друга, но передача ключа обычными методами не может быть произведена, ведь если ключ кто-то перехватит, то он сможет получить доступ ко всей информации, которая передается. Для решения этой проблемы и служат алгоритмы обмена ключами.

В криптографии эллиптических кривых одним из фундаментальных алгоритмов является обмен ключами Диффи-Хеллмана на эллиптических кривых (ECDH) [15].

Перед шифрованием, стороны A и B соглашаются на эллиптической кривой E порядка n и базисной точке P на E. Предполагается, что эти данные известны публично. После этого, генерация ключей проходит по алгоритму:

- A случайно выбирает свой личный ключ a – целое число от 2 до $n-1$;
- A вычисляет свой открытый ключ как aP;
- B случайно выбирает свой личный ключ b – целое число от 2 до $n-1$;
- B вычисляет свой открытый ключ как bP;
- A и B обмениваются своими открытыми ключами по незащищенному каналу;
- A вычисляет секрет как $a \cdot bP$;

– В вычисляет секрет как $b \cdot aP$.

Графически алгоритм представлен на рисунке 10.

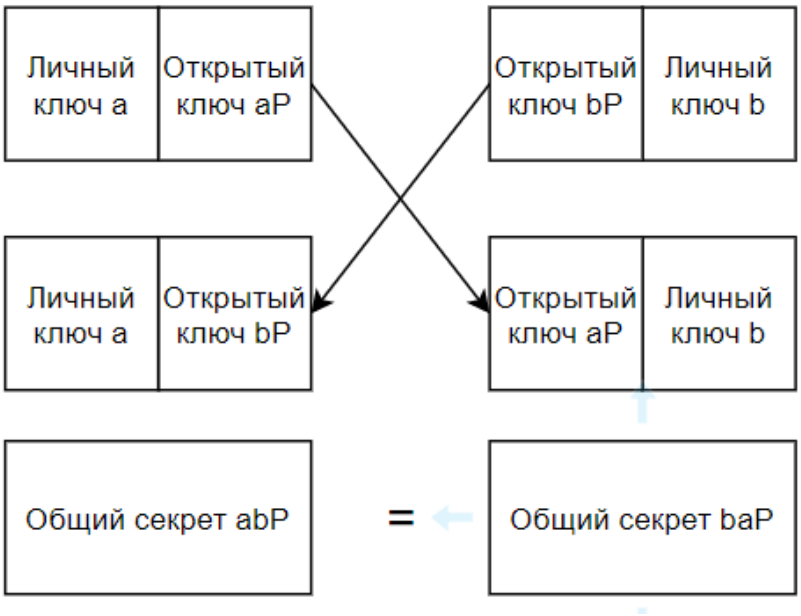

Рисунок 10 – Алгоритм ECDH

После выполнения алгоритма у A и B будет идентичный секрет abP, который можно использовать как ключ для алгоритма шифрования и расшифрования [13].

2.2 Цифровая подпись – ECDSA

Цифровая подпись позволяет пользователям получать информацию в форме документов, писем, приложений, зная ее источник. Другими словами, цифровая подпись позволяет аутентифицировать свою информацию перед другой стороной и показать, что она является целостной, невредоносной и не была никак модифицирована.

Алгоритм ставки цифровой подписи на основе эллиптических кривых (ECDSA) [16] происходит как:

– вычислить хэш сообщения h, используя криптографически сильную хэш-функцию (например SHA-256);
– выбрать случайно k – целое число от 1 до $n-1$;
– вычислить точку $R = kP$ и взять ее x-координату $r = R.x$ по модулю n;
– если полученное r равно нулю, вернуться на шаг 2;
– вычислить доказательство подписи $s = k^{-1} \cdot (h + r \cdot key)(mod n)$;
– если полученное r равно нулю, вернуться на шаг 2.

Подписью является пара целых чисел (r, s) от 1 до $n-1$. Подпись показывает точку R в r, а так же доказывает, что подписывающий знает сообщение через хэш h и ключ key. Доказательство s можно подтвердить с помощью соответствующего отрытого ключа.

Верификация подписи использует сообщение, подпись, полученную в результате работы алгоритма подписания, а также открытый ключ. Верификация происходит по следующему алгоритму:

1. вычислить хэш сообщения, используя ту же хэш-функцию, использованную при подписании;

2. вычислить обратное значение по модулю от доказательства подписи $s_i = s^{-1}(mod n)$;

3. восстановить случайную точку, использовавшуюся при подписании: $R_i = (h \cdot s_i)P + (r \cdot s_i)key$ и взять ее x-координату $r_i = R_i.x$;

4. если r и r_i равны, то подпись подтвердилась, в противном случае подпись не подтвердилась.

В целом идея проверки подписи состоит в восстановлении точки R_i и с помощью открытого ключа. Если R_i равна R, взятой случайно по время подписания, то подпись подтверждается [4].

Графически алгоритм представлен на рисунке 11.

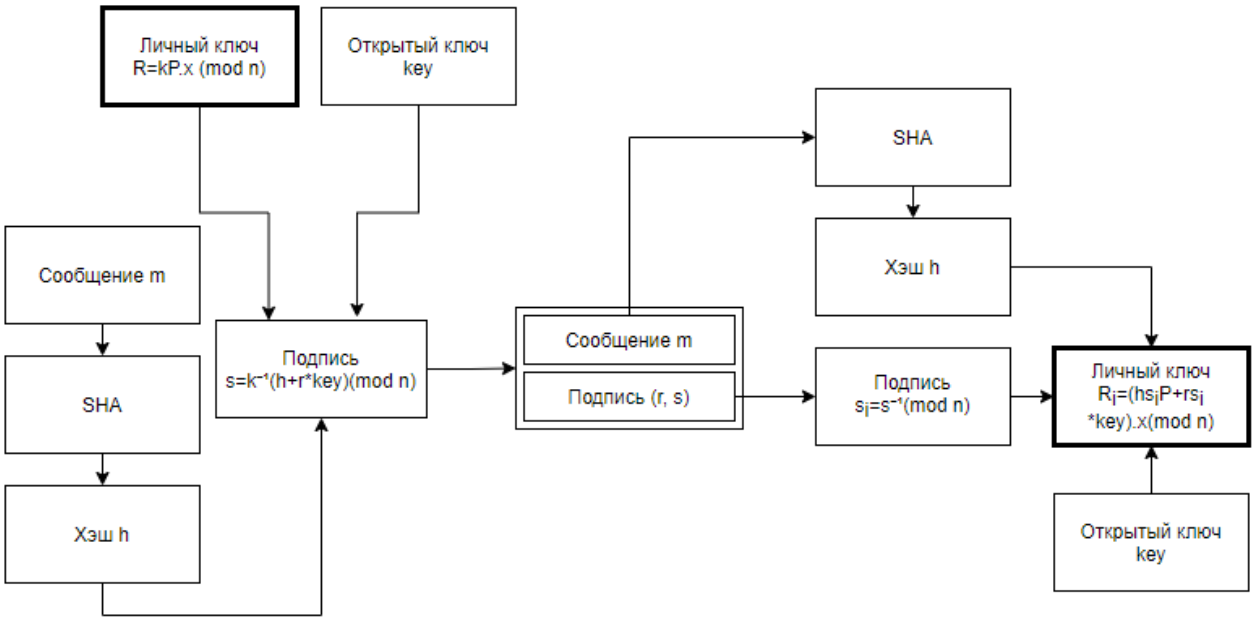

Рисунок 11 – Алгоритм ECDSA

Алгоритм цифровой подписи является незаменимой частью инфраструктуры открытых ключей (PIK).

2.3 Смешанные алгоритмы – ECIES

Самой широкоохватывающей системой шифрования, основанной на криптографии на эллиптических кривых, является Интегрированная Схема Шифрования на Эллиптических Кривых (ECIES). Эта схема является вариантом схемы Эль Гамаля, предложенной Абдалла, Белларе и Рожвэем.

Как следует из названия, ECIES – интегрированная схема шифрования [1], в которой используются следующие алгоритмы:

- ключевое соглашение (KA) – алгоритм, использующийся для составления общего секрета между двумя сторонами во время сообщения;
- функция выведения ключа (KDF) – алгоритм, производящий набор ключей из определенного ключевого материала и некоторых дополнительных параметров;

– шифрование (ENC) – симметричный алгоритм шифрования;
– код аутентификации сообщения (MAC) – некоторые данные, использующиеся для аутентификации сообщений;
– хэш (HASH) – алгоритм переработки, использующийся в алгоритмах KDF и функциях MAC.

В алгоритме участвуют две сообщающиеся стороны, A и B. A и B имеют открытые и личные ключи u, U и v, V соответственно. Личные ключи являются элементами конечного поля, а открытые ключи являются результатом произведения личных ключей с базисной точкой G.

Шифрование ECIES происходит по следующему алгоритму:

– A создает пару эфемерных ключей, состоящую из точки на конечном поле u и точки на эллиптической кривой uG;
– после генерации ключей A использует функцию ключевого соглашения для создания общего секрета uV;
– A использует секрет и по желанию другие параметры (например, двоичное представление U) как входные данные для алгоритма KDF, получая в результате строку, состоящую из двух ключей – ключа для шифрования k_{ENC} и ключа для алгоритма MAC k_{MAC};
– A шифрует сообщение m с помощью алгоритма шифрования ENC и ключа k_{ENC}, получая зашифрованное сообщение c;
– используя c, k_{MAC} и по желанию другие параметры (например фраза, согласованная заранее с B), A использует функцию MAC для получения тэга;
– A пересылает строку ($U||$тэг$||$c) B по незащищенному каналу.

Расшифрование происходит по следующему алгоритму:

– B разделяет строку на U, тэг и c;
– B использует U и свой личный ключ v для генерации общего секрета vU, который идентичен тому, что был у A;

– с помощью секрета vU и идентичных A дополнительных параметров, B использует функцию KDF и получает k_{ENC} и k_{MAC};
– используя k_{MAC}, с и идентичные A дополнительные параметры B получает тэг сообщения и сравнивает его с полученным от A тэгом, отвергая сообщение если они не совпадают;
– B применяет алгоритм ENC с ключом k_{ENC} для расшифровки с.

Графически алгоритм представлен на рисунке 12.

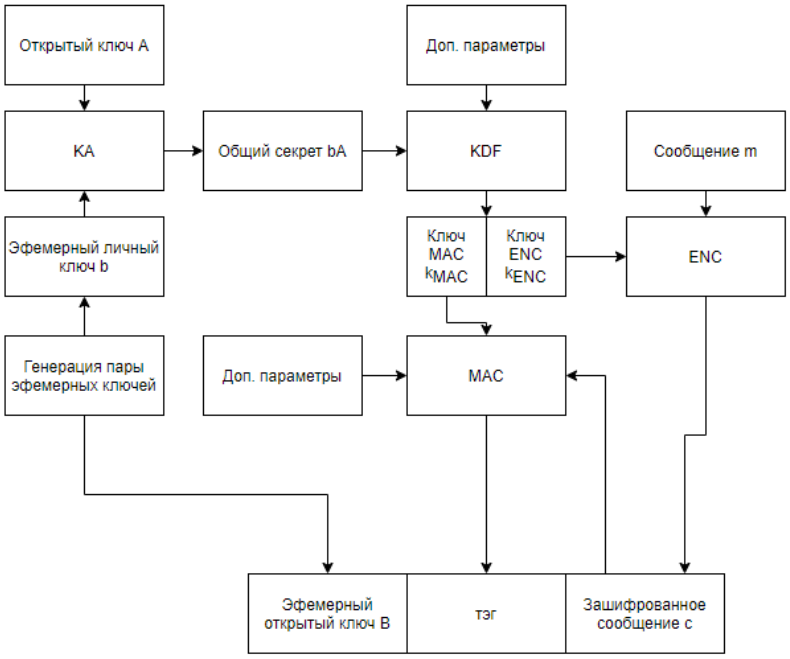

Рисунок 12 – Алгоритм ECIES (этап шифрования)

Не смотря на преимущества ECIES перед другими схемами шифрования, различия реализации между стандартами препятствует ее распространению.

Выводы по итогам второй главы: обмен ключами производится для безопасной передачи ключа шифрования, в криптографии на эллиптических кривых обмен ключами производится с помощью алгоритма ECDH, цифровая подпись необходима для подтверждения целостности информации после передачи, в криптографии на эллиптических кривых цифровая подпись

реализуется с помощью алгоритма ECDSA, смешанные алгоритмы представляют из семя комплекс нескольких алгоритмов, которые вместе направлены на исполнение единой задачи, наиболее распространенным смешанным алгоритмом в криптографии на эллиптических кривых является ECIES, который применяется для шифрования.

Глава 3 Программная реализация алгоритмов

3.1 Выбор средств

Для реализации алгоритмов был выбран язык программирования высокого уровня C++ [2, 3, 8].

В разработке использовалась библиотека C++ Givaro [19]. Givaro – библиотека для арифметических и алгебраических вычислений. Основной особенностью библиотеки является имплементация арифметики для многих математических объектов, таких как поля простых чисел, конечные поля, конечные кольца, многочлены и многие другие. Она также предоставляет структуры данных и классы для управления простыми алгебраическими объектами, такими как векторы и матрицы.

Из библиотеки Givaro используются следующие компоненты:

- ZpzDom<Integer> – поле простого числа Z_p, где p – простое число;
- Poly1Dom – для представления многочленов, принадлежащих $F_p[x]$, где p – простое число, и операций над ними;
- Integer – для представления длинных целых чисел (внутренне используется gmp) и операций над ними.

Помимо библиотеки Givaro, не использовалось более никаких сторонних ресурсов. Исходный код написан на чистом C++.

3.2 Структура проекта

Проект состоит из ряда классов и функций. Классы затрагивают математическую теорию и имплементируют составные части теории эллиптических кривых и арифметических операций над ними. Функции имплементируют алгоритмы криптографии, в которых используются вышеупомянутые классы. Весь код соответственно распределен по

заголовочным файлам .h, в которых описаны прототипы классов и функций, и по исходным файлам .cpp, в которых производится прописание прототипов. Соединение различных файлов происходит с помощью директив компилятору типа #include и ворот типа #infdef-#endif. Сам проект имеет вид заголовочной библиотеки.

Полный состав проекта:

- EC.h и EC.cpp – содержат классы конечных полей Fld, точек на эллиптических кривых ECpt, эллиптических кривых ElC и групп эллиптических кривых на поле ECgroup;
- ECC.h и ECC.cpp – содержат функции, реализующие криптографические алгоритмы на эллиптических кривых ECDH и ECDSA, а также вспомогательные объекты, такие как структуры подписей и ключей.

3.3 Реализация эллиптических кривых

Эллиптические кривые реализованы через иерархию классов. Каждый класс описывает отдельный аспект эллиптических кривых. Всего четыре класса:

- Fld – конечное поле и арифметика на поле;
- ECpt – точка на эллиптической кривой;
- ElC – эллиптическая кривая;
- ECgroup – группа эллиптической кривой над полем и арифметика в группе.

Диаграмма класса Fld представлена на рисунке 13.

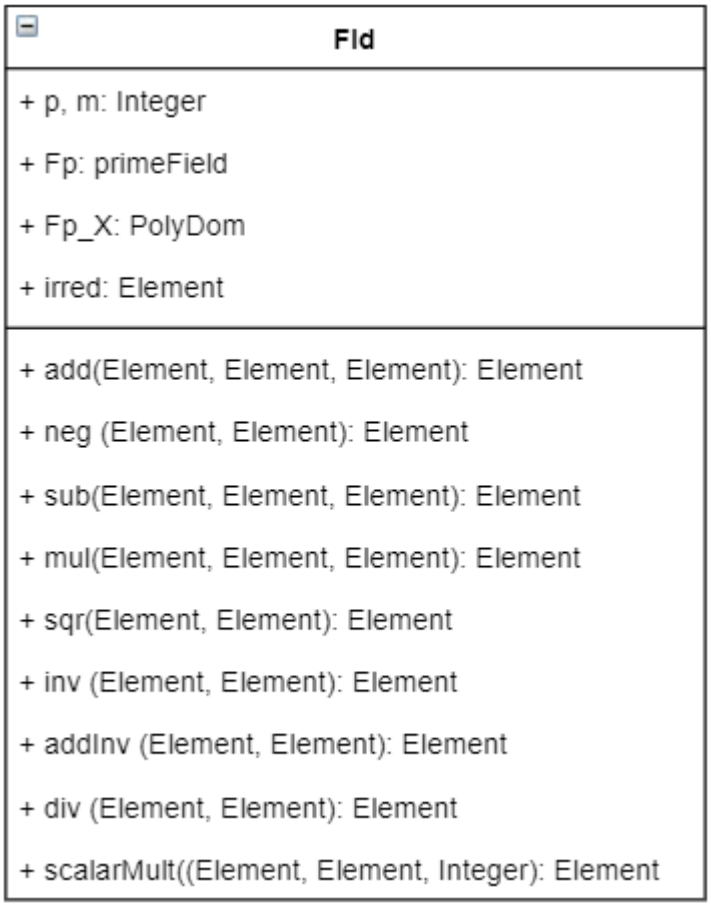

Рисунок 13 – Диаграмма класса Fld

Состав класса:

– p, m – характеристики поля F_{p^m}, при этом p – простое число и m>0;

– Fp – поле F_{p^m};

– Fp_X – многочлены над полем;

– irred – неприводимый многочлен.

Все элементы поля представлены в виде многочленов. Благодаря этому, для произведения арифметических операций над полем можно использовать встроенные функции библиотеки Givaro. Неприводимый многочлен также нужен для определенной операции.

Методы класса:

– add – сложение;

– neg – отрицание;

– inv – нахождение обратного элемента;

– addInv – сложение с обратным элементом;

– div – деление;

– scalarMult – скалярное умножение.

Все методы, кроме скалярного умножения, по сути являются обертками аналогичных методом библиотеки Givaro. Эти простые методы представлены на рисунке 14.

```
Fld::Element& Fld::add(Element& R, const Element& A, const Element& B) const
{
    Fp_X.add(R, A, B);
    return R;

Fld::Element& Fld::neg(Element& R, const Element& A) const
{
    Fp_X.neg(R, A);
    return R;
}
Fld::Element& Fld::inv(Element& I, const Element& A) const
{
    Element tmp, D;
    D = Fp_X.gcd(D, I, tmp, A, irred);
    if (Fp_X.isOne(D))
    {
        return I;
    }
    Fp_X.assign(I, Fp_X.zero);
    return I;
}
Fld::Element& Fld::addInv(Element& I, const Element& A) const
{
    Fp_X.sub(I, Fp_X.zero, A);
    return I;

}
```

Рисунок 14 – Методы класса Fld

В отличие от остальных методов, scalarMult представляет из себя реализацию алгоритма «удвой и добавь», о котором говорилось ранее. Метод представлен на рисунке 15.

```
Fld::Element& Fld::scalarMult(Element& R, const Element& A, Integer k) const
{
    k = k % p;
    if(m == 1)
    {
        Element eK;
        primeField::Element tmp;
        Fp_X.assign(eK, Fp.init(tmp,k));
        Fp_X.mul(R, A, eK);
        return R;
    }
    Element tmp, acc = A;
    Fp_X.assign(R, zero);
    while(k > 0)
    {
        if(k % 2)
        {
            Fp_X.add(R, R, acc);
        }
        acc = Fp_X.add(tmp, acc, acc);
        k = k / 2;
    }
    return R;
}
```

Рисунок 15 – Алгоритм «удвой и добавь», реализованный в методе scalarMult

Диаграмма класса ECpt представлена на рисунке 16.

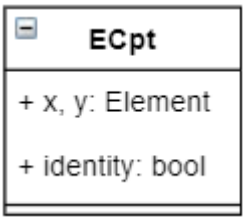

Рисунок 16 – Диаграмма класса ECpt

Состав класса:

– x, y – координаты точки;

– identity – является ли точка нейтральным элементом.

У этого класса определены лишь конструкторы и перегружены операторы сравнения. Стоит заметить, что если точка является нейтральным элементом, координаты перестают иметь значение. Более того, нейтральный элемент всегда наименьший в группе.

Перегруженные операторы сравнения представлены на рисунке 17.

```
bool ECpt::operator==(const ECpt& t) const
{
    return (identity && t.identity) || (!identity && !t.identity && x == t.x && y == t.y);
}

bool ECpt::operator< (const ECpt& t) const
{
    return identity ? !t.identity : (!t.identity && (x < t.x || (x == t.x && y < t.y)));
}
```

Рисунок 17 – Перегруженные операции сравнения класса Ecpt

Диаграмма класса ElC представлена на рисунке 18.

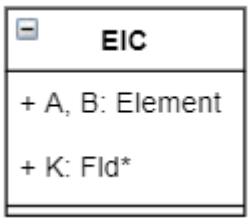

Рисунок 18 – Диаграмма класса ElC

Состав класса:

– A, B – коэффициенты уравнения Вейерштрасса;

– K – поле, над которым лежит эллиптическая кривая.

Аналогично классу ECpt, у класса ElC определены лишь конструкторы. С этим классом не подразумевается взаимодействовать напрямую, он лишь используется в других процедурах.

Диаграмма класса ECgroup представлена на рисунке 19.

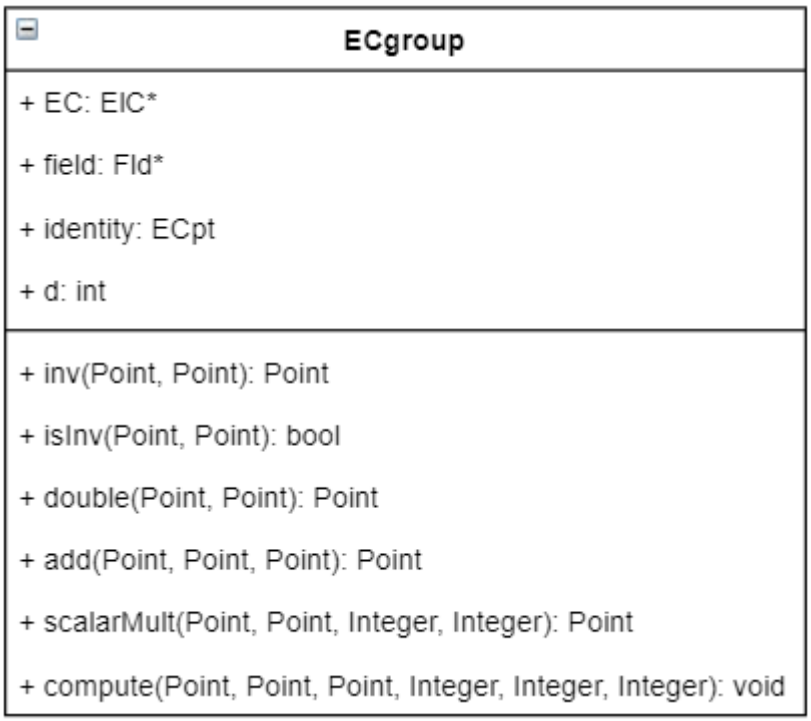

Рисунок 19 – Диаграмма класса ECgroup

Состав класса:

- EC – эллиптическая кривая;
- field – поле, над которым лежит эллиптическая кривая;
- identity – нейтральный элемент;
- d – ранг поля;

Можно видеть, что класс ECgroup группирует все остальные классы и интегрирует их для взаимодействия.

Методы класса:

- inv – обращение точки;
- isInv – проверяет, является ли одна точка обратной другой;
- Double – удвоение точки;
- add – сложение точек;
- scalarMult – скалярное умножение;

– compute – складывает два скалярных произведения.

После проверок на нейтральный элемент, метод inv использует метод арифметики на поле класса Fld addInv для расчета обратного значения точки.

Метод inv используется в методе isInv для проверки обратности. Если обратить точку и сравнить ее со второй точкой, они совпадут в том случае, если первая точка является обратной точкой для второй. Методы, связанные с обращением, представлены на рисунке 20.

```
const ECgroup::Point& ECgroup::inv(Point& Q, const Point &P)
{
    if(P.identity)
    {
        Q.identity = true;
        return Q;
    }
    Q.identity = false;
    Point T;
    Q.x = P.x;
    field->addInv(Q.y, P.y); //Q.y+P.y=0
    return Q;

}
bool ECgroup::isInv(const Point& Q, const Point &P)
{
    Point R;
    inv(R,P);
    if(R == Q)
    {
        return true;
    }
    return false;
}
```

Рисунок 20 – методы inv и isInv, связанные с обращением точек

Метод add, отвечающий за сложение, также использует арифметику на поле класса Fld. Способ его работы соответствует формулам, которые были описаны ранее. Метод сложения add представлен на рисунке 21.

```
ECgroup::Point& ECgroup::add(Point &R,Point &P, Point &Q)
{
    {
    }

    fieldElement x,y;
    fieldElement k, k2;
    fieldElement k_up, k_dn, xpq, xxp, kx;

    //k=(yp-yq)/(xp-yq)
    field->sub(k_up, Q.y, P.y);
    field->sub(k_dn, Q.x, P.x);
    field->div(k, k_up, k_dn);

    //k^2
    field->sqr(k2, k);

    //xr=k^2-xp-xq
    field->add(xpq, P.x, Q.x);
    field->sub(x, k2, xpq);

    //yr=-(yp+k(xr-xp))
    field->sub(xxp, P.x, x);
    field->mul(kx, k, xxp);
    field->sub(y, kx, P.y);

    R.x = x;
    R.y = y;
    return R;
}
```

Рисунок 21 – Метод add, складывающий две точки на конечном поле

Метод Double также отвечает за сложение, однако используется в особом случае, когда $P = Q$, то есть существуют только две точки. Аналогично методу add, метод Double использует арифметику на полях класса Fld, а способ его работы соответствует формулам, которые были описаны ранее. Метод сложения с двумя точками Double представлен на рисунке 22.

```
ECgroup::Point& ECgroup::Double(Point &R,Point &P)
{
    {
    }
    fieldElement x, y, xp2;
    fieldElement k, k2;
    fieldElement _3xp2, _2yp, k_up, _2xp, xxp;

    //k=(3xp^2+a)/(2yp)
    field->sqr(xp2, P.x);
    field->scalarMultiply(_3xp2, xp2, 3);
    field->scalarMultiply(_2yp, P.y, 2);
    field->add(k_up, _3xp2, ec->A);
    field->div(k, k_up, _2yp);

    //k^2
    field->sqr(k2, k);

    //xr=k2-2xp
    field->scalarMultiply(_2xp, P.x, 2);
    field->sub(x, k2, _2xp);

    //yr=-(yp+k(xr-xp))
    field->sub(xxp, P.x, x);
    field->mul(kx, k, xxp)
    field->sub(y, kx, P.y);

    R.x = x;
    R.y = y;
    return R;
}
```

Рисунок 22 – Метод Double, производящий сложение при особом случае $P = Q$

Метод scalarMult производит скалярное умножение и очень схож с аналогичным методом класса Fld. Как и тот метод, он является реализацией алгоритма «удвой и добавь», о котором говорилось ранее, с дополнительным учетом некоторых тонкостей, которые приносит группа. Главное отличие состоит в том, что этот метод использует выше описанные методы add и Double в своих вычислениях. Метод представлен на рисунке 23.

```
ECgroup::Point& ECgroup::scalarMult(Point& R, Point& P,
                                    Integer k, Integer order)
{
    if (P.identity)
    {
        R.identity = true;
        return R;
    }
    Point temp(false);
    R.identity = true;
    Point acc(P);
    if(order != -1)
    {
        k = k % order;
    }
    while(k > 0)
    {
        if(k % 2)
        {
            add(temp, R, acc);
            R = temp;
        }
        Double(temp, acc);
        acc = temp;
        k = k / 2;
    }
    return R;
}
```

Рисунок 23 – Метод scalarMult, производящий скалярное умножение в группе

Наконец, метод compute является методом-утилитой. Он используется в реализации алгоритма ECDSA и существует он для того, чтобы не захламлять код самого алгоритма ECDSA. Этот метод выполняет два скалярных умножения и складывает их результаты. Метод представлен на рисунке 24.

```
void ECgroup::compute(ECpt& R, Integer a,
                      ECpt& P, Integer b,
                      ECpt& Q, Integer n)
{
    ECpt t1,t2;
    scalarMult(t1, P, a, n);
    scalarMult(t2, Q, b, n);
    add(R, t1, t2);
}
```

Рисунок 24 – Метод compute

Была описана структура классов, реализующих эллиптические кривые и операции над ними. Можно видеть, что все классы тесно между собой связаны. Полная иерархия классов представлена на рисунке 25.

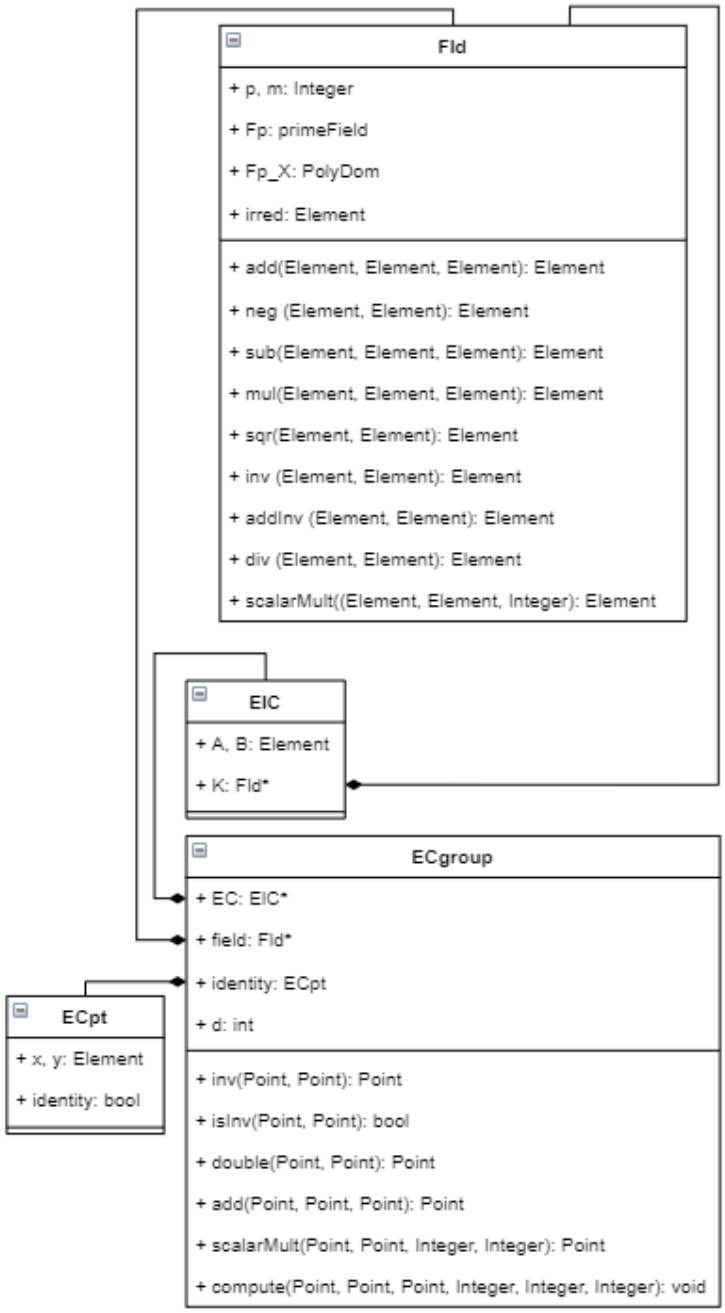

Рисунок 25 – Иерархия классов, реализующих эллиптические кривые

Можно видеть, что классы связаны отношениями композиции, то есть подчиненные классы Fld, ElC, ECpt не могут существовать без содержащего их класса ECgroup.

3.4 Реализация ECDH

Реализация алгоритма ECDH состоит из одной функции.

На вход подаются группа эллиптической кривой EC, порядок группы n, базисная точка P, а также переменные для хранения личных и открытых ключей a, b, Ka и Kb. После выполнения функции в переменные Ka и Kb будет записан секрет.

Функция самостоятельно выбирает личные ключи a и b, используя функцию библиотеки Givaro random_lessthan. После этого используя функцию скалярного умножения scalarMult, определенную ранее, находятся значения Pa и Pb – открытые ключи. Далее открытые ключи перемножаются с противоположными личными ключами, и результат записывается в переменные Ka и Kb. При правильной работе алгоритма Ka и Kb будут идентичны.

Функция представлена на рисунке 26.

```
void ECDH(Integer& a, Integer& b,
          ECpt& Ka, ECpt& Kb,
          ECpt& P, ECgroup& EC, Integer& n)
{

    ECpt Pa, Pb;

    Integer::random_lessthan(a, n);
    EC.scalarMult(Pa, P, a, n);

    Integer::random_lessthan(b, n);
    EC.scalarMult(Pb, P, b, n);

    EC.scalarMult(Ka, Pb, a, n);
    EC.scalarMult(Kb, Pa, b, n);

}
```

Рисунок 26 – Функция ECHD

Такая имплементация функции позволяет видеть все использующиеся параметры, однако для скртия параметров достаточно лишь сократить подпись функции до Ka, Kb, P, EC, n.

3.5 Реализация ECDSA

Реализация алгоритма ECDSA состоит из двух функций – ECDSAsign для подписания сообщений и ECDSAverify для подтверждения подписей.

Функции ECDSAsign на вход подаются личный ключ K, сообщение m, группа эллиптической кривой EC, порядок группы n, базисная точка P, а также переменная S для хранения результирующей подписи. После выполнения функции в переменную S будет записана результирующая подпись.

Функции выбирает случайное число k, используя функцию библиотеки Givaro random_lessthan, и находит результат скалярного умножения Pk, используя функцию scalarMult. С помощью функции toInt x-координата

результирующей точки преобразовывается в целочисленный тип для последующей операции деления по модулю. Последним шагом происходит вычисление доказательства подписи $s = k^{-1} \cdot (h + r \cdot key)(mod n)$. Результат записывается в переменную S.

Операции находятся в двух циклах с постусловиями, которые проверяют состояние подписи. Циклы вложены таким образом, что если какая-либо часть подписи обратится в ноль, весь процесс начинается сначала. Циклы будут работать, пока обе части подпили не примут ненулевые значения.

Функция представлена на рисунке 27.

```
void ECDSAsign(SignatureECDSA& S, Key& K, Integer& m,
               ECpt& P, ECgroup& EC, Integer& n);
{
    Integer k, _x = 1, h;
    ECpt temp;
    h = H(message, n);

    do
    {
        do
        {
            Integer::random_lessthan(k, n);

            EC.scalarMult(temp, P, k, n);

            toInt(_x, temp.x, n, EC);
            S.r = _x % n;

        } while(S.r == 0);

        invin(k, n);
        S.s = (k * (h + S.r * K.d)) % n;

    } while(S.s == 0);
}
```

Рисунок 27 – Функция ECDSAsign

Функции ECDSAverify на вход подаются сообщение m, группа эллиптической кривой EC, порядок группы n, базисная точка P, а также переменная S для хранения результирующей подписи. Кроме того, на вход также подается точка R, которая является произведением P и личного ключа.

В первую очередь функция проверяет принадлежность числовых значений подписи к полю. Если подпись находится вне границ поля, то проверка автоматически завершается с отрицательным результатом. Далее вычисляется точка R_i по формуле $R_i = (h \cdot s_i)P + (r \cdot s_i)key$ с помощью функции compute, которая перемножает две пары точек и складывает результаты. Тут же производится еще одна проверка: если R_i обратилось в ноль, подпись не подтверждается. В противном случае берется x-координата R_i и сС помощью функции toInt преобразовывается в целочисленный тип для последующей операции деления по модулю. Наконец, результат вычислений сравнивается с подписью. Если результаты сходятся, подпись подтверждается.

Функция представлена на рисунке 28.

```
bool ECDSAverify(SignatureECDSA& S, Integer& m,
                 ECpt& P, ECpt& R, ECgroup& EC, Integer& n);
{
    if (S.r > n-1 || S.r < 1 || S.s > n-1 || S.s < 1)
    {
        return false;
    }

    Integer h, w, u1, u2;
    ECpt X;
    h = H(m, n);

    inv(w, S.s, n);
    u1 = (h * w) % n;
    u2 = (S.r * w) % n;
    EC.compute(X, u1, P, u2, R, n);

    if (X.identity)
    {
        return false;
    }

    toInt(h, X.x, n, ECpt);
    w = h % n;

    if(w == S.r)
    {
        return true;
    }

    return false;
}
```

Рисунок 28 – Функция ECDSAverify

Выводы по итогам третьей главы: для реализации выбран язык C++ с библиотекой Givaro, эллиптические кривые реализованы как структура классов, реализован алгоритм ECDH, состоящий из одной функции, реализован алгоритм ECDSA, состоящий из двух функций.

Заключение

Выпускная квалификационная работа была посвящена реализации и исследованию криптографических алгоритмов на эллиптических кривых. Поставленная цель достигнута. Задачи, поставленные в начале работы, были выполнены.

В ходе выполнения работы из теории эллиптических кривых была выделены значимые части, стоящие в основе криптографии на эллиптических кривых.

Были проанализированы фундаментальные криптографические алгоритмы, основанные на эллиптических кривых. В том числе были рассмотрены направление обмена ключами на примере алгоритма ECDH, направление цифровой подписи на примере алгоритма ECDSA и направление шифрования на примере смешанного алгоритма ECIES.

Для программной реализации были выбраны алгоритмы ECDH и ECDSA. На основе объектно-ориентированного программирования были разработаны классы Fld, ElC, ECpt, ECgroup, реализующие программно свойства конечных полей, эллиптических кривых, точек на эллиптических кривых, и групп эллиптических кривых над конечными полями соответствуенно.

Используя вышеописанные классы, были реализованы ранее рассмотренные алгоритмы обмена ключами ECDH, состоящий из одной функции, и цифровой подписи ECDSA, состоящий из двух функций: функции подписи и функции верификации.

Результатом работы является заголовочная библиотека классов, написанная на языке C++, которая реализует математические свойства эллиптических кривых и может использоваться для дальнейшего построения алгоритмов, а также уже реализованные алгоритмы ECDH и ECDSA.

Список используемой литературы

1. Aumasson, J-P. Serious Cryptography: a Practical Introduction to Modern Encryption / J-P. Aumasson No Starch Press. - 2018.
2. Bancila, M. Modern C++ Programming Cookbook / M. Bancila Packt Publishing. - 2017.
3. Bancila, M. The Modern C++ Challenge / M. Bancila Packt Publishing. - 2018.
4. Easttom W. Modern Cryptography / W. Easttom Applied Mathematics for Encryption and Information Security. - Springer. - 2021.
5. Khaleel, A. Emerging Security Algorithms and Techniques / A. Khaleel CRC Press, Taylor & Francis Group. - 2019.
6. Neapolitan, R. E. Foundations of Algorithms. / R. E. Neapolitan Jones & Bartlett Learning. - 2015.
7. Shemanske T. R. Modern Cryptography and Elliptic Curves: a Beginners Guide / T. R. Shemanske American Mathematical Society. - 2017.
8. Welschenbach, M. Cryptography in C++ and C / M. Welschenbach Apress. - 2013.
9. Yang H. EC Cryptography Tutorials - Herong's Tutorial Examples / H. Yang. - 2019.
10. Friedl S. An Elementary Proof of the Group Law for Elliptic Curves. / S. Friedl - Groups Complexity Cryptology 10(1). - 2017. - pp. 117–123.
11. Harkanson, R. Applications of Elliptic Curve Cryptography / R. Harkanson, Y. Kim Proceedings of the 12th Annual Conference on Cyber and Information Security Research. - 2017. - pp. 1–7.
12. Koblitz, A. H. Elliptic Curve Cryptography: The Serpentine Course of a Paradigm Shift / A. H. Koblitz Journal of Number Theory 131(5). - 2011. - pp. 781–814.

13. Kodali R. K. Energy Efficient ECC Encryption Using ECDH / R. K. Kodali, N. V. S. Narasimha Sarma. Lecture Notes in Electrical Engineering Emerging Research in Electronics, Computer Science and Technology. - 2013. - pp. 471–478.

14. Setiadi I. Elliptic curve cryptography: Algorithms and implementation analysis over coordinate systems / I. Setiadi, A. I. Kistijantoro, A. Miyaji 2015 2nd International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA). - 2015. - pp. 1-6.

15. The Elliptic Curve Diffe-Hellman (ECDH) // Портал KoçLab [Электронный ресурс]. URL: http://koclab.cs.ucsb.edu/teaching/ecc/project/2015Projects/Haakegaard+Lang.pdf (дата обращения: 01.06.2021).

16. The Elliptic Curve Digital Signature Algorithm (ECDSA) // Портал кафедры информационных технологий университета Майами [Электронный ресурс]. URL: https://www.cs.miami.edu/home/burt/learning/Csc609.142/ecdsa-cert.pdf (дата обращения: 01.06.2021).

17. The Elliptic Curve Group Law // Портал кафедры математики Брауновского университета [Электронный ресурс]. URL: https://www.math.brown.edu/reschwar/M1540B/elliptic.pdf (дата обращения: 01.06.2021).

18. The Euclidean Algorithm and Multiplicative Inverses // Портал кафедры математики университета Уты [Электронный ресурс]. URL: https://www.math.utah.edu/~fguevara/ACCESS2013/Euclid.pdf (дата обращения: 01.06.2021).

19. Givaro: Givaro Documentation // Документация Givaro в электронном виде [Электронный ресурс]. URL: https://casys.gricad-pages.univ-grenoble-alpes.fr/givaro/givaro-html/index.html (дата обращения: 01.06.2021).

20. Schoof’s algorithm // Портал кафедры математики Массачусетского технологического института [Электронный ресурс]. URL:

https://math.mit.edu/classes/18.783/2015/LectureNotes9.pdf (дата обращения: 01.06.2021).